



Generalitat de Catalunya
**Centre de Telecomunicacions
i Tecnologies de la Informació**

Guia de desenvolupament i ús plantilla d' API

Febrer 2024

Índex de Continguts

1	<i>Introducció</i>	3
2	<i>Abast o enfocament del document</i>	3
3	<i>Definició i disseny de la plantilla de desenvolupament</i>	4
3.1	Elements de la plantilla	5
3.1.1	Informació	5
3.1.2	Via d'accés Base (Base Path)/ Servidors	6
3.1.3	Properties.....	7
3.1.4	Paths.....	11
3.1.5	Definicions/Components	14
3.1.6	Assembly	15
4	<i>Guia d' ús de la plantilla</i>	15
4.1	Passos previs a la importació de la plantilla al toolkit	16
4.2	Importació de la plantilla dins del toolkit	17
4.3	Aplicació i ús de la plantilla de desenvolupament base	21
4.4	Aplicació i ús de la plantilla estesa	50
5	<i>Consideracions a tenir en compte per a l' ús de la plantilla</i>	60
6	<i>Annexos</i>	62
6.1	Fitxer de plantilla	62
6.2	Errors base definits en la política de gestió d' errors.....	62
6.3	Plantilla de producte.....	64

1 Introducció

En aquesta guia, es presenta el disseny tècnic de la plantilla de desenvolupament d' APIs, que ha estat ideat com un mecanisme per estandarditzar, integrar i generar un entorn de treball cohesiu dins de l' API Manager de CTTI. L' objectiu principal és proporcionar als desenvolupadors una plantilla que els permeti crear APIs de manera eficient i consistent, seguint les millors pràctiques i directrius establertes.

Aquesta plantilla servirà com un punt de partida per al desenvolupament d' APIs, facilitant la creació d' APIs d' alta qualitat i promovent la reutilització de components i la interoperabilitat entre diferents projectes.

2 Abast o enfocament del document

L' objectiu principal d' aquest document és proporcionar una guia completa del desenvolupament i ús de la plantilla d' APIs definida, amb un enfocament específic en la millora i optimització del treball dels desenvolupadors dins de l' API Manager de CTTI. Aquesta plantilla ha estat dissenyada per oferir un conjunt de directrius i millors pràctiques que permetin als desenvolupadors aprofitar al màxim les capacitats de l' API Manager i garantir un desenvolupament eficient i segur de les APIs.

En aquest document, s'abordan aspectes clau com la definició detallada dels elements de l'estructura de la plantilla, les configuracions prèvies necessàries, així com una guia d'instal·lació i ús d'aquesta, amb exemples il·lustratius per a diferents casos d'ús. D' aquesta manera, es defineix un *Framework de desenvolupament* que serveixi com a base sòlida per a tots els desenvolupadors d' APIs sobre la plataforma de l' API Manager de CTTI.

Aplicació del Document al 100%:

1. Nous Desenvolupaments REST:

Quan s' estiguin realitzant nous desenvolupaments REST dins de l' API Manager de CTTI, el document s' aplicarà en la seva totalitat per garantir la coherència i eficiència des del principi.

Casos que Requereixen Adaptacions:

1. Canvis en requisits de les polítiques personalitzades, extensions o modificació de la plantilla:

Si hi ha canvis substancials en els requisits de les polítiques i extensions de què es fa ús a la plantilla, el document podria requerir ajustaments per garantir l' alineació amb les noves necessitats.

2. Actualitzacions Tecnològiques:

En situacions en què hi hagi actualitzacions tecnològiques significatives, pot ser necessari revisar i adaptar el document per aprofitar les noves capacitats o abordar canvis en l'entorn tecnològic.

3 Definició i disseny de la plantilla de desenvolupament

L'enfocament de disseny d'aquesta plantilla és brindar als desenvolupadors d' APIs a la plataforma de l' API Manager de CTTI un conjunt de directrius clares i concises. Aquesta plantilla ofereix flexibilitat perquè cada desenvolupador pugui adaptar-se a les necessitats específiques del seu projecte, alhora que garanteix el compliment dels estàndards de seguretat requerits per CTTI. En seguir aquestes directrius, els desenvolupadors podran crear APIs d' alta qualitat i coherents, promovent la interoperabilitat i facilitant el manteniment i l'evolució de les APIs al llarg del temps.

La plantilla base que es disposarà per a les **APIs REST** servirà de temperat sobre la qual construir les APIs.

En aquesta plantilla, s'incorporaran les polítiques principals que han de contenir la majoria de les APIs, juntament amb les propietats necessàries per al seu correcte funcionament, així com el de les extensions disponibles a l' API Manager.

Es desenvoluparan dues versions de la plantilla com a temperat **només per a les APIs REST**, una a OpenAPI 3.0 i una altra en 2.0. El motiu de tenir dues versions de la mateixa plantilla és donar suport i ajuda a aquells projectes que encara desenvolupen a OpenAPI 2.0, bé perquè els seus projectes es vegin afectats per les limitacions d'API Connect Instància Reservada (v10.0.5.x) amb la versió 3.0, o bé perquè no hagin pogut migrar a la 3.0 encara. Un exemple de les limitacions actuals és la impossibilitat per convertir un servei SOAP definit per WSDL en una API 3.0, que sí que permet la 2.0.

En el següent enllaç, s'ofereixen més detalls del que suporta i el que no OpenAPI 3.0 per a una Instància Reservada:

https://www.ibm.com/docs/en/api-connect/10_reserved_instance?topic=definition-openapi-30-support-in-api-connect

post:

	OpenAPI 2.0	OpenAPI 3.0
Via d' Accés	basePath: /codi/basepath_api	servers: - url: /codi/basepath_api
Esquemes	definitions: outputError: type: object	components: schemas: outputError:

REF: 2023-00001

		type: object
Referència al body	<pre> post: responses: '200': description: success schema: \$ref: '#/definitions/outputData' consumes: [] produces: [] parameters: - in: body name: body schema: \$ref: '#/definitions/inputData' </pre>	<pre> post: requestBody: content: '*/*': schema: \$ref: '#/components/schemas/inputData' required: false responses: '200': description: success content: '*/*': schema: \$ref: '#/components/schemas/outputData' </pre>

Les plantilles es troben adjuntes per al seu ús en l' apartat d '[annexos en el punt 6.1](#):

- Versió OpenAPI 3.0: plantillaAPI.yaml
- Versió OpenAPI 2.0: plantilla-api_2.0.yaml

En els següents apartats es detallarà la definició de cadascun dels elements principals de la plantilla de desenvolupament però, si es vol procedir directament amb l' ús de la plantilla, pot continuar pel punt 4.

3.1 Elements de la plantilla

A continuació, es detallen els elements clau que es definiran en la plantilla de desenvolupament d' APIs, sobre els quals treballarà i modificarà el desenvolupador.

3.1.1 Informació

Aquest apartat fa esment a la informació essencial sobre l'API, la qual es trobarà dins de l'etiqueta **"info"** de la plantilla de desenvolupament, i inclourà elements com **títol**, **descripció**, **versió**, **x-ibm-name** i **contacte**. Omplir bé aquesta secció aportarà major claredat, comprensió i seguiment dels canvis i actualitzacions realitzats al llarg del temps.

