Table of Contents

[API MANAGEMENT 3](#_Toc134180609)

[API MANAGEMENT COMPONENTS 3](#_Toc134180610)

[API GATEWAY 3](#_Toc134180611)

[MANAGEMENT PLANE 3](#_Toc134180612)

[DEVELOPER PORTAL 3](#_Toc134180613)

[CREATING AN APIM 3](#_Toc134180614)

[CREATING AN API 4](#_Toc134180615)

[PRODUCT 10](#_Toc134180616)

[CREATING A PRODUCT 11](#_Toc134180617)

[PERMISSION TO THE PRODUCT(FOR GUEST USER) 12](#_Toc134180618)

[ACCESSING THE API FROM PRODUCT FOR DEV PORTAL(AS GUEST) 12](#_Toc134180619)

[SUBSCRIPTION APPROVAL 13](#_Toc134180620)

[PRODUCT POLICIES 18](#_Toc134180621)

[MOCKING API RESPONSES 19](#_Toc134180622)

[STEP 1: SET UP the API TO BE MOCKED 19](#_Toc134180623)

[STEP 2: SET UP THE MOCKED the RESPONSE 20](#_Toc134180624)

[STEP 3: ADD POLICY TO MOCK RESPONSE 21](#_Toc134180625)

[STEP 4: TEST THE API 21](#_Toc134180626)

[THROTTLING 22](#_Toc134180627)

[API REVISIONS 22](#_Toc134180628)

[CREATING API REVISIONS 23](#_Toc134180629)

[VERSIONS 24](#_Toc134180630)

[VERSION VERSUS REVISIONS 24](#_Toc134180631)

[CREATING VERSIONS 25](#_Toc134180632)

[ADDING VERSIONED API TO PRODUCT 25](#_Toc134180633)

[TESTING VERSIONED API 26](#_Toc134180634)

[API MANAGEMENT POLICIES 26](#_Toc134180635)

[AUTHENTICATION POLICIES 26](#_Toc134180636)

[ACCESS RESTRICTION POLICIES 27](#_Toc134180637)

[CACHING POLICIES 27](#_Toc134180638)

[ADDING CACHE POLICY 27](#_Toc134180639)

[TESTING THE CACHINGd 28](#_Toc134180640)

[CROSS DOMAIN POLICIES 28](#_Toc134180641)

[TRANSFORMATION POLICIES 28](#_Toc134180642)

[ADVANCED POLICIES 29](#_Toc134180643)

[DEBUGGING POLICIES 29](#_Toc134180644)

# API MANAGEMENT

* API Management helps organizations publish APIs to external, partner, and internal developers to unlock the potential of their data and services.

## API MANAGEMENT COMPONENTS

Azure API Management is made up of an

* **API GATEWAY,**
* **A MANAGEMENT PLANE**
* **A DEVELOPER PORTAL**.

### API GATEWAY

The **API gateway** is the endpoint that:

* Accepts API calls and routes them to appropriate backends
* Verifies API keys and other credentials presented with requests
* Enforces usage quotas and rate limits
* Transforms requests and responses specified in policy statements
* Caches responses to improve response latency and minimize the load on backend services
* Emits logs, metrics, and traces for monitoring, reporting, and troubleshooting

### MANAGEMENT PLANE

The **management plane** is the administrative interface where you set up your API program. Use it to:

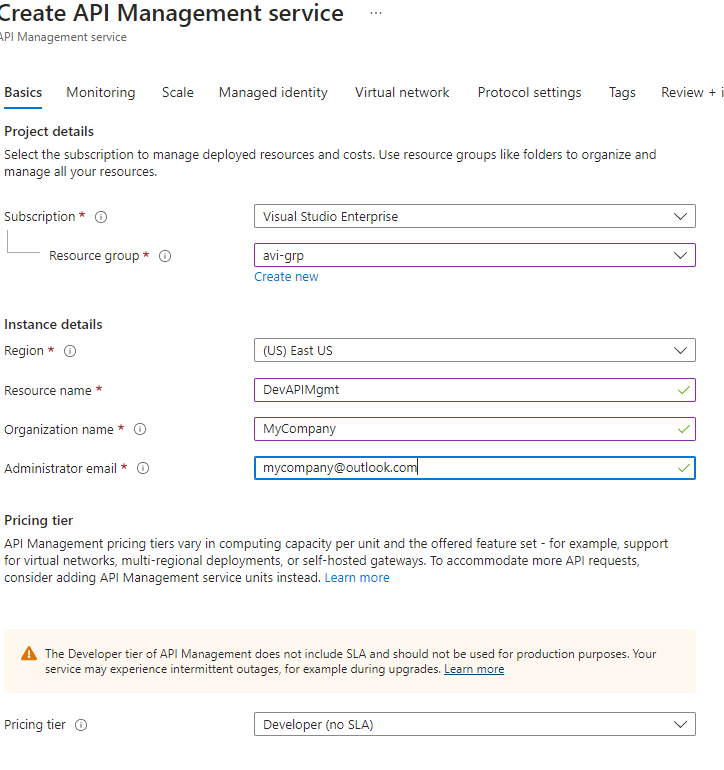
* Provision and configure API Management service settings
* Define or import API schema
* Package APIs into products
* Set up policies like quotas or transformations on the APIs
* Get insights from analytics
* Manage users

### DEVELOPER PORTAL

The **Developer portal** is an automatically generated, fully customizable website with the documentation of your APIs. Using the developer portal, developers can:

* Read API documentation
* Call an API via the interactive console
* Create an account and subscribe to get API keys
* Access analytics on their own usage
* Download API definitions
* Manage API key

## CREATING AN APIM

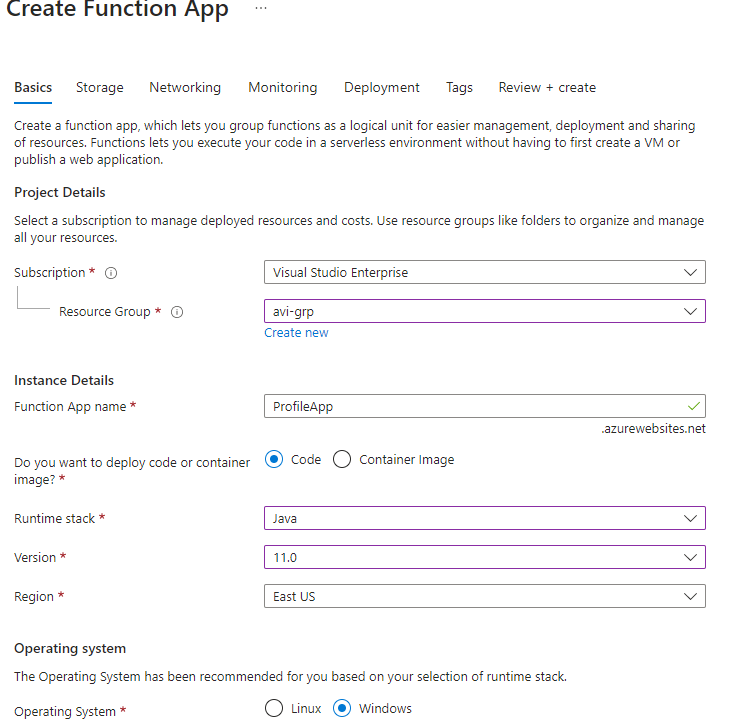


### CREATING AN API

* Use case: Profile Management – where user can create , update, delete and retrieve profiles
* To create an API in APIM we can make use multiple Azure services like Azure function Apps. In this example -we will make use of function app to create API.

