**APIC**

* API Connect is one of the integration tools of Datapower.
* To develop API’s, we use IBM API Connect.
* The runtime of APIC is Datapower.
* APIC is based on specific user feedback, the features will be available only to that user.
* When you provide services with extra features rather than what is required, then it is of no use.
* APIC is customer level, means we can change our service according to the need of cost. (Eg: Hotstar, Aha, Netflix, prime etc.)
* Socialization means the API is visible to public.
* Monetization is making money with the use of API. (Eg: Recharge using different plans).

**APIC Components:**

1. **Cloud Manager:**

* It contains LDAP, SSL (or) TLS, Gateway, servers, provider org., members.
* Cloud Manager is only accessible for Admins / Architects.

1. **API Manager:**

* It consists of API’s and products, Members, Catalogs and spaces, Drafts, Dashboard, Analytics, Catalog settings, Admin config and roles.
* The configurations that are done in provider org. will be only visible in APIM.
* Drafts consists of Products and API’s, and Products consists of API’s and plans.

1. **Developer Portal:**

* It is a consumer organization.
* It is a marketplace for apic.
* It is based on Drupal (open-source framework).
* API’s are available in developer portal when API providers publish it in API manager.
* Each catalog has its own developer portal.
* It also provides forums, blogs, Support, and can review API’s too.
* In API Products, we can subscribe to API based on plan.
* Developer portal is always mapped to catalog.

1. **Gateway Service:**

* Whenever we want to call any API service, we need to call the endpoint.

If API is deployed, the end point is used.

Graphical user interface, text, application, website

Description automatically generated

1. **Developer Toolkit:**

* It provides a command line tool for creating and testing API.
* It is similar to command prompt.

1. **Analytics:**

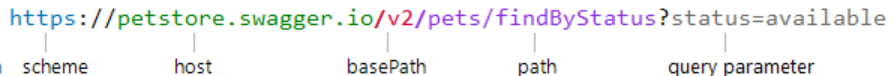
* KIBANA visualization tool is used.
* The data is related to transactions.

**SWAGGER:**

Swagger is Open API. It is a universal format to define API structure with .yml and .Raml extension.

**SECTIONS:**

* **Info:** It contains basic information like title of api, name, version, description.
* **Schemes:** It defines which transfer protocol you want API to use. (Http, https, ws, wss).
* **Host:** What ever the gateway is in catalog will get reflected here. Default is $(catalog.host)
* **Basepath:** It is the url prefix for all API paths. It should start with (/) following with path name.



* **Consumes:** The type of data that API takes as input. (request)
* **Produces:** The type of data that API gives as output. (response)
* **Lifecycle:** It contains 3 phases:
* **Identified:** API is in initial phase and is not designed fully nor implemented.
* **Realized:** It is a default state. It is in implementation phase and is ready for development.
* **Specified:** API is fully designed but not yet implemented.
* **Security Definition:** It shows which type of security we want to use. It is of 3 types
* API Key
* Basic
* OAuth
* **Security:** It just shows the selection of how many types of security implementation.
* **Properties:** It is used to define values.
* **Paths:** It is used to add parameters by assigning name, and data type. Also used to add different types of operations (GET, PUT, PATCH, POST, DELETE).

**Phases of API:**

* Create (Developing and writing api definition)
* Run (deploying api)
* Manage
* Secure

**Product Life Cycle:**

It consists of 5 stages:

* **Stage:** It is the initial stage. When a product is in staged state, it is not visible and cannot be subscribed by the customers.
* **Publish:** When a product is published, the product is visible to the customers and are ready to subscribe it.
* **Retire:** When you retire a product, the product version cannot be viewed and plans are not subscribed.
* **Deprecate:** When you deprecate a product, the users who are subscribed already can use it, but no new users can subscribe to product.
* **Supersede:** Upon developing the new version of existing API, the old version will get deprecated and the new version will get superseded and will be available for the customers to subscribe.

**SECURITY IMPLEMENTATIONS:**

3 types of implementations on API

* **Client/API keys:** We will be having a client ID and a client secret.
* **Basic credentials:** We will be having a username and a password.
* **OAUTH (2.0):** It uses tokens to prove identity between client and server. It allows third party websites or applications to access the data without requiring the user to share personal info.

**ERIE Architecture:**

LOB’s under ERIE

Provider Org.

* Claims
* CSP
* Enterprise
* Commercial
* Personal & Life
* Billing
* Experience
* Data
* Business Enablement.

**Policies:**

We use 2 languages in API Connect: XSLT, Gateway Script.

* In Datapower we call Actions, In APIC we call Policies.
* We can drag & drop the actions in the assemble tab.

Policies are categorized into 4 ways:

* Logic
* Transforms
* Policies
* Security

1. **LOGIC:** It is divided into 4 policies,

* **If:** If the condition is true it will enter loop.
* **Operation Switch**: We can choose operation that is already defined in design path. We cannot edit it manually and there will be no otherwise case.
* **Switch:** Used to define conditions or operations. We can edit condition and we have otherwise case in switch.
* **Throw:** It is similar to try and catch.

1. **TRANSFORMS:** It is divided into 5 policies,

* **JSON-XML:** It just converts JSON to XML.
* **Map:** Used to map structures which are already defined in definitions.
* **Redaction:** If we want to remove the policy completely or remove a field we use redaction. It is nothing but hiding (or) masking password using \*.
* **XML-JSON:** It just converts XML to JSON.
* **XSLT**

1. **POLICIES:** It is divided into 6 policies,

* **Activity log**
* **Gateway script**
* **Invoke:** If I want to send a request to server, I need to have end point of that server. That end point will be given in invoke.
* **Proxy:** It is used to expose the dummy service which acts as the original.
* **Set variable**
* **Validate:** It checks whether the schema is properly or not.

1. **SECURITY:** divided into 4 policies,

* Generate JWT
* Generate LTPA token
* Validate JWT
* Validate Username token