La informació que haurà de contenir cadascun dels camps dins de l'apartat de **"informació"** de la plantilla seria:

- Etiqueta: info
- Contingut:

- **title:** Nom de l' API. Ha de ser prou descriptiu. Ha de començar pel codi d' aplicació, cada paraula començar en majúscula i ha de contenir espais entre cada paraula en cas d' existir. Ex: **3292 Api Example**.
- **description:** Descripció de la funcionalitat que ofereix l' API amb el suficient detall perquè s' adquireixi el coneixement del que ofereix l' API.
- **version:** Valor numèric (independent de la versió de l'aplicació i el producte) que indicarà la iteració o número de versió de l'API i s'utilitza per fer seguiment de l'arxiu de configuració.
- **x-ibm-name:** Nom que s' utilitzarà per proporcionar un nom específic i únic per a l' API dins l' entorn de l' API Manager de CTTI. Serà igual que el Títol amb totes les lletres en minúscules i amb guions en lloc d' espais. Ex: **3292-api-example**.
- **contact:** S'haurà de posar la persona responsable de l'API i el seu email ja que, si una altra persona ha de fer un canvi, aquesta pugui informar el responsable dels canvis que es realitzaran i veure el possible impacte.

```
openapi: 3.0.0
info:
  title: Plantilla API
  description: |
    **Funcionalidad del API**
  contact:
    name: Arquitectura Integración
    email: 3292@gencat.cat
  version: 1.0.0
  x-ibm-name: plantilla-api
```

Nota: La implementació d' aquesta etiqueta no difereix si es desenvolupa a OpenAPI 3.0 o 2.0.

3.1.2 Via d'accés Base (Base Path)/ Servidors

L' apartat Base Path/Servidors de la plantilla de desenvolupament d' APIs és on es definirà la ruta base o URL principal que s' utilitzarà per accedir a l' API. Aquesta secció de la plantilla proporciona un camp per especificar aquesta ruta base, que és fonamental per establir l' estructura i la ubicació dels endpoints de l' API.

El Base Path defineix la part inicial de la URL que s' utilitzarà per accedir als diferents recursos i funcionalitats de l' API. Per exemple, si el Base Path s'estableix com a **"/api/v1"**, totes les rutes de l'API es construiran a partir d'aquesta base, com **"/api/v1/users"** o **"/api/v1/products"**.

El **Base Path** haurà de ser unívoc i que no pugui generar cap tipus de solapament amb altres URLs d' APIs publicades per altres lots dins de l' API Manager de CTTI.

Amb l'objectiu d'homogeneïtzar i controlar la definició del *Base Path* dins de l'arquitectura d'integració de CTTI, actualment l'equip del SIC obliga totes les APIs a publicar amb un *basePath* que, almenys, contingui el */[codi]/[basepath_api]*.

Nota: *Es recomana revisar les regles de nomenclatura i bones pràctiques indicades per CTTI al web de Canigó.*

Un exemple de *basepath* a OpenAPI 3.0 seria el següent:

- Etiqueta: **servers**
- Contingut:
 - *url:* *Basepath de l' API. S'ha de formar de la manera següent:*
/codi/basepath_api/.
Exemple de basepath: /3292/exemple-basepath/

```
servers:
  - url: /3292/ejemplo-basepath
```

D'altra banda, si per necessitats del projecte s'ha d'utilitzar OpenAPI 2.0, un exemple seria el següent:

- Etiqueta: **basePath**
- Contingut: *Basepath de l' API. S'ha de formar de la manera següent:*
/codi/basepath_api/.
Exemple de basepath: /3292/exemple-basepath/

```
basePath: /3292/ejemplo-basepath
```

3.1.3 Properties

L'apartat "**properties**" a la plantilla de desenvolupament d'APIs és on es detallen les propietats necessàries per configurar i personalitzar el comportament de l'API, abastant aspectes com autenticació, seguretat, transformacions de dades, control d'execució i més. També s'inclouen les propietats requerides per les polítiques associades a l'API, permetent una configuració completa i adaptada a les necessitats específiques del projecte.

És important tenir clar quines extensions i polítiques s'utilitzaran i, amb això, mantenir únicament les propietats d'aquelles que s'utilitzin, alleugerint d'aquesta manera la lògica de l'API.

A l'hora de definir el valor de les propietats de la plantilla, s'han de tenir en compte les consideracions següents:

- S'hauran de generar les polítiques necessàries, tant per al funcionament de l'API com per a les polítiques i extensions que s'utilitzaran.
- A la plantilla apareixeran totes les diferents polítiques i extensions de les quals es fa ús a la plantilla. No inclourà les polítiques necessàries per al funcionament d'altres

REF: 2023-00001

polítiques personalitzades que es podrien fer servir en CTTI, aquestes haurien de ser afegides pel desenvolupador en cas de necessitar usar les polítiques que no apareixen a la plantilla.

- Cada property comptarà amb un valor d'exemple que s'ha de modificar, perquè el desenvolupador tingui una referència.
- Cada property tindrà una descripció que informa per a quines polítiques i extensions és necessària, així com la seva utilització.
- Per a cadascuna d' aquestes properties, es generarà l' estructura de valors per entorn. Si fos necessari la configuració per entorn d'alguna altra, s'haurà d'afegir durant el desenvolupament.

A continuació, s'indiquen les polítiques definides per a la plantilla de desenvolupament, incloent les necessàries per a l'ús de l'API, així com de les polítiques i extensions incloses. A més, s'han afegit com a exemple les polítiques i extensions que es poden fer servir dins de l'API Manager, però que no estan incloses per defecte a la plantilla base. (Marcades en color blau).

Nota: Més endavant en el document es donarà un detall més en profunditat sobre l'ús de cadascuna de les properties de les polítiques i extensions bàsiques incloses en la plantilla, així com de les utilitzades en una expansió dels requisits mínims de la plantilla.

```
properties:
  target-url:
    value: 'https://example.com/test'
    description: >-
      URL del sistema destino. **Property generada por defecto por API
      Connect**
    encoded: false
  erroresParticularesCtti:
    value: >-
      {"CTE0190" : {"httpCode": 400, "httpMessage": "Bad Request",
        "moreInformation": "No se dispone de información sobre el servicio en
        estos momentos."}}
    description: >-
      **Policy: ctti-error-management** Json con los errores particulares de
      este API. **Si no se va a añadir un error particular eliminar esta
      property**
  ips-validas:
    value: '175.0.0.0'
    description: >-
      **Policy: ctti-validate-ip** Listado de IPs, separadas por ",",
      utilizadas para la validación de la política ctti-validate-ip. **Si no se
      utiliza la política de validación de ip eliminar la property**
  get-variables:
    value: 'var1,var2'
    description: >-
      **Policy: ctti-get-variables** Listado de variables, separadas por ",",
      que se desean recuperar con la política ctti-get-variables. **Si no se
```



```

    utiliza la política eliminar la property**
fichero-variables:
  value: 'espacio/nombreFichero.js'
  description: >-
    **Policy: ctti-get-variables** Ruta del fichero sobre el que buscar las
    variables con la política ctti-get-variables. **Si no se utiliza la
    política eliminar la property**
cors-enabled:
  description: >-
    **Extension: ctti-request-cors** Indica si se realiza la extension
    ctti-request-cors. **Si se deja a true se ejecutará la extension de CORS,
    si no se va a ejecutar la extension CORS se puede borrar**
  value: 'false'
cors-origins:
  value: 'https://origen1,https://origen2,https://origen3'
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Origin permitidos para las peticiones a la API. Los
    valores deben ir separados por comas **Si no se va a ejecutar la
    extension CORS se puede borrar**
cors-headers:
  value: 'Origin,X-Requested-With,Content-Type,Accept,Authorization'
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Allow-Headers permitidos para las peticiones a la API.
    Los valores deben ir separados por comas y sin espacios **Si no se va a
    ejecutar la extension CORS se puede borrar**
cors-methods:
  value: 'POST,GET'
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Allow-Methods permitidos para las peticiones a la API.
    Los valores deben ir separados por comas **Si no se va a ejecutar la
    extension CORS se puede borrar**
cors-credentials:
  value: 'false'
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene el valor de la
    cabecera HTTP Allow-Credentials que devolverá la API. Los posibles
    valores son true o false **Si no se va a ejecutar la extensión CORS se
    puede borrar**
cabeceras-seguridad:
  description: >-
    **Extension: ctti-response-headers-secure** JSON con las cabeceras que
    se debe permitir su transmisión en la respuesta del API. Si se indican
    con valor vacío se transmitirá un valor por defecto contenido en la
    extensión **La propiedad no puede borrarse y debe contener las cabeceras
    que se desean transmitir**
  value:      "{      \t\"Strict-Transport-Security\":      \t\"max-age=86400;
includeSubDomains\",      \t\"X-Content-Type-Options\":      \t\"nosniff\",\t\"X-Frame-
Options\":      \t\"deny\",      \t\"Content-Security-Policy\":      \t\"default-src 'self'\",
\t\"Location\":      \t\"\",      \t\"Access-Control-Allow-Headers\":      \t\"Content-Type,