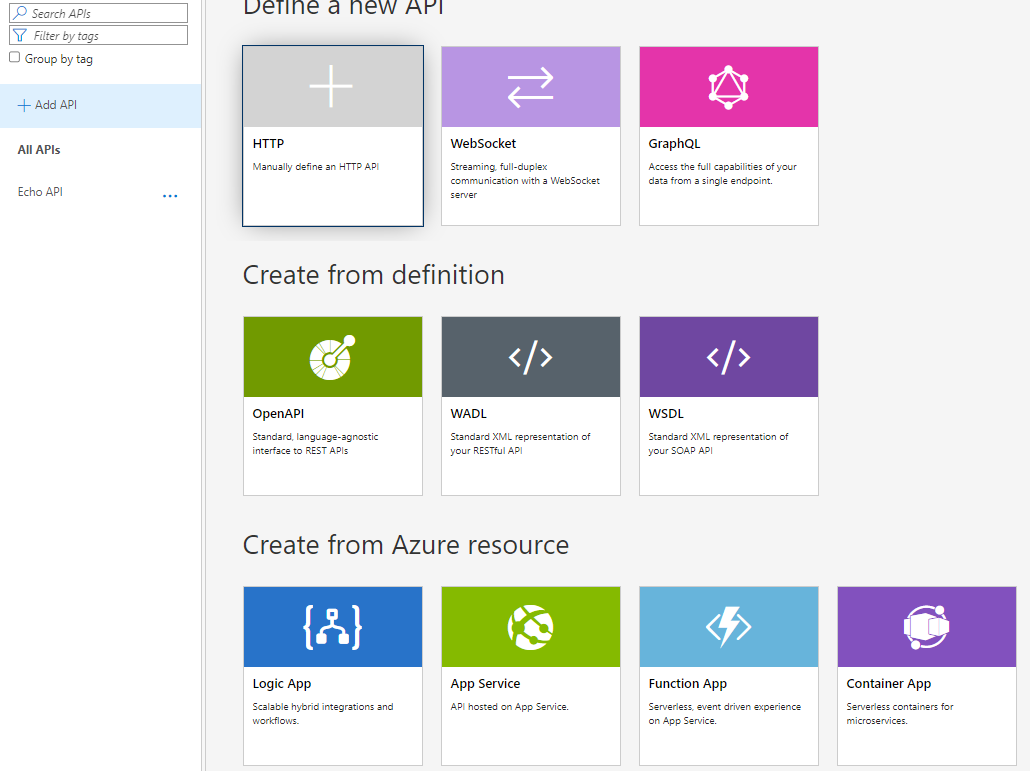
#### STEP 1: CREATE A FUNCTION APP

* Create a Function APP and create Functions within the function app



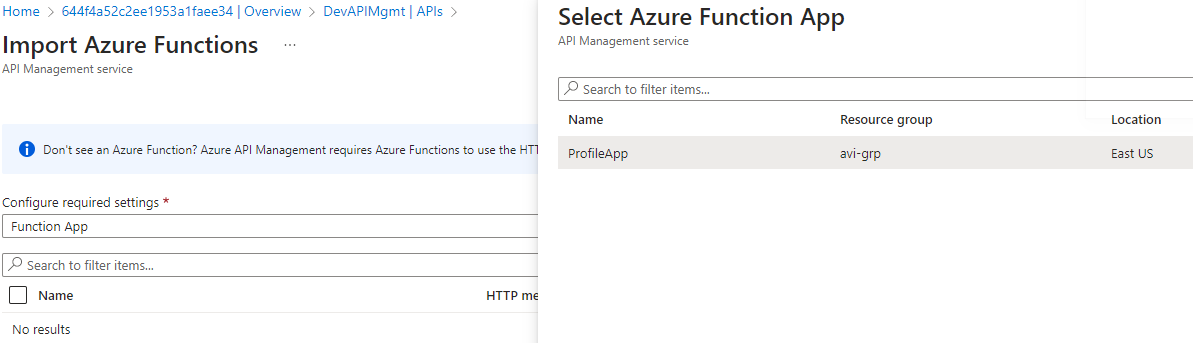
#### STEP 2: CREATE AN API

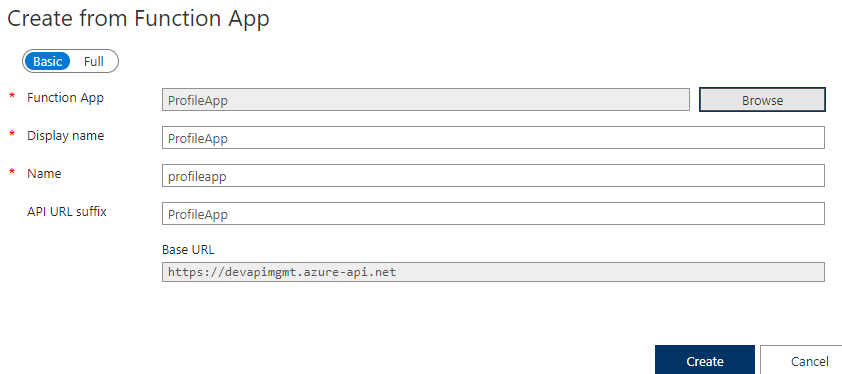
* The API below will be created based on Function App
* The API is Profile API (CRUD for User Profile)



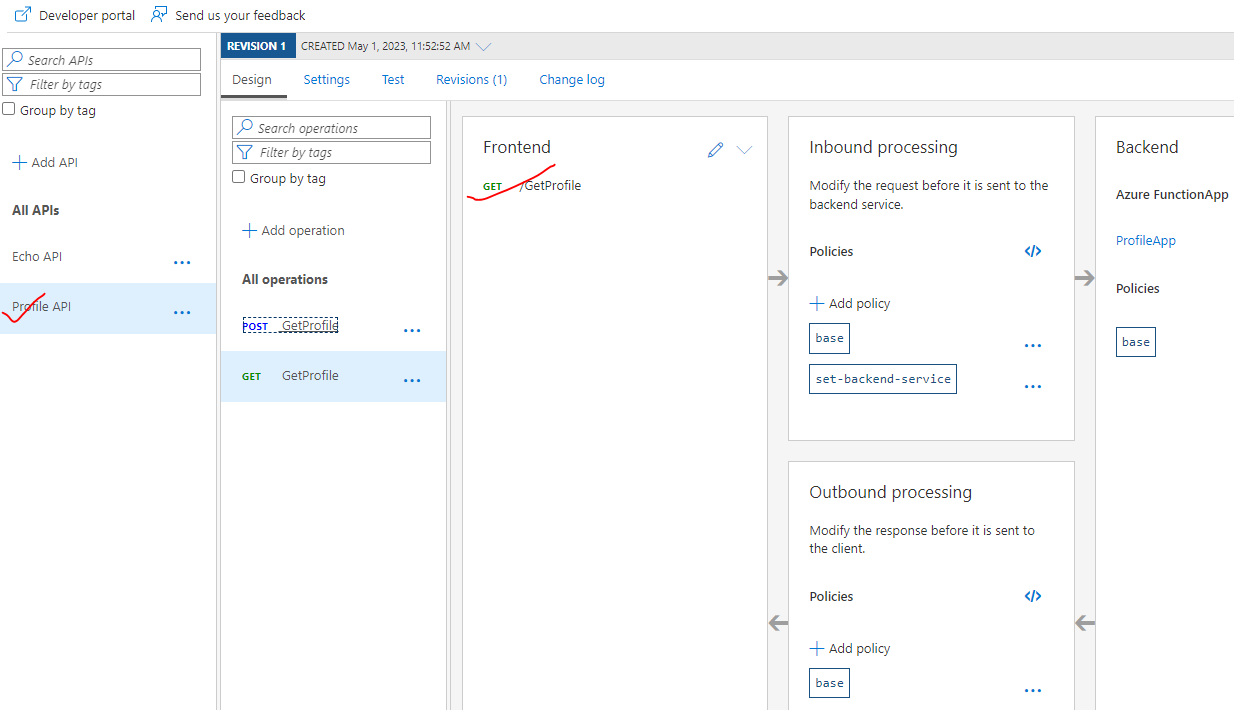
Select and import the function app





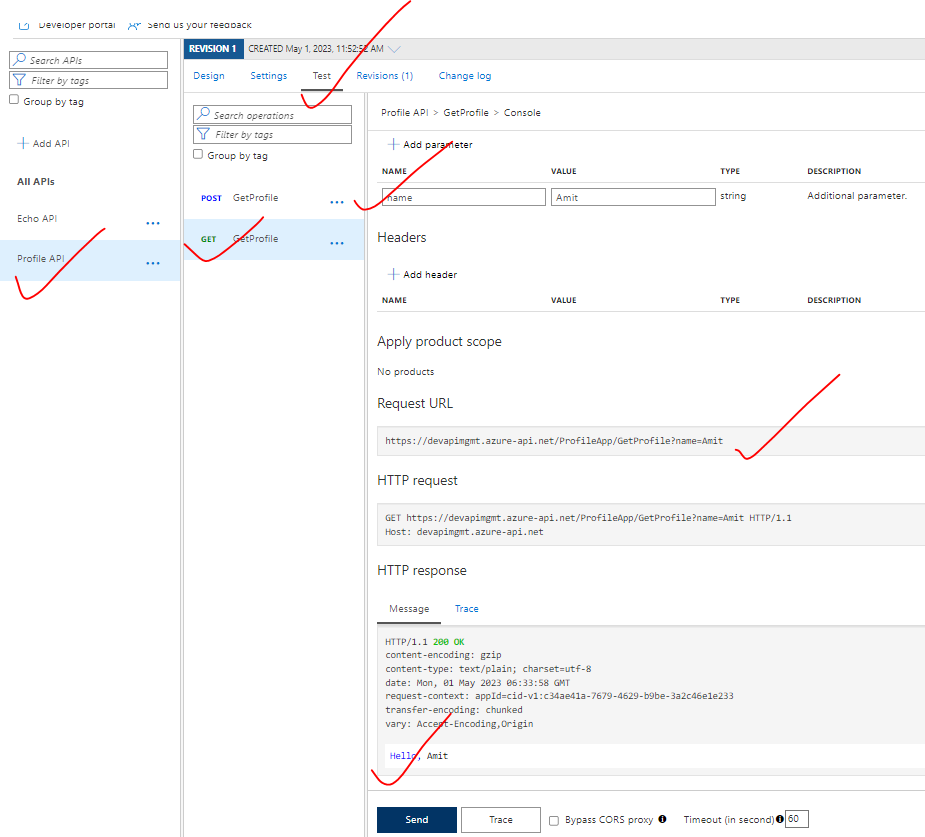


* The Profile API has been added to APIM which has GET and POST operations
* The below diagram shows that the **Profile API** has been built on top of **GetProfile** function app



#### STEP 3: TEST THE API

* We will be testing the API via APIM / API Gateway
* Select the API 🡪 Select Operation 🡪 Test 🡪 Enter Require Parameters 🡪 Send



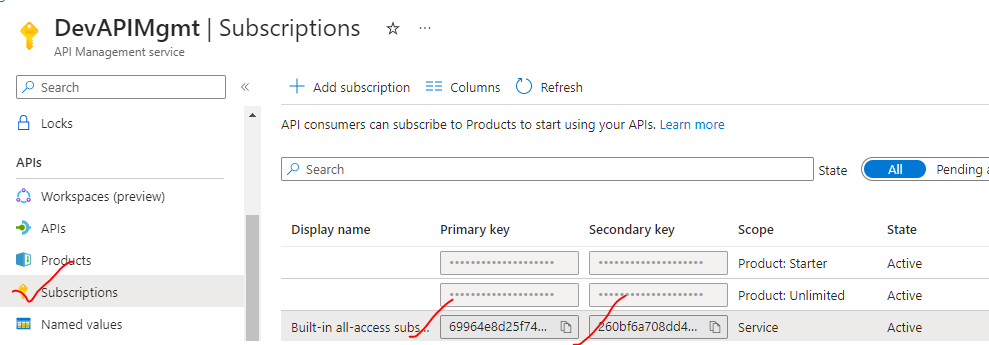
#### TEST THE API USING DEVELOPER PORTAL

|  |  |
| --- | --- |
|  |  |

* Note to test an API from APIM / Developer portal need to make use of subscription key.

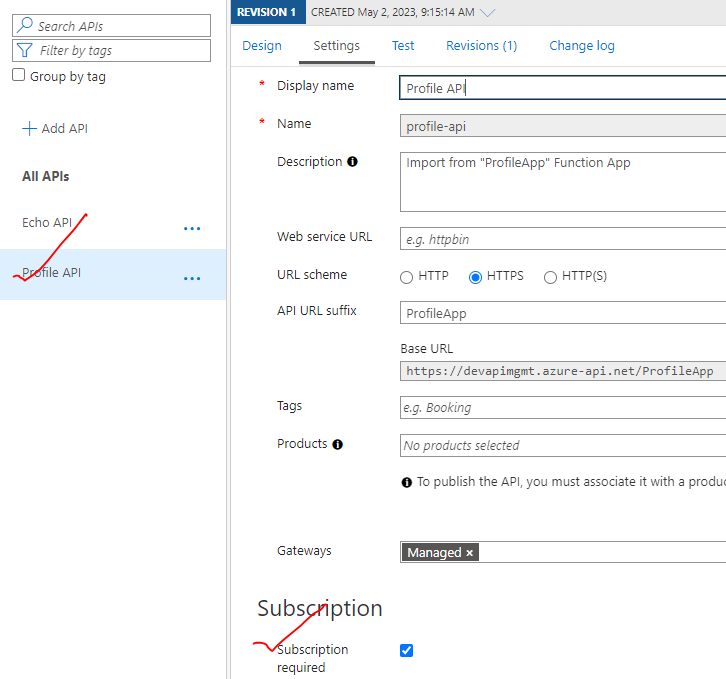
**GET SUBSCRIPTION KEY**

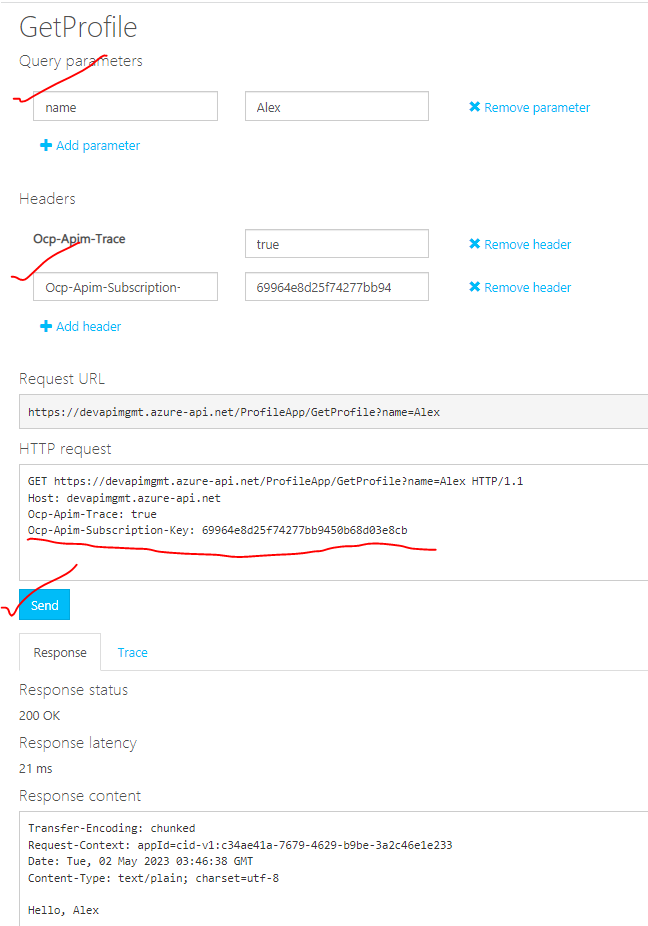
* Go to APIM 🡪 Subscriptions 🡪 We can make use of either primary or secondary subscription key
* Copy either of the key and add it in the API request in Developer portal.



|  |  |
| --- | --- |
| HEADER NAME | HEADER VALUE |
| Ocp-Apim-Subscription-Key | Primary / secondary key from APIM subscription |

* The API need subscription because the “Subscription Required” option is enabled. Note – if we uncheck it we can able to call the API without “Ocp-Apim-Subscription-Key” header. However it is not recommended in PROD env.



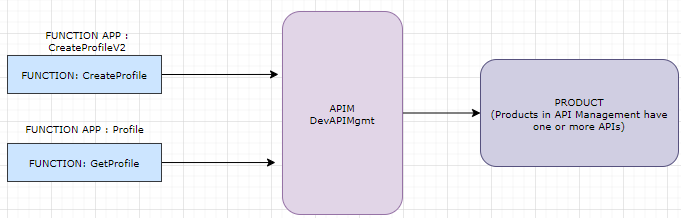


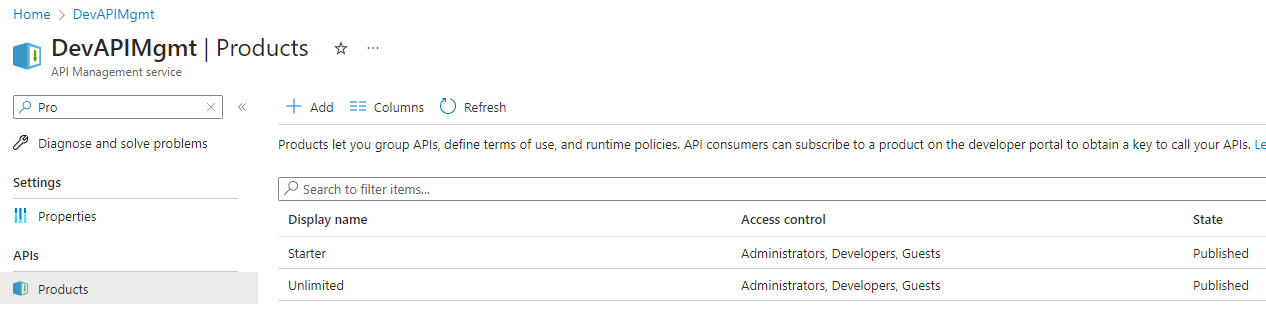
## PRODUCT

* Products in API Management have one or more APIs, and are configured with a title, description, and terms of use.
* Products can be **Open** or **Protected**.

|  |  |
| --- | --- |
| **OPEN** | * Open products can be used without a subscription |
| **PROTECTED** | * Protected products must be subscribed to before they can be used |

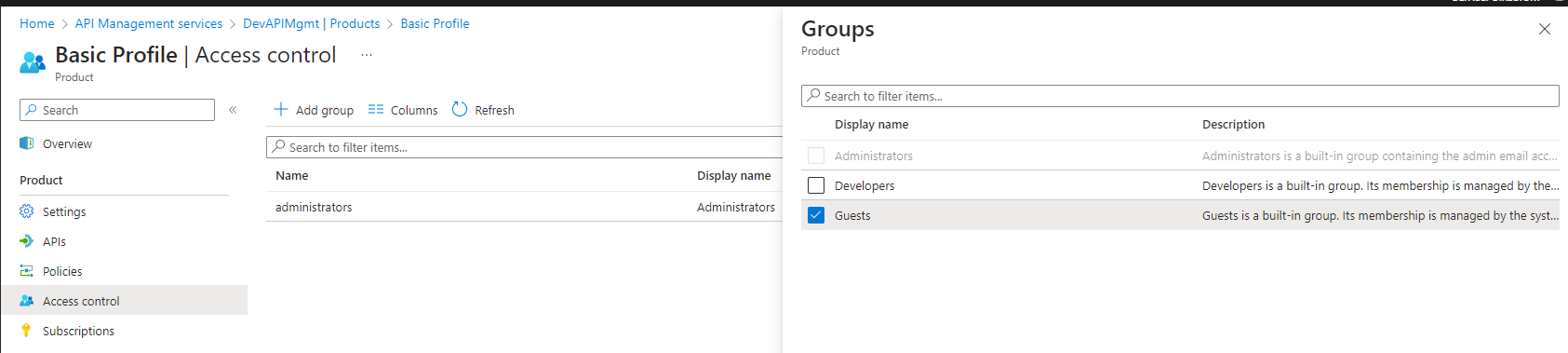
### CREATING A PRODUCT

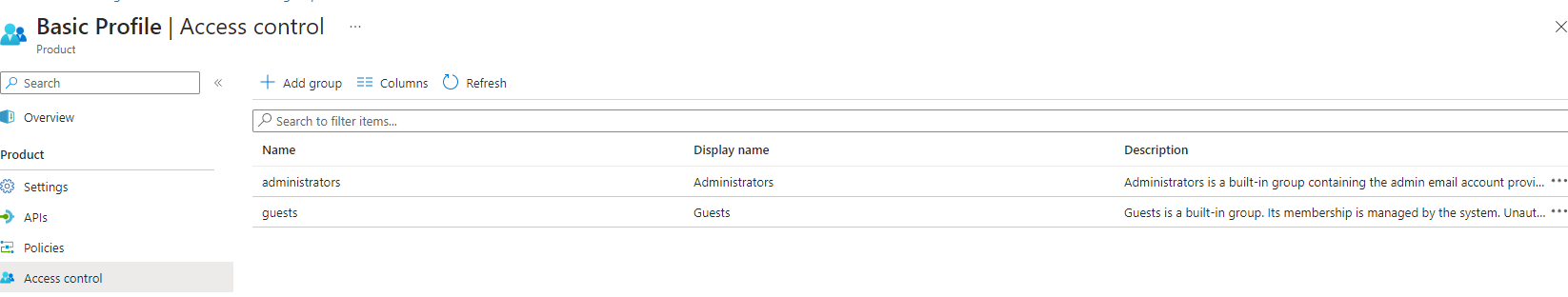




|  |  |
| --- | --- |
|  | * Give the display name * Select APIs which we want to be a part of Product. * To product cane be accessed from developer portal   Note –  **AS AN ADMIN USER**   * We can able to view all the product , * The subscription key will be already pre-populated for all the APIs. That’s one of the benefit of adding API in a Producy   **AS GUEST USER**   * But to view the products as guest / anonymous user we need to provides access on the product(as below) |

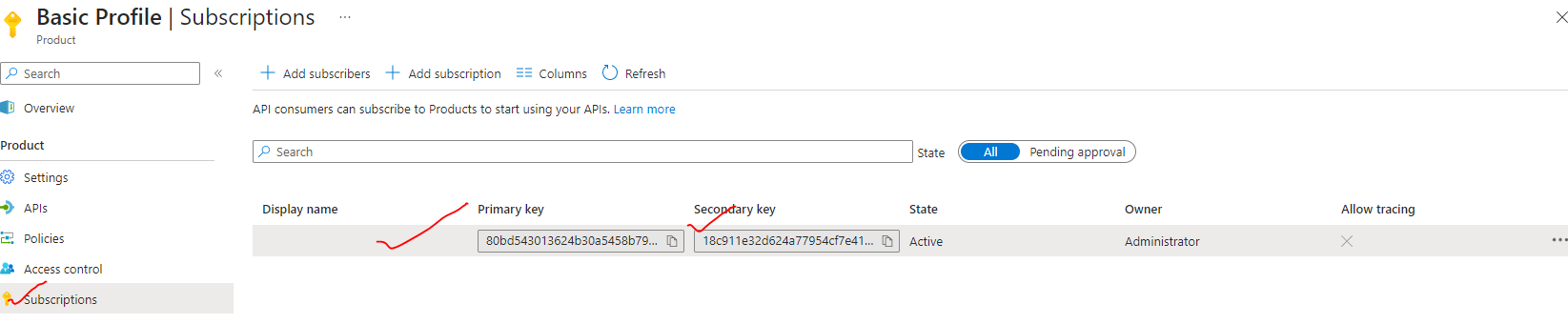
### PERMISSION TO THE PRODUCT(FOR GUEST USER)