```

```

Authorization\", \t\"Access-Control-Allow-Origin\": \"https://example.com\" ,
\t\"Access-Control-Allow-Methods\": \"GET, POST\", \t\"Access-Control-Allow-
Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\")\"
cabecerasSeguridad-enabled:
description: >-
**Extension: ctti-response-headers-secure** Indica si se realiza la
extension ctti-response-headers-secure. **Si se deja a true se ejecutará
la extension de validación de cabeceras de seguridad, si no se va a
ejecutar la extension de validación de cabeceras de seguridad se puede
borrar**
value: 'false'
activity-log:
enabled: true
success-content: payload
error-content: payload
catalogs:
public_pre:
properties:
target-url: https://examplepre.com/test
ips-validas: 175.0.0.0,175.0.0.1
fichero-variables: cd3292/nombreficheroPRE.js
cors-enabled: 'false'
cors-origins: https://origen1,https://origen2,https://origen3
cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
cors-credentials: 'false'
cabeceras-seguridad: \"{ \t\"Strict-Transport-Security\": \"max-age=86400;
includeSubDomains\", \t\"X-Content-Type-Options\": \"nosniff\",\t\"X-Frame-
Options\": \"deny\", \t\"Content-Security-Policy\": \"default-src 'self'\",
\t\"Location\": \"\", \t\"Access-Control-Allow-Headers\": \"Content-Type,
Authorization\", \t\"Access-Control-Allow-Origin\": \"https://example.com\" ,
\t\"Access-Control-Allow-Methods\": \"GET, POST\", \t\"Access-Control-Allow-
Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\")\"
cabecerasSeguridad-enabled: 'false'
public:
properties:
target-url: https://examplepro.com/test
ips-validas: 175.0.0.0,175.0.0.1
fichero-variables: cd3292/nombreficheroPRO.js
cors-enabled: 'false'
cors-origins: https://origen1,https://origen2,https://origen3
cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
cors-credentials: 'false'
cabeceras-seguridad: \"{ \t\"Strict-Transport-Security\": \"max-age=86400;
includeSubDomains\", \t\"X-Content-Type-Options\": \"nosniff\",\t\"X-Frame-
Options\": \"deny\", \t\"Content-Security-Policy\": \"default-src 'self'\",
\t\"Location\": \"\", \t\"Access-Control-Allow-Headers\": \"Content-Type,
Authorization\", \t\"Access-Control-Allow-Origin\": \"https://example.com\" ,
\t\"Access-Control-Allow-Methods\": \"GET, POST\", \t\"Access-Control-Allow-
Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\")\"
cabecerasSeguridad-enabled: 'false'
privat_pre:
properties:

```

```

target-url: https://examplepre.com/test
ips-validas: 175.0.0.0,175.0.0.1
fichero-variables: cd3292/nombreficheroPRE.js
cors-enabled: 'false'
cors-origins: https://origen1,https://origen2,https://origen3
cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
cors-credentials: 'false'
cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400;
includeSubDomains\", \t\"X-Content-Type-Options\": \"nosniff\", \t\"X-Frame-
Options\": \"deny\", \t\"Content-Security-Policy\": \"default-src 'self'\",
\t\"Location\": \"\", \t\"Access-Control-Allow-Headers\": \"Content-Type,
Authorization\", \t\"Access-Control-Allow-Origin\": \"https://example.com\" ,
\t\"Access-Control-Allow-Methods\": \"GET, POST\", \t\"Access-Control-Allow-
Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\"}"
cabecerasSeguridad-enabled: 'false'
privat:
  properties:
    target-url: https://examplepro.com/test
    ips-validas: 175.0.0.0,175.0.0.1
    fichero-variables: cd3292/nombreficheroPRO.js
    cors-enabled: 'false'
    cors-origins: https://origen1,https://origen2,https://origen3
    cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
    cors-credentials: 'false'
    cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400;
includeSubDomains\", \t\"X-Content-Type-Options\": \"nosniff\", \t\"X-Frame-
Options\": \"deny\", \t\"Content-Security-Policy\": \"default-src 'self'\",
\t\"Location\": \"\", \t\"Access-Control-Allow-Headers\": \"Content-Type,
Authorization\", \t\"Access-Control-Allow-Origin\": \"https://example.com\" ,
\t\"Access-Control-Allow-Methods\": \"GET, POST\", \t\"Access-Control-Allow-
Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\"}"
    cabecerasSeguridad-enabled: 'false'

```

3.1.4 Paths

L'apartat de l'etiqueta "*paths*" a la plantilla de desenvolupament d'APIs és on s'especificaran les rutes i els endpoints disponibles a l'API. Aquesta secció és fonamental per definir i documentar les diferents operacions i recursos que es poden accedir a través de l' API.

En l'apartat de l'etiqueta "*paths*", s'enumeraran les rutes i es detallaran les operacions HTTP associades a cadascuna d'elles, com GET, POST, PUT o DELETE. A més, es proporciona informació addicional sobre els paràmetres requerits, les respostes esperades i qualsevol altra configuració específica per a cada endpoint.

Els paths han de ser definits de la manera habitual dins de l'etiqueta "*paths*", havent-se d'indicar correctament cada codi de resposta HTTP amb el seu corresponent esquema de resposta.

Un exemple de definició de Paths, amb els seus corresponents subcamps, per a les APIs amb versió OpenAPI 3.0 seria el següent:

- Etiqueta: *paths*
- Contingut:
 - *</path>*: Cadascun dels paths a definir dins l'API. *<verb>*: Cadascun dels verbs d'aquest path a definir a l'API.
 - *responses*: Conté els codis HTTP i les definicions de les respostes. Cada codi HTTP de resposta per a aquesta operació i verb que pugui tenir l' API ha d' estar definit. Cada codi HTTP de resposta ha de tenir la referència a la seva definició de sortida.
 - *requestBody*: Conté la definició del body de la request.
 - *parameters*: Conté la definició dels paràmetres de la request.

```

paths:
  /prueba-piloto:
    get:
      parameters:
        - name: user
          in: query
          schema:
            type: string
      responses:
        '200':
          description: success
          content:
            '*/*':
              schema:
                type: string
        '400':
          description: Bad Request
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/outputError'
        '401':
          description: Unauthorized
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/outputError'
        '500':
          description: Internal Server Error
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/outputError'
    post:
      requestBody:
        content:
          '*/*':
            schema:
              $ref: '#/components/schemas/inputData'
        required: false
      responses:
        '200':
          description: success
          content:
            '*/*':
              schema:
                type: string
        '400':
          description: Bad Request
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/outputError'
        '401':

```

REF: 2023-00001

3.1.5 Definicions/Components

En l'apartat "**Definicions**"/"**Components**" de la plantilla és on s'inclourà la informació sobre els camps i estructures de dades que podem utilitzar a l'API, sent un exemple la definició de l'objecte "**outputError**" amb les seves respectives.