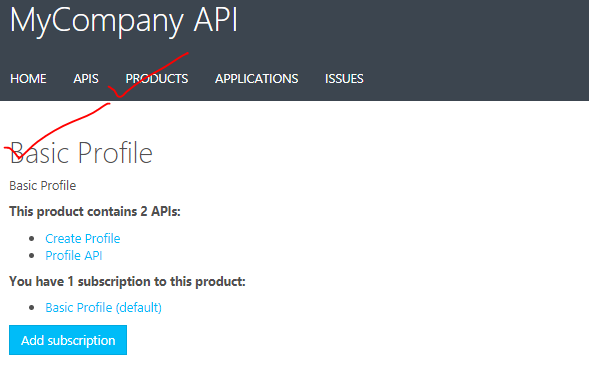


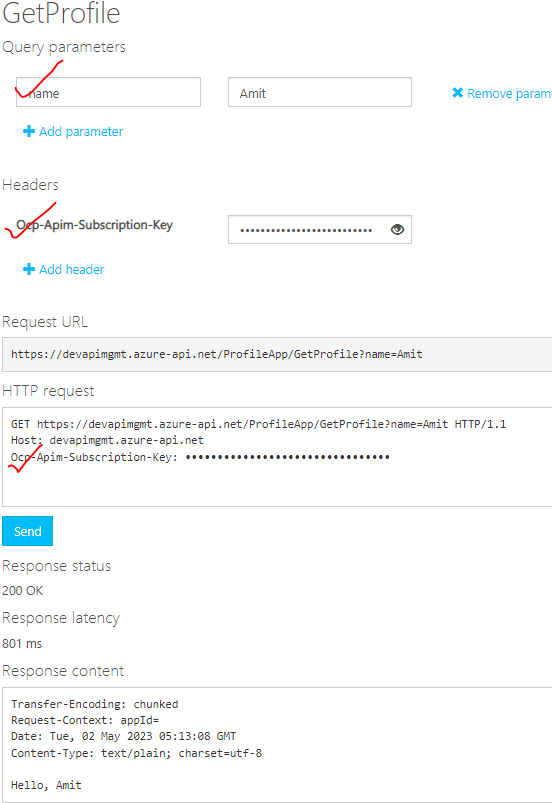


### ACCESSING THE API FROM PRODUCT FOR DEV PORTAL(AS GUEST)

* To access the API in a “Product” 🡪 Use Primary / Secondary key of the Product(show below)





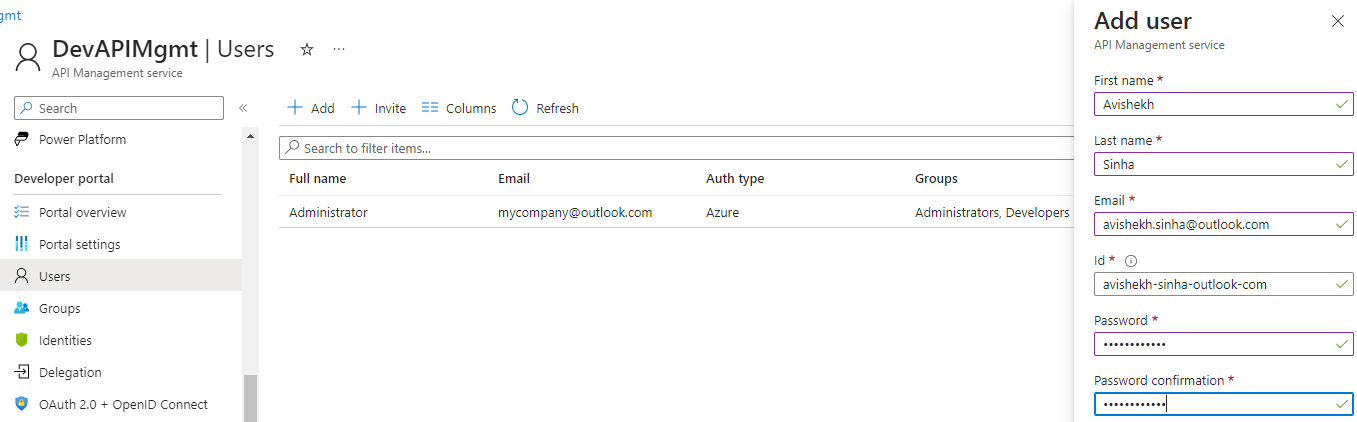


### SUBSCRIPTION APPROVAL

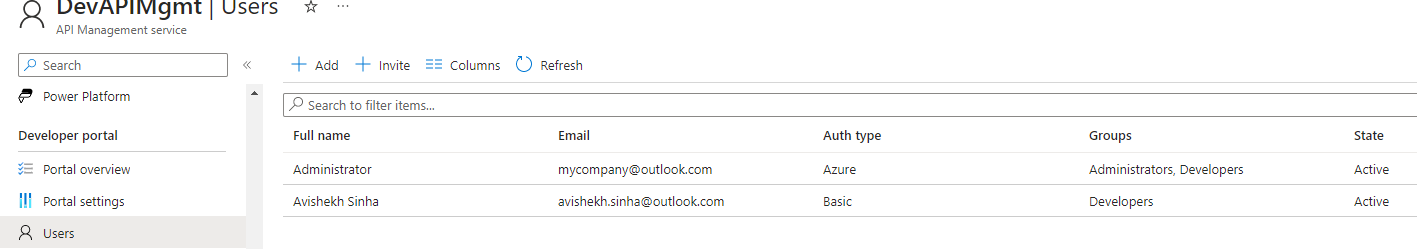
* Subscription approval is configured at the product level and can either require administrator approval, or be auto approved.

|  |  |
| --- | --- |
|  | USE CASE   * Create a user inside a product.(Developer Group) * Provider access to the product * To use the API user must request for the subscription * The request can be auto approved or manual approval by the admin. The setting is based on following configuration. |

#### STEP 1: CREATING USER IN PRODUCT

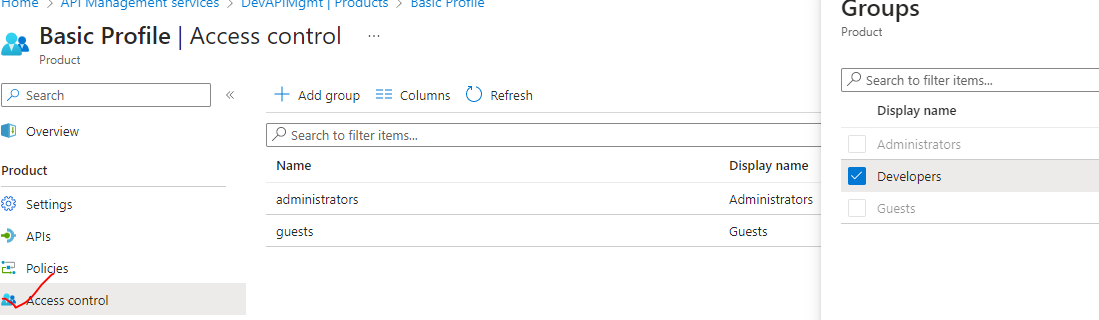


* User Added in Developer Group. Hence to give access to the user we need to give access of product to the developer group.



#### STEP 2: ACCESS TO USER ON PRODUCT

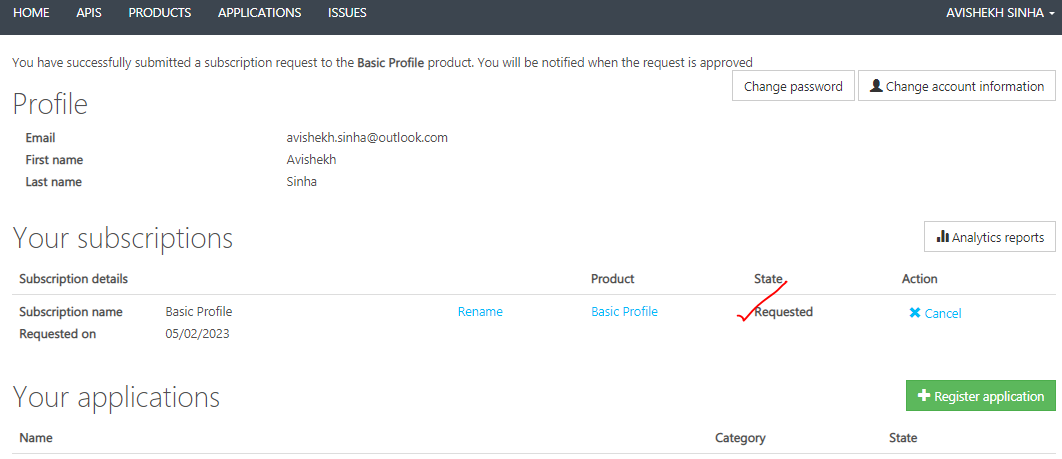
* Access given on product to the developer group



#### STEP 3: REQUEST FOR SUBSCRIPTION

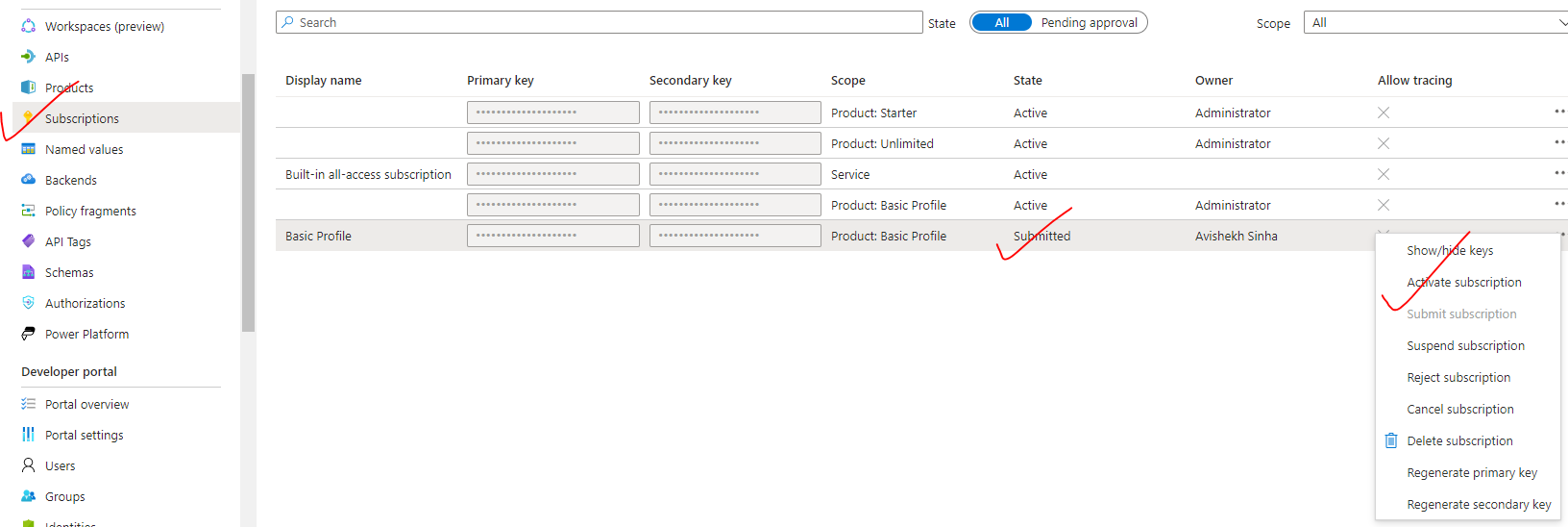
|  |  |
| --- | --- |
| * Sign in with the credential of the new user to view the product | * To make of APIs in the product -the user need go subscribe for it. |

SUBSCRIPTION REQUESTED



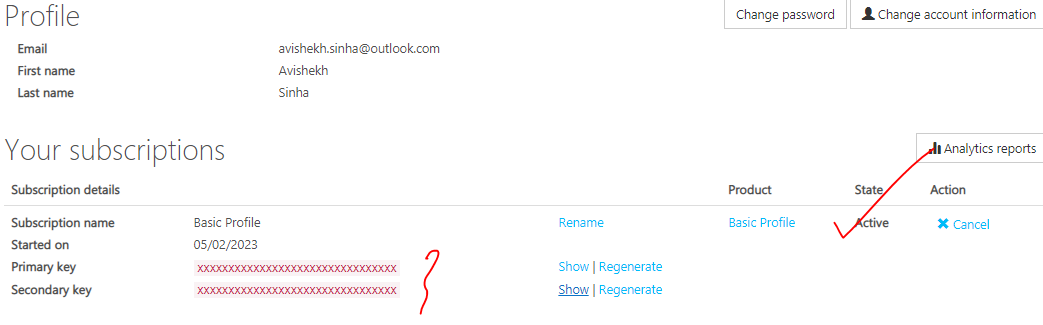
|  |  |
| --- | --- |
|  | Note:   * A user can request for multiple subscription for the same product. To limit the number of subscriptions we can configured using “subscription count limit” |

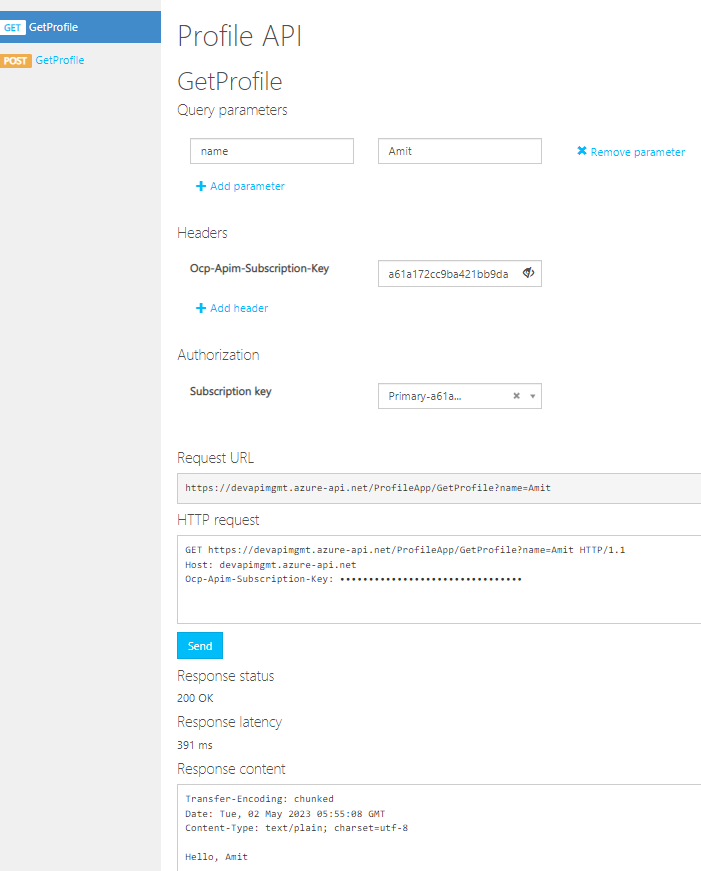
#### STEP 4: SUBSCRIPTION APPROVAL



#### STEP 5: ACCESSING THE API APPROVAL

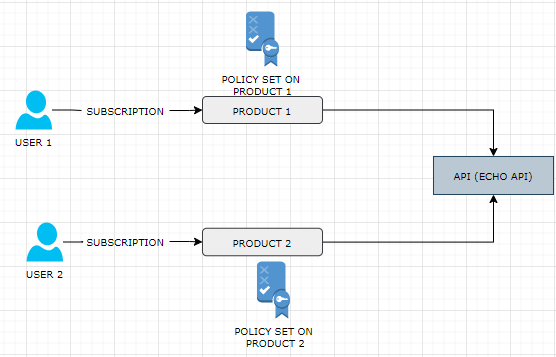
* USER CAN NOW SEE THE PRIMARY / SECONDARY KEY AFTER APPROVAL .
* Navigate 🡪 Product 🡪 Basic Profile(Product Name) 🡪 API , to access the API. Note the key will already be pre-populated for the user.





### PRODUCT POLICIES

* The product policies can be OUTBOUND or INBOUND
* Let’s consider below use case – We have 2 products in the APIM which has same API configured. Each product has its won policy (inbound/outbound).



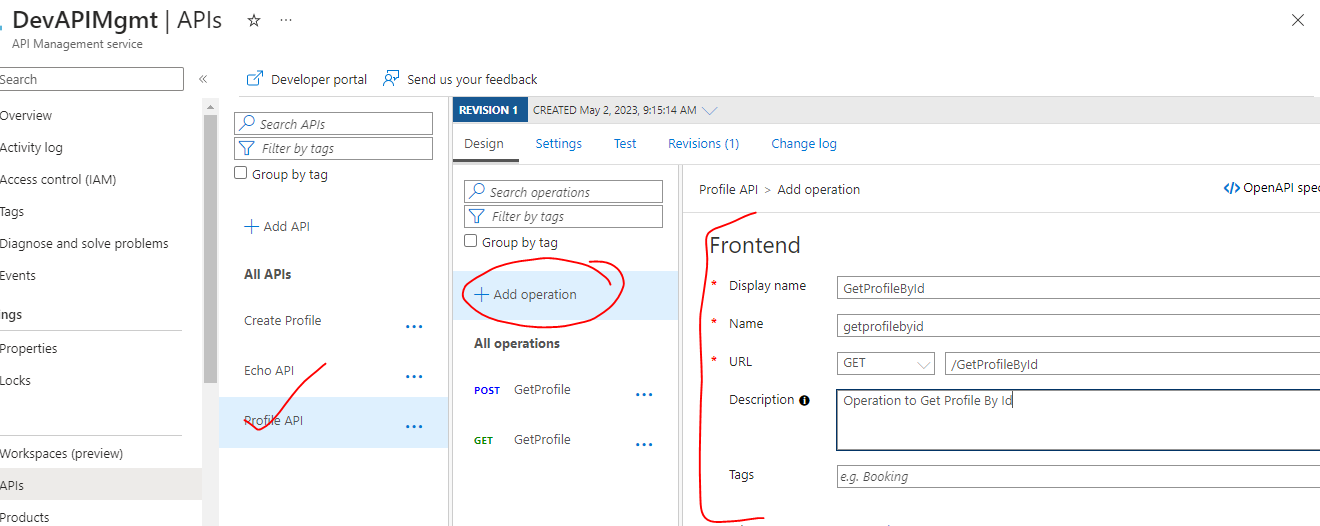
|  |  |
| --- | --- |
| POLICY ON PRODUCT -1 | POLICY ON PRODUCT -2 |
| <policies>      <inbound>          <rate-limit calls="5" renewal-period="60" />          <quota calls="100" renewal-period="604800" />          <base />      </inbound>      <backend>          <base />      </backend>      <outbound>          <base />      </outbound>      <on-error>          <base />      </on-error>  </policies> | <policies>      <inbound>          <base />      </inbound>      <backend>          <base />      </backend>      <outbound>          <base />      </outbound>      <on-error>          <base />      </on-error>  </policies> |

* Hence the policy says – a user subscribed to **product-1** can able to call the “Echo API” 5 times in an hour(rate limit) but the user subscribed to **product-2** can make unlimited number or calls to API