La definició dels camps es realitzarà utilitzant la convenció de nomenclatura **lowerCamelCase**, la qual cosa implica que el nom del camp comença amb una lletra minúscula i les paraules subsegüents comencen amb una lletra majúscula. De la mateixa manera, s'ha d'entendre el nom del camp i és convenient afegir una breu descripció i exemple, perquè qualsevol persona que vegi l' API al portal per subscriure's pugui entendre les definicions.

En aquest punt és on s'afegiran les validacions que han de realitzar les polítiques de validació.

Un exemple de Definicions per a les APIs amb versió OpenAPI 3.0 seria el següent:

- Etiqueta: **components**
- Contingut:
 - **schemas**:
 - **<schemaName>**: Cadascun dels esquemes que es necessiti definir per complir totes les casuístiques de l'API.

```
components:
  schemas:
    inputData:
      type: object
      properties:
        nombre:
          type: string
        apellidos:
          type: string
        dni:
          type: string
    outputError:
      type: object
      properties:
        httpCode:
          type: integer
          description:Codigo de estado de respuesta HTTP
        httpMessage:
          type: string
          description: Breve descripcion del codigo de estado HTTP
        moreInformation:
          type: string
          description: Descripcion detallada del error
```

3.1.6 Assembly

L'apartat de l'etiqueta "*assembly*" a la plantilla de desenvolupament d'APIs s'utilitza per definir el procés d'acoblament i execució de l'API, prenent com a referència l'arquitectura base definida. D'aquesta manera, es pot assegurar que l' API compti amb els components bàsics en l' acoblament.

En l' acoblament, es definirà el procés d' execució de l' API i, per això, farem ús de diferents components, entre els quals es troben les extensions i polítiques. A la plantilla, ja estan incloses les propietats necessàries per a cadascuna d' aquestes extensions i polítiques, indicades anteriorment en aquest document amb una breu explicació de cadascuna.

La plantilla de desenvolupament d'APIs contindrà com a base en l'acoblament l'ús de les polítiques *validate-request*, *invoke-log* (abans del component *invoke*), *invoke* i *invoke-log* (després del component *invoke*). També s'inclourà la política de "*Gestió d' errors*" associada a un catch que capturarà els errors del tipus "*BadRequestError*", "*ConnectionError*" i "*JavaScriptError*", així com per a l'error "*default*".

Consideracions a tenir en compte:

- **Validació de camps:** S' afegirà un GatewayScript amb un exemple de crida a la llibreria de validació de l' entrada.
- **Control d'errors (Catch):** Es crearà un catch default i se li inclourà la política de gestió d'errors, configurada per produir una resposta JSON.

4 Guia d' ús de la plantilla

En aquest apartat, es proporcionarà una descripció detallada dels passos necessaris per utilitzar la plantilla de desenvolupament d' APIs a l' API Manager de CTTI. S'abordarà des de la importació de polítiques i extensions fins a la configuració de les polítiques de l'API necessàries. A més, es brindaran exemples i casos d'ús per il·lustrar cada pas i facilitar la seva comprensió i aplicació.

Per facilitar la comprensió dels següents apartats del document, seguidament es descriuen alguns conceptes a tenir en compte a l' hora de desenvolupar APIs dins l' API Connect:

- **Política (*Catalog-scoped user-defined policy*):** Les polítiques customitzades, o USER DEFINED dins de la terminologia d'IBM, són fragments de configuració dissenyats per controlar un aspecte concret del procés en el servei de gateway durant el maneig d'una invocació d'API en temps d'execució, satisfent així els requisits exclusius del nostre projecte i permetent la personalització dels controls d'accés, seguretat i configuració dependent de la funcionalitat de cada política.
- **Extensió (*Global-scoped user-defined policy*):** Són similars a les polítiques customitzades descrites anteriorment però amb la diferència que, una vegada

implantades en els Gateways per a un determinat catàleg, sempre s'executaran per a totes les APIs que s'executin en aquest catàleg, d'aquí el nom de **Global-scoped**. Les polítiques es poden usar o no en funció de les necessitats del projecte però, en el cas de les extensions, generalment s'executen sempre, encara que algunes poden desactivar-se segons els valors que s'indiquin en les propietats corresponents, com és el cas de la de validació de CORS.

- **Propietats (de l'API):** Les propietats (propietats) d'una API es refereixen als atributs configurables que defineixen el comportament i les característiques d'una API específica. Aquestes propietats poden incloure informació com l'autenticació requerida, els límits d'ús, les polítiques de seguretat, les transformacions de dades, les configuracions de caixet, entre altres aspectes.
- **Propietats (internes de la política):** Les propietats internes d'una política són configuracions i atributs específics d'una política que s'utilitzen dins d'una API. Aquestes propietats permeten personalitzar i ajustar el comportament de la política. Aquestes poden variar depenent del tipus de política i de les funcionalitats que ofereix.
- **Pre-Request:** es refereix a la fase primerenca de processament d'una sol·licitud API, just abans que s'entri a l'acoblament/lògica de l'API en concret. Durant aquesta etapa, les extensions customitzades "**pre-request**" ens permeten realitzar accions anticipades, com la validació de paràmetres, la modificació d'encapçalats o l'enriquiment de dades, tot això dissenyat per millorar i adaptar la sol·licitud abans que sigui enviada a la destinació. Això és particularment útil per realitzar ajustos anticipats i adaptar la sol·licitud segons les necessitats específiques de l'usuari o del sistema.
- **Post-Response:** es refereix a la fase posterior al processament d'una sol·licitud API, una vegada que la resposta ha estat generada. Durant aquesta etapa, les extensions customitzades "**post-response**" ens atorguen la capacitat d'aplicar lògica personalitzada a la resposta, permetent la modificació, millora o anàlisi de les dades generades abans que es lliurin al sol·licitant.

4.1 Passos previs a la importació de la plantilla al toolkit

Per poder utilitzar la plantilla d'API proporcionada i seguir els passos d'importació que s'indiquen en els apartats posteriors del document, és imprescindible haver importat prèviament les polítiques en l'entorn local o Local Testing Environment ja que, si no és així, quan es vagi a realitzar la importació de la plantilla, es produirà un error, al no trobar-les dins del Toolkit.

Les polítiques imprescindibles que s'han d'importar a l'API Designer, perquè la plantilla de desenvolupament base d'APIs a importar sigui funcional, serien les següents:

- Ctti-invoke-log ([Logs de Invoke](#))
- Ctti-error-management ([Gestió d'errors](#))
- Ctti-validate-request ([Validació de request](#))

Per a això, es disposa d'una guia d'importació de polítiques, on s'indiquen els passos a seguir per a la importació de les polítiques desenvolupades, així com els passos per crear un scriptor cmd que faciliti l'automatització del procés d'arrencada de LTE, juntament amb la importació dels arxius corresponents a aquestes polítiques.



Manual_importacio_p
olitiques_v1.0.pdf

Nota: Cal destacar que, encara que no es vagi a fer ús d'alguna de les polítiques disponibles, és recomanable la importació de totes les polítiques per evitar qualsevol alerta en la importació de la plantilla.