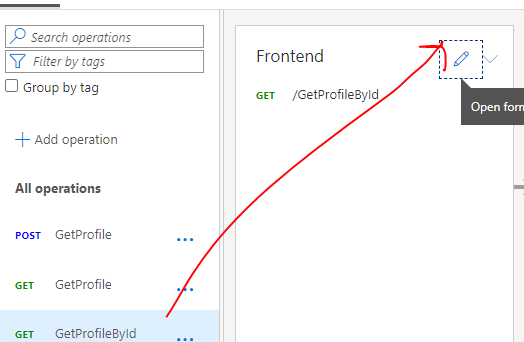
## MOCKING API RESPONSES

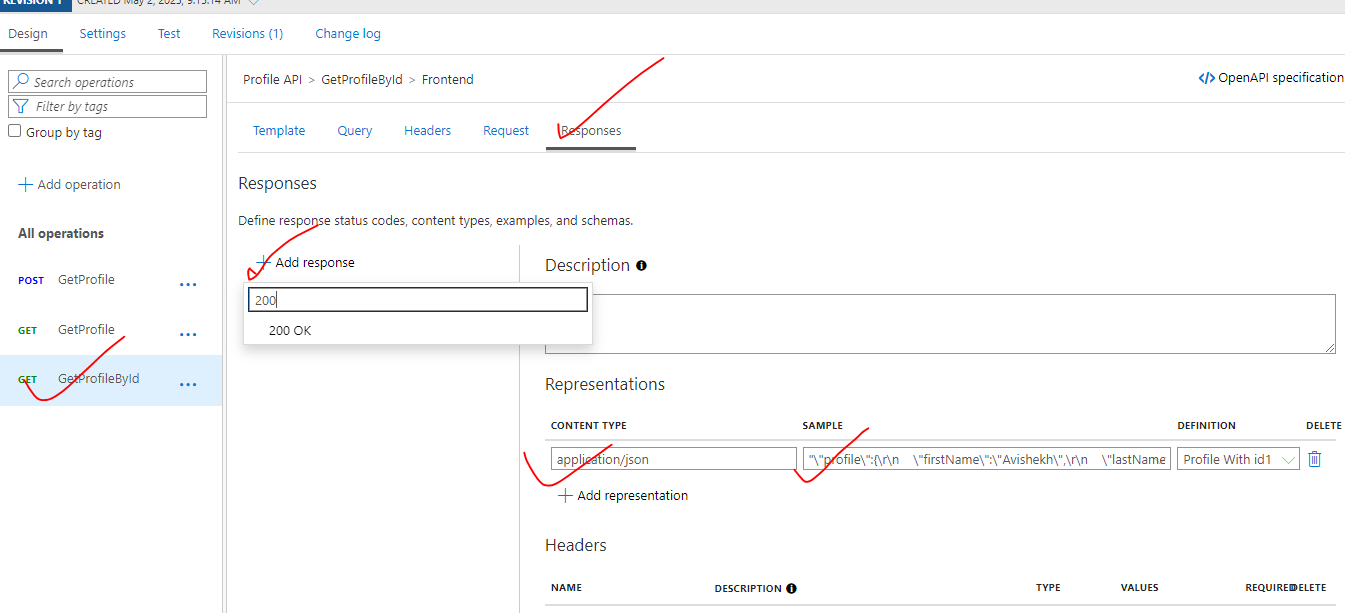
* We can mock the response of the API response (response of a API operations) . This is useful when the backend API is not ready to be used.

### STEP 1: SET UP the API TO BE MOCKED

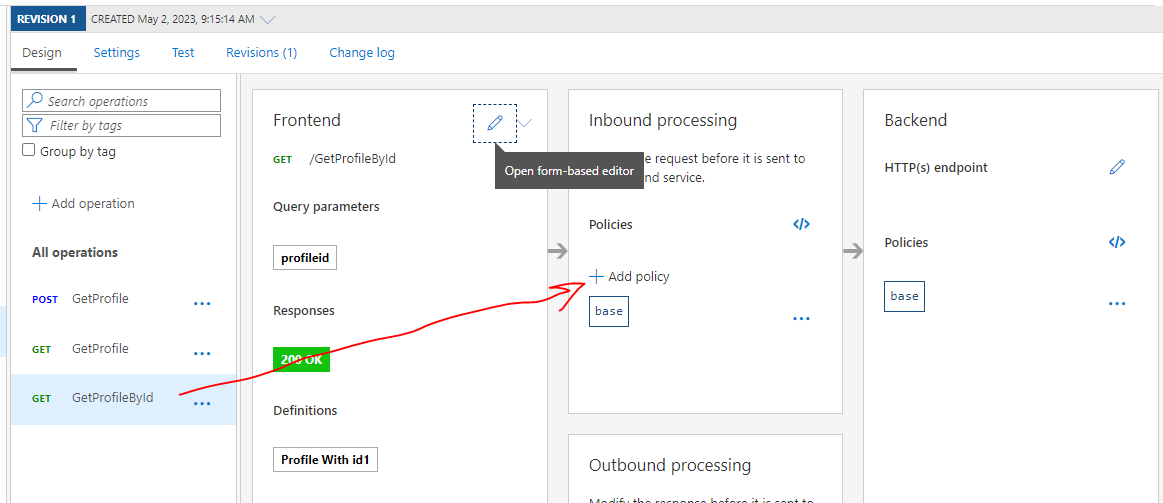


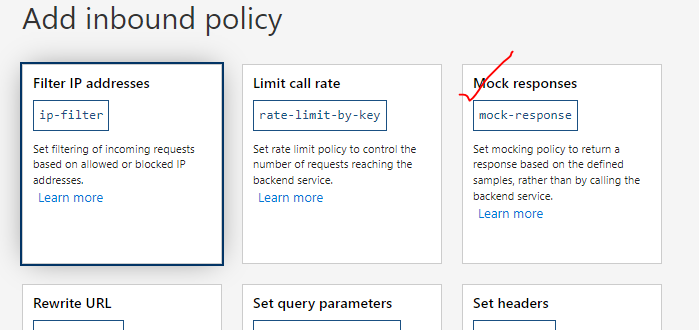
### STEP 2: SET UP THE MOCKED the RESPONSE



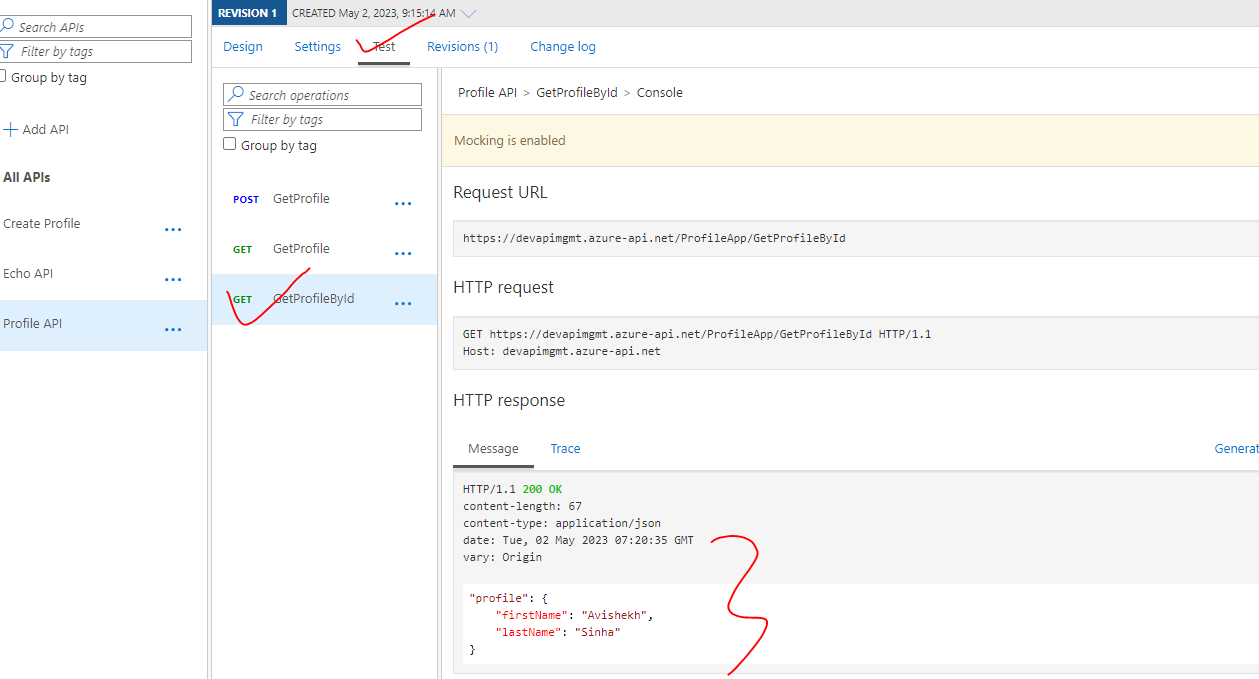


### STEP 3: ADD POLICY TO MOCK RESPONSE



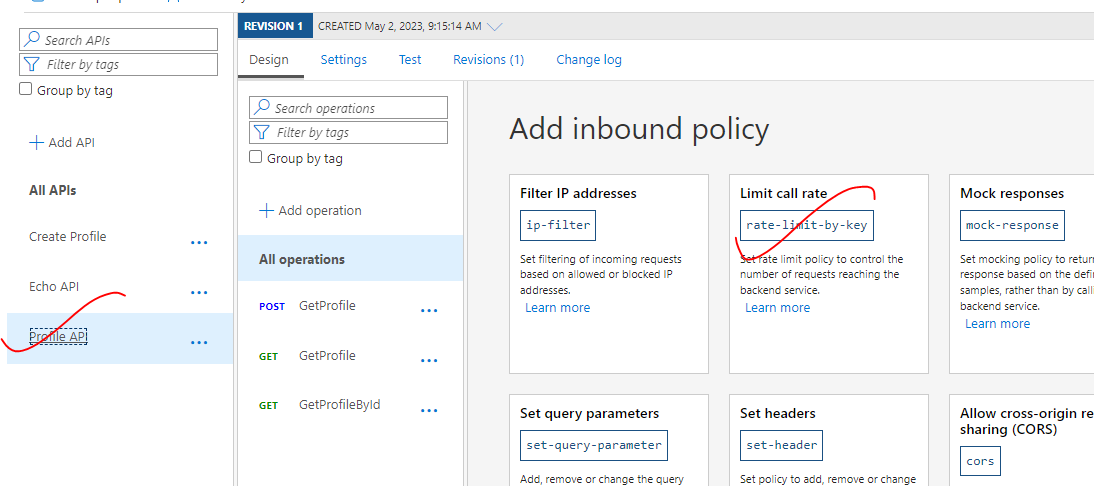


### STEP 4: TEST THE API

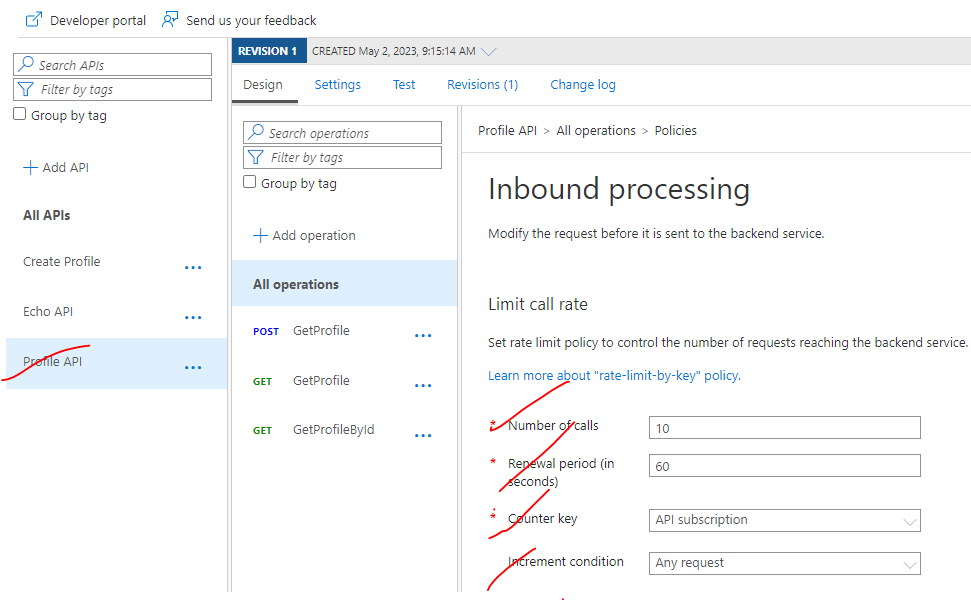


## THROTTLING

* Throttling is a protection mechanism where we can specify how many calls your API or operation or product can handle during a certain period.



* Below configuration will rate limit the incoming request. Hence – the API can accept 10 calls for 1 hours.



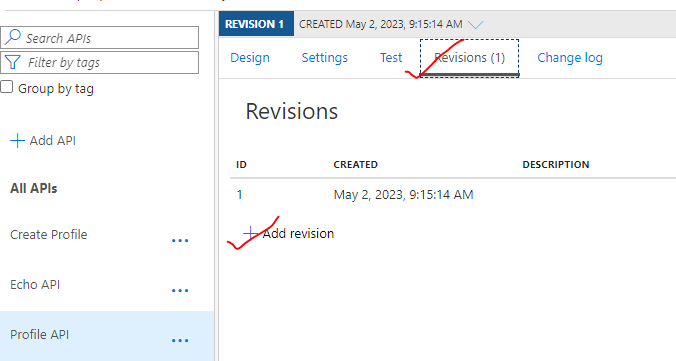
## API REVISIONS

* API management revisions allow us to update the API definition and settings and put it in a separate revision separate from the current working revision of the API.
* This is because we might have a situation where there are some developers already consuming the current revision of the API and we want to do some changes to those API definition
* To avoid making any disturbance to those developers. The best is to put our changes in a separate revision until we test it and make sure it works as expected, and then we can shift the traffic or move the traffic to the new revision.

### CREATING API REVISIONS

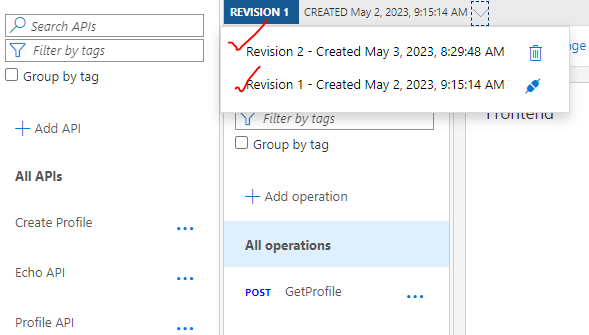
#### STEP 1: CREATE REVISION

* Select an API 🡪 Revision 🡪 Add Revision





* We can switch between the revisiom from the drop down
* Revision 1 is the current revision and Revision 2 is the new revision whuch we created . Note - Revision -2 will not be available in developer portal as its not the “current”(as shown above)



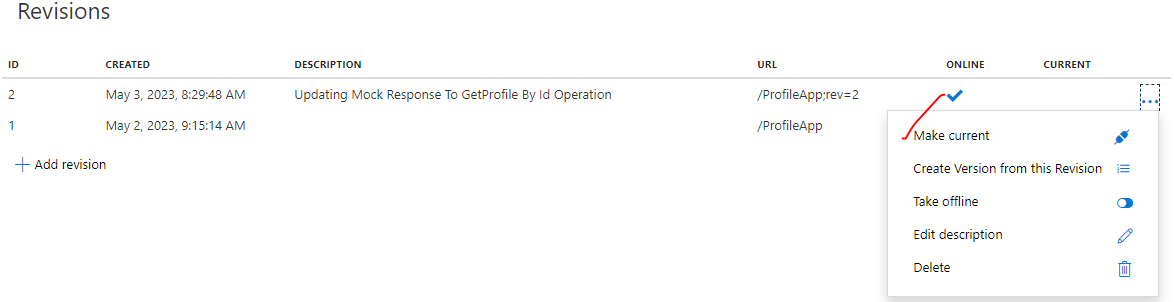
#### STEP 2: UPDATE THE API / OPERATION IN THE NEW REVISION (REVISION 2)

* To simulate this – we can make use of mocked response

|  |  |
| --- | --- |
| **MOCKED RESPONSE OF REVISION -1** | **MOCKED RESPONSE OF REVISION -2** |
|  |  |

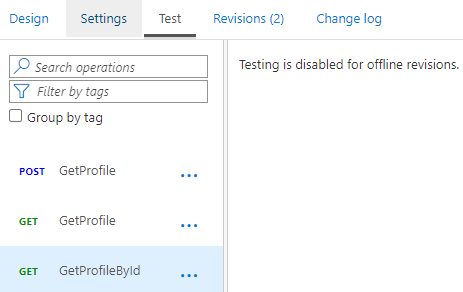
#### STEP 3: SWITCH THE NEW REVISION -2 TO CURRENT

* Make the Revision -2 as current



* **We can also take revision -1 - offline(as shown below). But if we take revision 1 offlline – it will be not be available for testing from Portals also .**





## VERSIONS

### VERSION VERSUS REVISIONS

The main difference between version and revisions is”

* We can have multiple online revisions at the same time, but we can only have one current revision for our API at a certain point of time.
* However, with versions, we can have multiple versions up and running at the same time for our consumers to use.

### CREATING VERSIONS

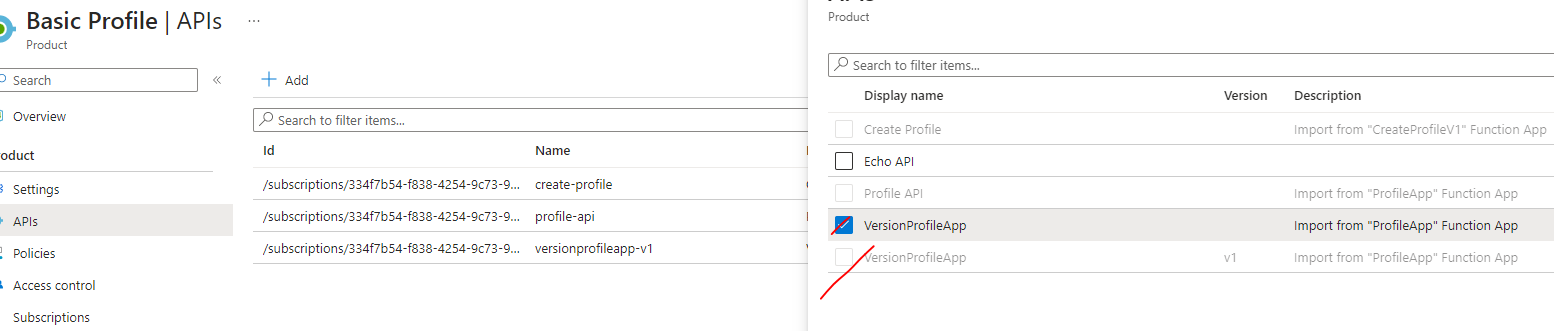
|  |  |
| --- | --- |
|  | * To create version – let create a new API that must be versioned. Lets say we name it as “VersionProfileApp” * To add new version for the API |

* While creating the version we are also adding it to an existing product. This is optional

|  |  |
| --- | --- |
|  |  |

### ADDING VERSIONED API TO PRODUCT

* Go to product 🡪 Select the product 🡪 API 🡪 Add all the version of API to make it as part of product
* We can either have both versions in the same product or we can have each version in separate products based on your structure and how you organize your APIs in your API management.
* Note – if the API is part of a product – while testing the API via developer portal – the subscription key will be pre-populated



### TESTING VERSIONED API

|  |  |
| --- | --- |
|  | * The versioned API can be tested from APIM (Design) or developer portals. * We can select the respective version to validate the APIs * As it has been added to the product – it can be tested via Products too |

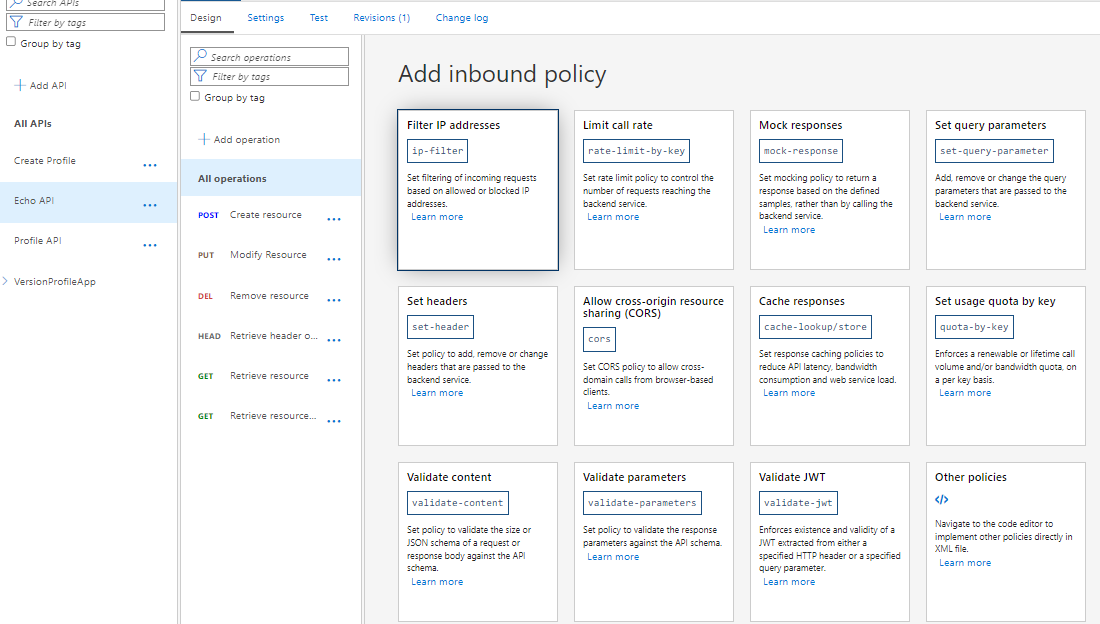
## API MANAGEMENT POLICIES

* **APIM are XML documents that has a sequence of inbound and outbound statements** that is going to control your API

The policies can be applied in following scopes

* + **GLOBAL POLICY** - This will be applied to all the APIs in the API management.
  + **PRODUCT LEVEL POLICY** – It will be applied all API is hosted or assigned in this product (e.g.like starter product that has a limit to five calls per minute.)
  + **API LEVEL POLICY** - The policy at the API level, which is going to affect all operations in this API.
  + **OPERATION LEVEL POLICY** - It's going to affect a certain operation only.