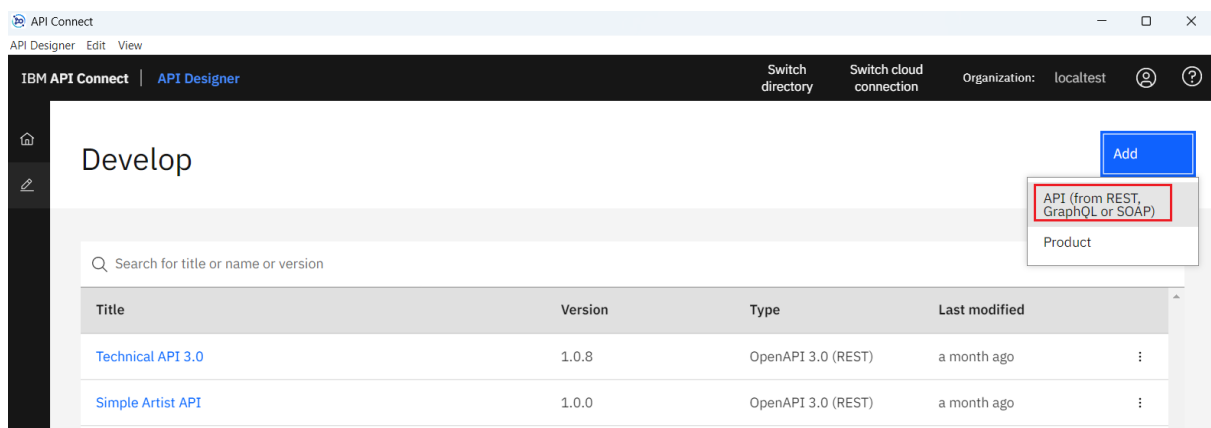
4.2 Importació de la plantilla dins del toolkit

Un cop s'han realitzat els passos previs descrits en el punt anterior, per començar a utilitzar la plantilla, es podria generar una nova definició OpenAPI i enganxar la plantilla sobre ella, o generar-la amb l'opció importar un OpenAPI existent en crear un nou API:

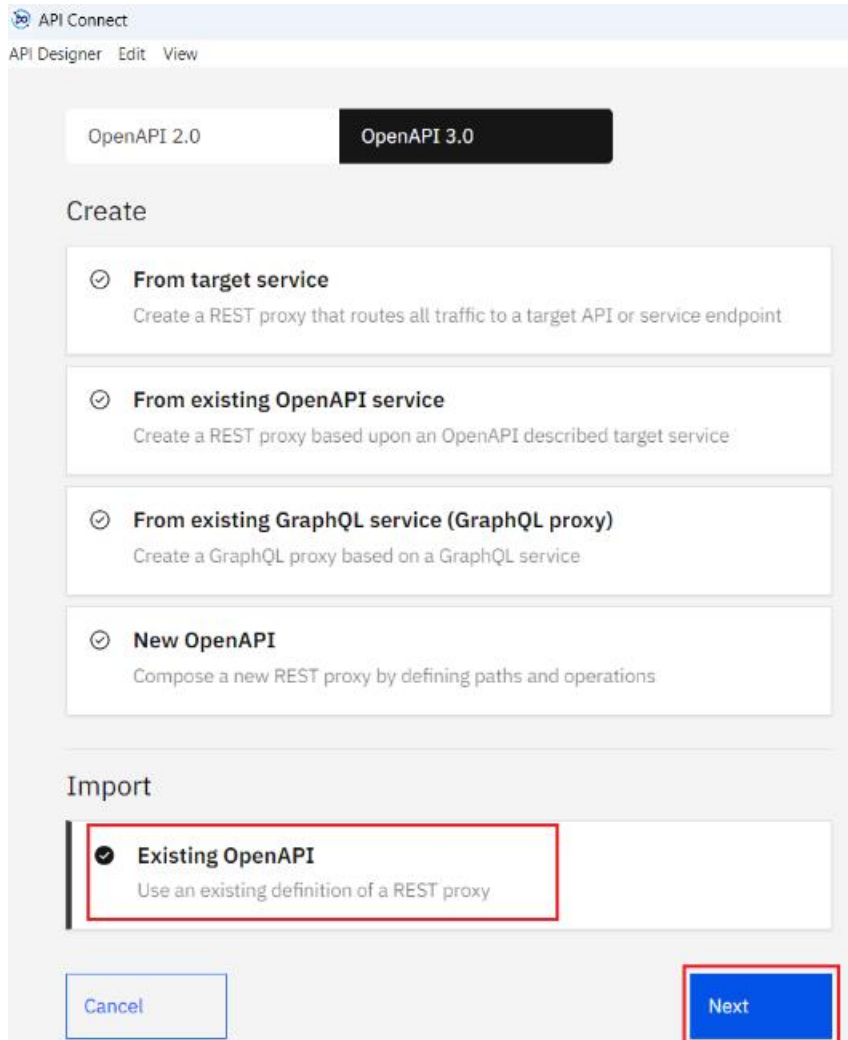
****IMPORTANT:** Si ja s'ha importat l'API de plantilla i no se li han canviat les dades no permetrà importar de nou la plantilla. **

Els passos per importar la plantilla des del toolkit serien els següents:

- Un cop obert el Toolkit, en la vista de Disseny, es pitjaria sobre el botó **Add** i se seleccionaria API.



- En la següent vista, se seleccionaria **OpenAPI 3.0** (o **OpenAPI 2.0** si no es pot usar OpenAPI 3.0 en el projecte concret), es marcaria l'opció d'**Existing OpenAPI** i es polsaria **Next**.



API Connect
API Designer Edit View

OpenAPI 2.0 OpenAPI 3.0

Create

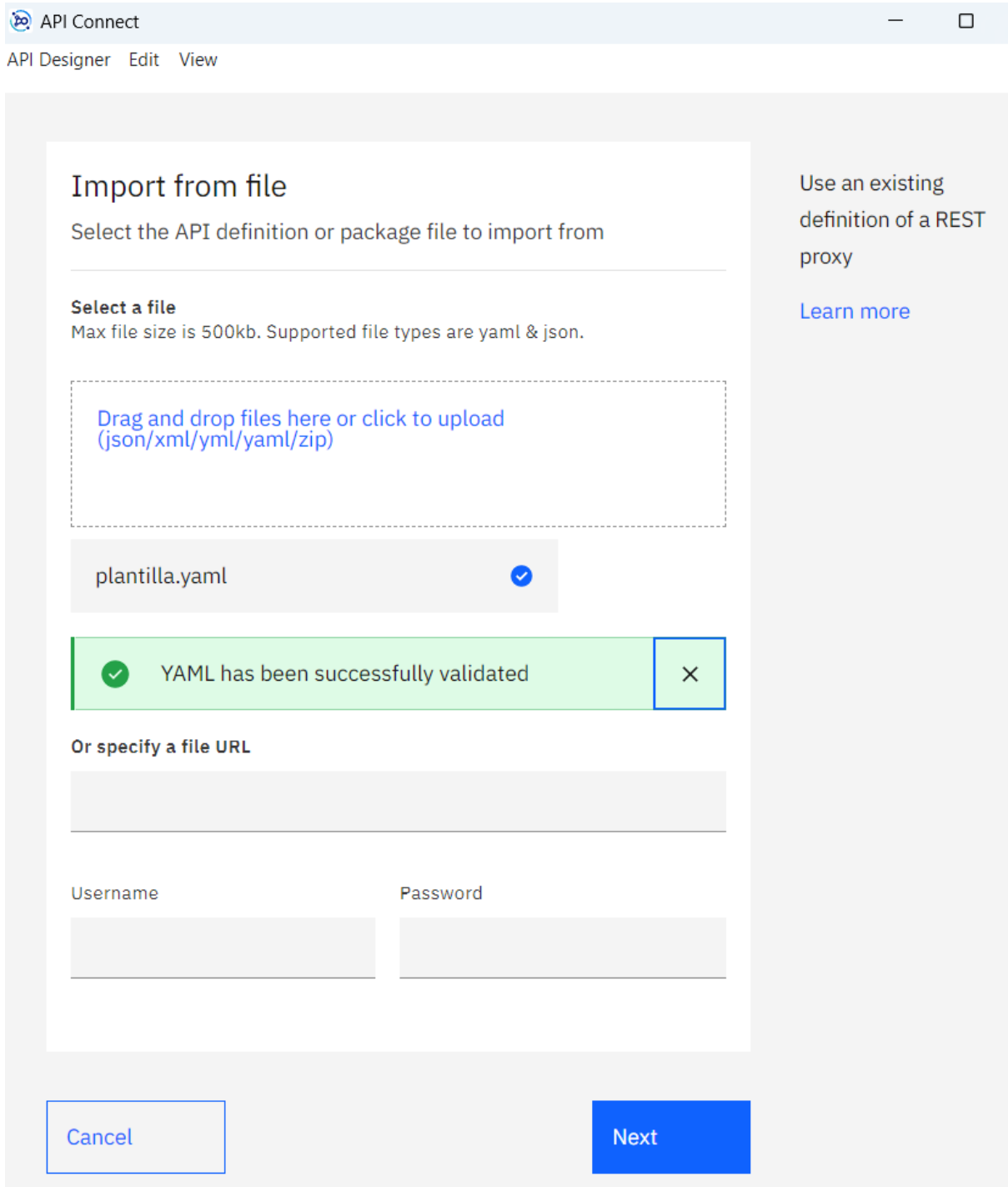
- From target service**
Create a REST proxy that routes all traffic to a target API or service endpoint
- From existing OpenAPI service**
Create a REST proxy based upon an OpenAPI described target service
- From existing GraphQL service (GraphQL proxy)**
Create a GraphQL proxy based on a GraphQL service
- New OpenAPI**
Compose a new REST proxy by defining paths and operations