These are few commonly used inbound / outbound policies.



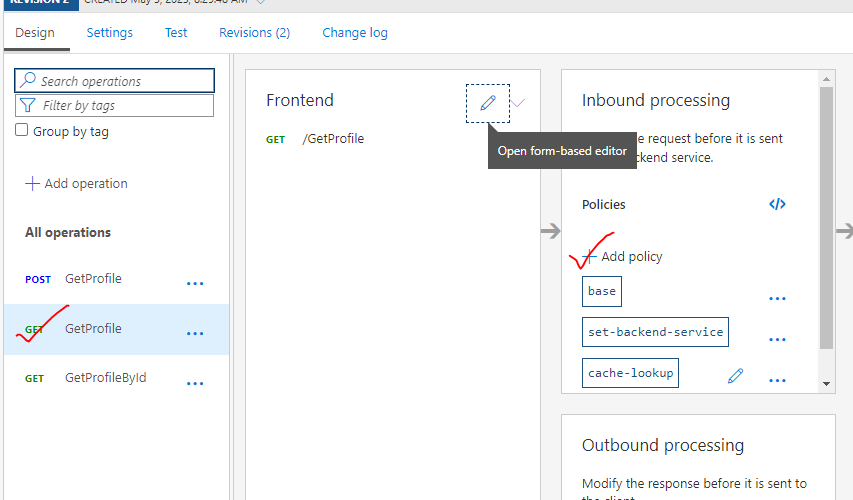
### AUTHENTICATION POLICIES

### ACCESS RESTRICTION POLICIES

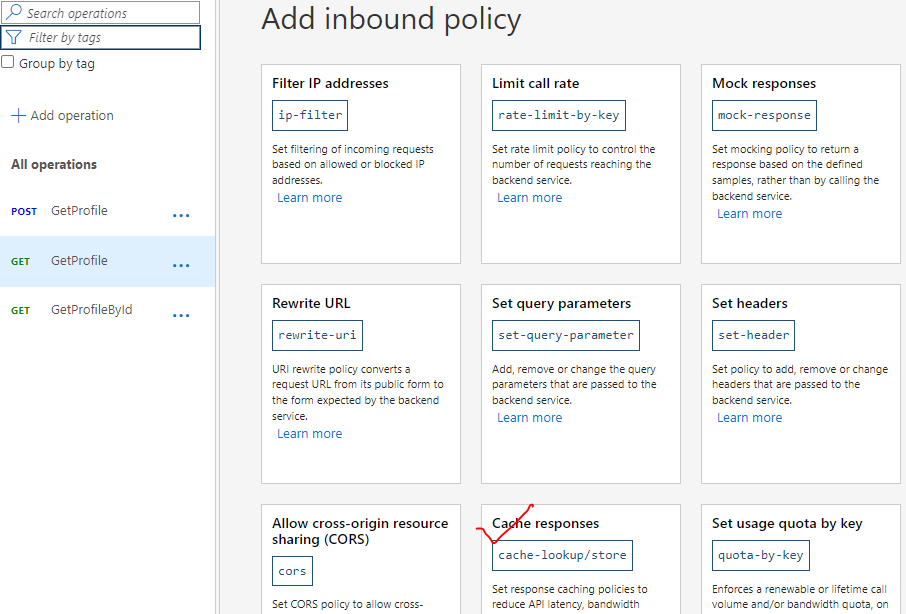
### CACHING POLICIES

#### ADDING CACHE POLICY

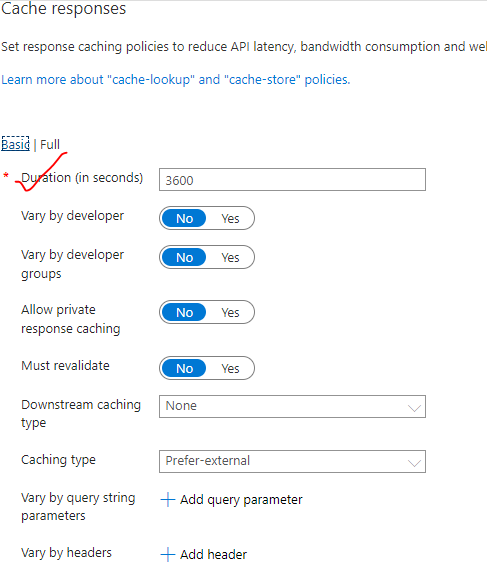
* To apply the policy on the operation level . In the design mode 🡪 Select the operation 🡪 Add policy



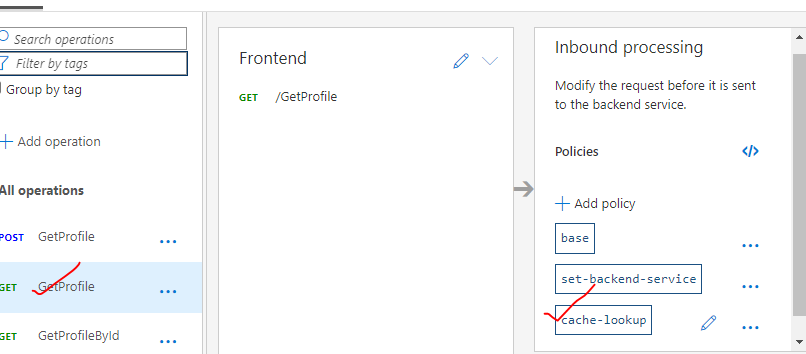
* Select the inboind cache policy



* Provide the cache duration 🡪 Save



Cache policy added to the API operation



#### VALIDATION

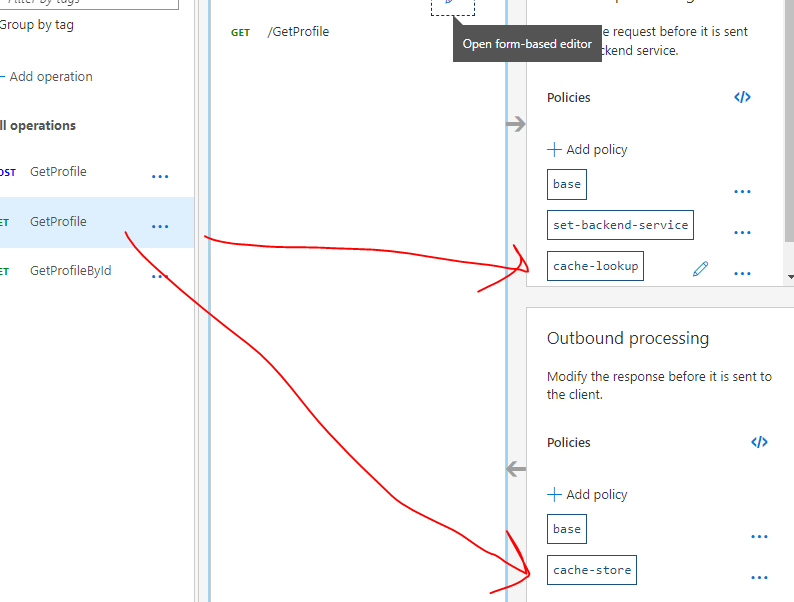
* To validate the cache. Go to test and send the request. After the first request the subsequent responses will come from cache.
* Trace the API to view the responses



#### POLICY FILE

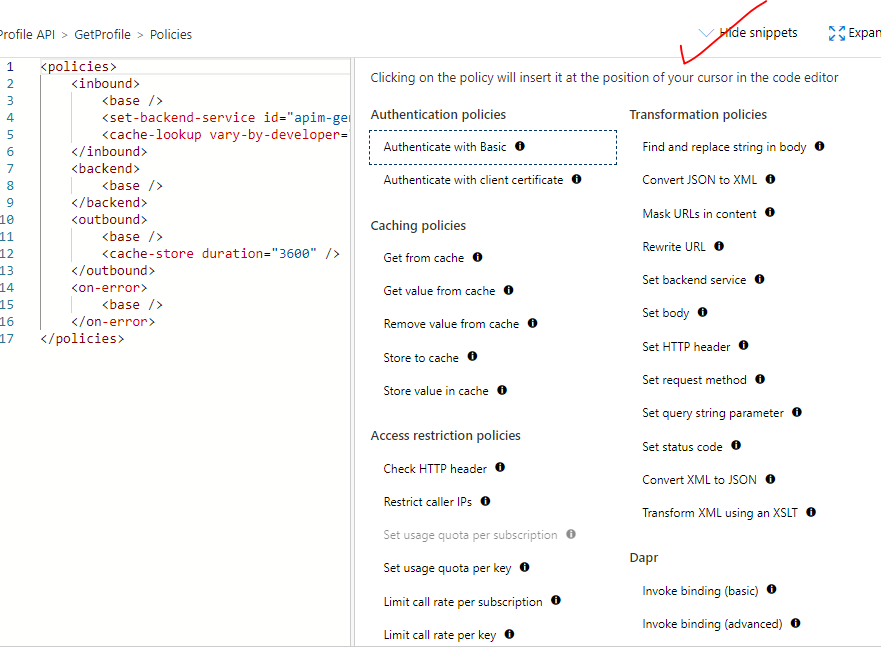
|  |
| --- |
| <policies>      <inbound>          <base />          <set-backend-service id="apim-generated-policy" backend-id="profileapp" />          <cache-lookup vary-by-developer="false" vary-by-developer-groups="false" allow-private-response-caching="false" must-revalidate="false" downstream-caching-type="none" />      </inbound>      <backend>          <base />      </backend>      <outbound>          <base />          <cache-store duration="360" />      </outbound>      <on-error>          <base />      </on-error>  </policies> |

* Adding a cache-lookup in the inbound policy adds a “cache store” in the outbound policy. Both inbound and outbound policy go hand in hand for caching to work



Note –

1. We can add other policies as shown below
2. Documentation : <https://learn.microsoft.com/en-us/azure/api-management/authentication-certificate-policy>



#### ADDING AND READING FROM CACHE

### CROSS DOMAIN POLICIES

### TRANSFORMATION POLICIES

### ADVANCED POLICIES

### DEBUGGING POLICIES