Import

- Existing OpenAPI**
Use an existing definition of a REST proxy

Cancel Next

- A la següent pantalla, es carregaria el **YAML** de la plantilla que s'hauria modificat en la fase de Disseny i es polsaria **Next**.



API Connect

API Designer Edit View

Import from file

Select the API definition or package file to import from

Select a file
Max file size is 500kb. Supported file types are yaml & json.

Drag and drop files here or click to upload
(json/xml/yml/yaml/zip)

plantilla.yaml ✓

✓ YAML has been successfully validated ✕

Or specify a file URL

Username Password

Cancel Next

Use an existing definition of a REST proxy

[Learn more](#)

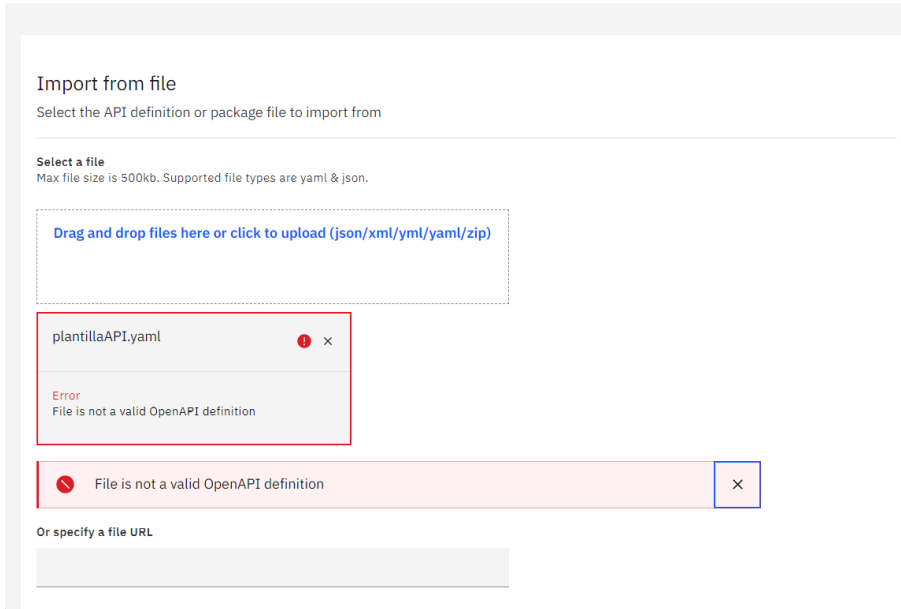
Amb això, el YAML de la plantilla de l'API ja estaria carregat al Toolkit per poder treballar amb ell.

Nota: En el cas que es produeixi un error en importar la plantilla a l' API Designer, com en el següent cas:

REF: 2023-00001

API Designer Edit View
Develop / Select API type /

Import API



Import from file
Select the API definition or package file to import from

Select a file
Max file size is 500kb. Supported file types are yaml & json.

Drag and drop files here or click to upload (json/xml/yml/yaml/zip)

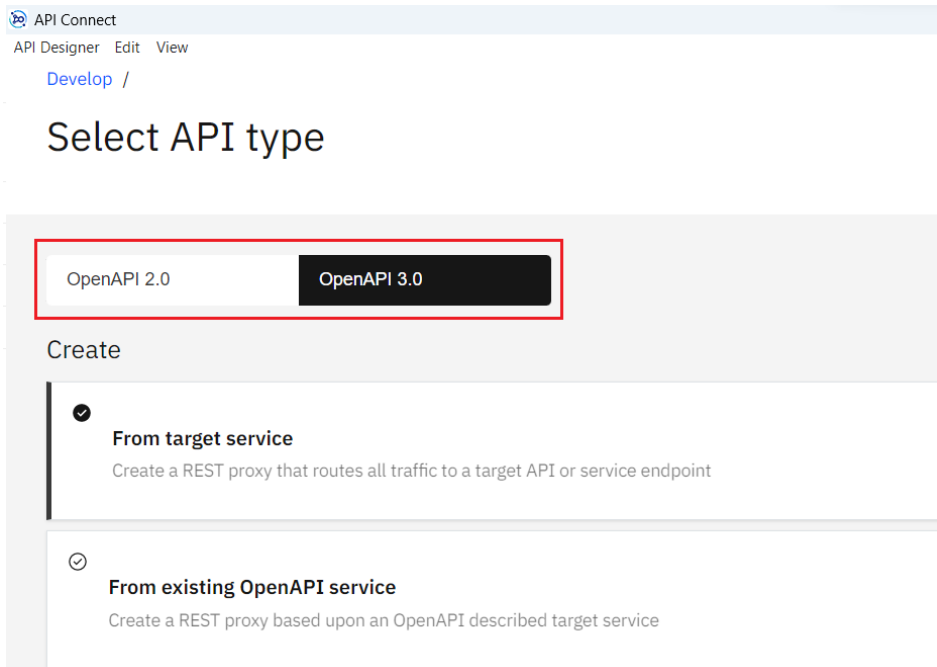
plantillaAPI.yaml ✖

Error
File is not a valid OpenAPI definition

File is not a valid OpenAPI definition ✖

Or specify a file URL

Llavors, s'ha d'assegurar que s'ha escollit correctament la versió de la plantilla API que s'importarà en la interfície del toolkit.



API Connect
API Designer Edit View
Develop /

Select API type

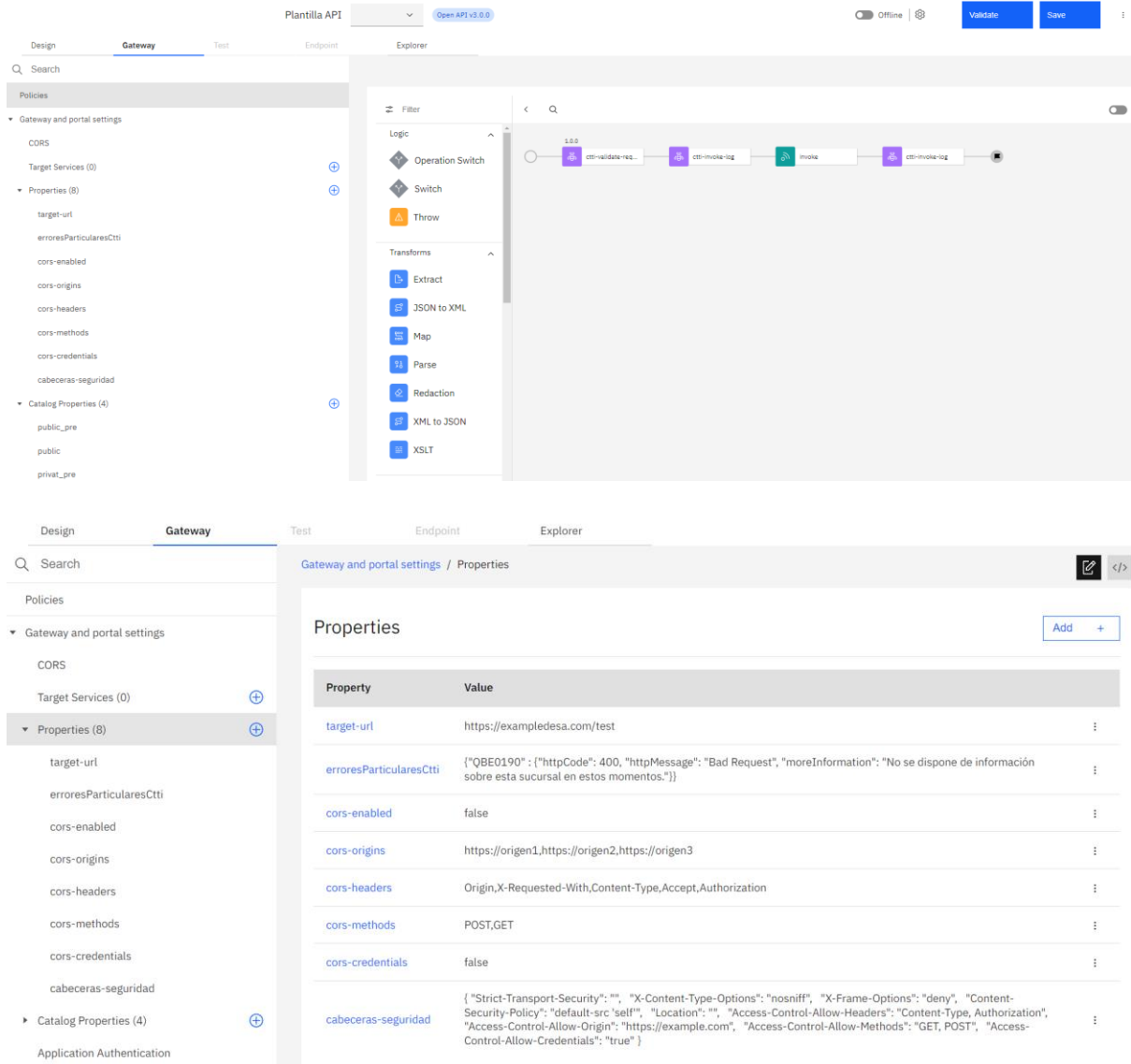
OpenAPI 2.0 OpenAPI 3.0

Create

- From target service
Create a REST proxy that routes all traffic to a target API or service endpoint
- From existing OpenAPI service
Create a REST proxy based upon an OpenAPI described target service

4.3 Aplicació i ús de la plantilla de desenvolupament base

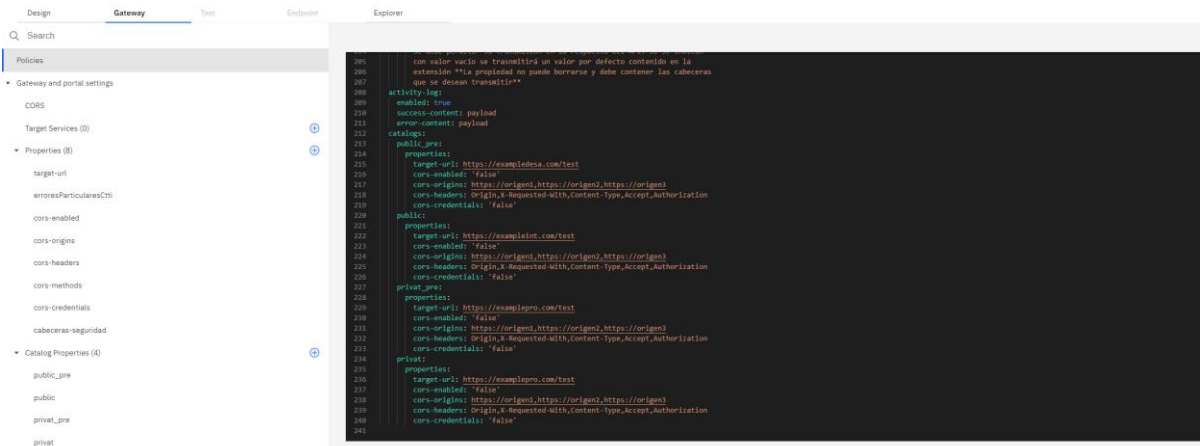
Un cop s' hagi importat el YAML de l' API al Toolkit seguint els passos descrits a l' apartat anterior, ja es tindria una arquitectura d' API base per poder treballar, assegurant d' aquesta manera que es comptaria amb els components bàsics a l' acoblat, requerits per CTTI.



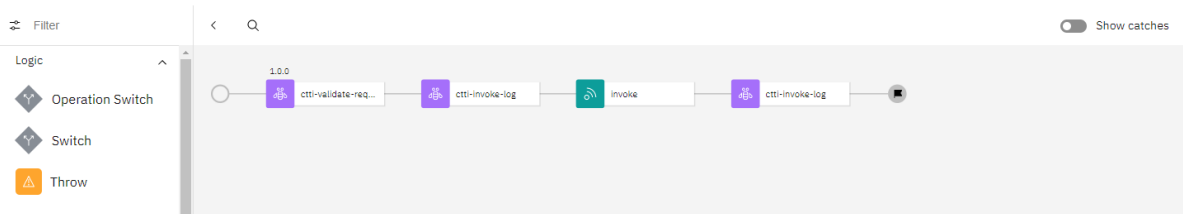
The screenshot displays the API Gateway configuration interface. The top navigation bar includes 'Plantilla API', 'Open API v3.0.0', 'Offline', 'Validate', and 'Save'. The main interface is divided into several sections:

- Design View:** Shows a flow diagram with components like 'Operation Switch', 'Switch', 'Throw', 'Extract', 'JSON to XML', 'Map', 'Parse', 'Redaction', 'XML to JSON', and 'XSLT'.
- Properties View:** Displays a table of properties for the 'Gateway and portal settings'.

Property	Value
target-url	https://exampledesa.com/test
erroresParticularesCtti	{ "QBEO190" : { "httpCode": 400, "httpMessage": "Bad Request", "moreInformation": "No se dispone de información sobre esta sucursal en estos momentos." } }
cors-enabled	false
cors-origins	https://origen1,https://origen2,https://origen3
cors-headers	Origin,X-Requested-With,Content-Type,Accept,Authorization
cors-methods	POST,GET
cors-credentials	false
cabeceras-seguridad	{ "Strict-Transport-Security": "", "X-Content-Type-Options": "nosniff", "X-Frame-Options": "deny", "Content-Security-Policy": "default-src self", "Location": "", "Access-Control-Allow-Headers": "Content-Type, Authorization", "Access-Control-Allow-Origin": "https://example.com", "Access-Control-Allow-Methods": "GET, POST", "Access-Control-Allow-Credentials": "true" }



La plantilla de desenvolupament base compta a nivell d'acoblament amb les polítiques "**ctti-validate-request**", "**ctti-invoke-log**" (abans de realitzar la invocació al backend), "**invoke**" i "**ctti-invoke-log**" (després de la invocació al backend). Igualment es disposarà de cadascuna de les propietats bàsiques necessàries per al funcionament de l' API i de les corresponents polítiques i extensions.



A continuació, es procedeix a detallar cadascuna de les **polítiques** que s' inclouen, així com el seu funcionament i ús.

- **ctti-validate-request:**

Aquesta política permetrà al desenvolupador realitzar la validació del missatge d' entrada de la invocació a l' API contra el seu esquema custom definit i generat en les operacions del disseny de l' API, validant que els paràmetres requerits a l' entrada estan correctament informats.

Aquesta política podrà validar:

- *Queryparams* definit en **x-customPaths**
- *Headers* definit en **x-customPaths**
- *Body* definit en **x-customDefinitions**

Nota: Cal destacar que la política només realitzarà la validació sobre el cos (body) de la resposta, si s'ha enviat en format JSON. Altrament, la política no procedirà a realitzar la validació del body.

La política no requerirà cap paràmetre d' entrada, ni l' ús de cap propietat específica. Igualment, el desenvolupador haurà d' incloure totes aquelles validacions que s' hagin de realitzar en la definició del YAML de l' API. El detall de com s' hauran de realitzar aquestes accions es detalla a continuació.

Els elements que consultarà la política per veure les dades a validar estaran contingudes dins l'etiqueta del YAML *x-customPaths* → *path* → *verbo* → *request* que es troba dins de l'etiqueta *x-ibm-configuration* de la definició de l'API.

Un exemple de *x-customPaths* seria el següent:

```
x-customPaths:
  /gestion-consulta:
    post:
      responses:
        '200':
          description: success
          schema:
            type: string
        '201':
          description: 201 Create
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputData'
        '400':
          description: 400 Bad Request
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
      request:
        params:
          param1:
            obligatorio: true
            type: string
            maxLength: 30
        headers:
          header1:
            obligatorio: true
            type: string
            maxLength: 30
        body:
          $ref: '#/x-ibm-configuration/x-customDefinitions/inputData'
```

Les validacions de *queryparams* i *headers* s' hauran d' introduir directament sobre aquesta secció del YAML, mentre que la definició de validacions del body s' haurà d' indicar dins l' estructura *customDefinitions* i indicar la seva referència dins l' **etiqueta body**.

Per incloure dins de *x-customDefinitions* l' esquema de validació entrada que es requereixi, el qual s' ha referenciat en *x-customPaths* anteriorment, i que tindrà tots els camps i les seves propietats de validació a realitzar per la política, cal seguir els següents passos:

- Dins de *x-customDefinitions*, es crearà una nova etiqueta, que ha de tenir el mateix nom que s' ha especificat en la referència dins de *x-customPaths* com a request d' una operació perquè pugui ser aplicable. En el nostre cas d'exemple exposat anteriorment, seria *"inputData"*.
- L'objecte de l'esquema a crear haurà de tenir el camp *"type"*, per indicar el tipus d'objecte que és (object, array, etc..), i el camp *"properties"*, on s'inclouran tots aquells camps que han de ser validats per la política.
- Dins de *"properties"*, s'inclouran tots aquells camps i les seves corresponents propietats de validació que la política ha de realitzar sobre cadascun d'ells.

Per exemple, per a la següent estructura JSON:


```
{
  "descripcion": "CamposEjemplo",
  "flag": false,
  "estado": "OK",
  "arrayDeObjetos":[
    {
      "descripcion": "Objeto1",
      "flag": true
    },
    {
      "descripcion": "Objeto2",
      "flag": false
    }
  ],
  "arrayPersona":[
    {
      "arrayObject1":[
        {
          "documentName": "Paco",
          "documentApellido": "Sanchez"
        },
        {
          "documentName": "Antonio",
          "documentApellido": "Perez"
        }
      ]
    },
    {
      "arrayObject2":[
        {
          "documentName": "Anselmo",
          "documentApellido": "Sanchez"
        },
        {
          "documentName": "Enrique",
          "documentApellido": "Sanchez"
        }
      ]
    }
  ]
}
```

Podria ser validada amb la següent definició de YAML:

```
x-customDefinitions:
  inputData:
    type: object
    properties:
      descripcion:
        obligatorio: false
        type: string
        maxLength: 3200
      flag:
        obligatorio: true
        type: boolean
      estado:
        obligatorio: false
        type: string
        list: ['OK', 'KO']
      arrayDeObjetos:
        $ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'
      arrayPersona:
        obligatorio: true
        type: object
        arrayObject:
          documentName:
            obligatorio: true
            type: string
            minLength: 1
          documentApellido:
            obligatorio: true
            type: string
            minLength: 1
          arraySimple:
            obligatorio: true
            array:
              obligatorio: false
              type: number
      arrayDeObjetos:
        type: array
        properties:
          descripcion:
            obligatorio: false
            type: string
            maxLength: 3200
          flag:
            obligatorio: true
            type: boolean
```

Nota: Com es pot veure en l'exemple, es podran definir objectes dins d'objectes a través d'una referència a la definició d'aquest amb el comando "\$ref", com passa per a l'objecte "arrayDeObjetos" (\$ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'), el qual es troba dins de la mateixa estructura, o incloent la definició de l'objecte pròpiament dit, com és el cas de l'objecte "arrayPersona".

Per indicar si l'esquema admet propietats addicionals, s'haurà d'incloure l'etiqueta "*additionalProperties*" amb el seu valor booleà corresponent dins de la definició referenciada.

Un exemple seria el següent:

```
x-customDefinitions:
  inputData:
    properties:
      descripcion:
        obligatorio: false
        type: string
        maxLength: 3200
      flag:
        obligatorio: true
        type: boolean
      estado:
        obligatorio: false
        type: string
        list: ['OK', 'KO']
      arrayDeObjetos:
        $ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'
        additionalProperties: true
```

Les propietats de validació que es permetran amb aquesta política per a cadascun dels camps que s'especifiquin en la definició del YAML (headers, params i body), seran les següents:

- **obligatorio**
 - Indicarà si un camp és obligatori o no.
 - Valors possibles: **true** o **false**.
- **type**
 - Tipus del camp indicat.
 - Valors possibles: **string**, **number**, **boolean**, **object** o **array**.
 - En el cas que sigui un arrelament d' objectes o un conjunt d' objectes, s' haurà de definir com a **object**.
 - En el cas que sigui un **array** simple, s'ha de definir com **array**.
- **regexp**
 - Expressió regular que haurà de complir el camp. Al YAML s' haurà d' incloure entre cometes. Exemple: `"/^d{2}/d{2}/d{2,4}$/"`.
- **list**
 - Llista de possibles valors on haurà d' estar el valor que se li passi. Aquesta propietat **SI** exigeix que el valor informat en la petició coincideixi amb el definit en l' esquema, incloent-hi les majúscules i minúscules. Exemple:
 - Llista: `[MuyBien,AlgoDesgastado,RecienFabricado]`
 - Valor vàlid: `MuyBien`
 - Valor NO vàlid: `muybien`

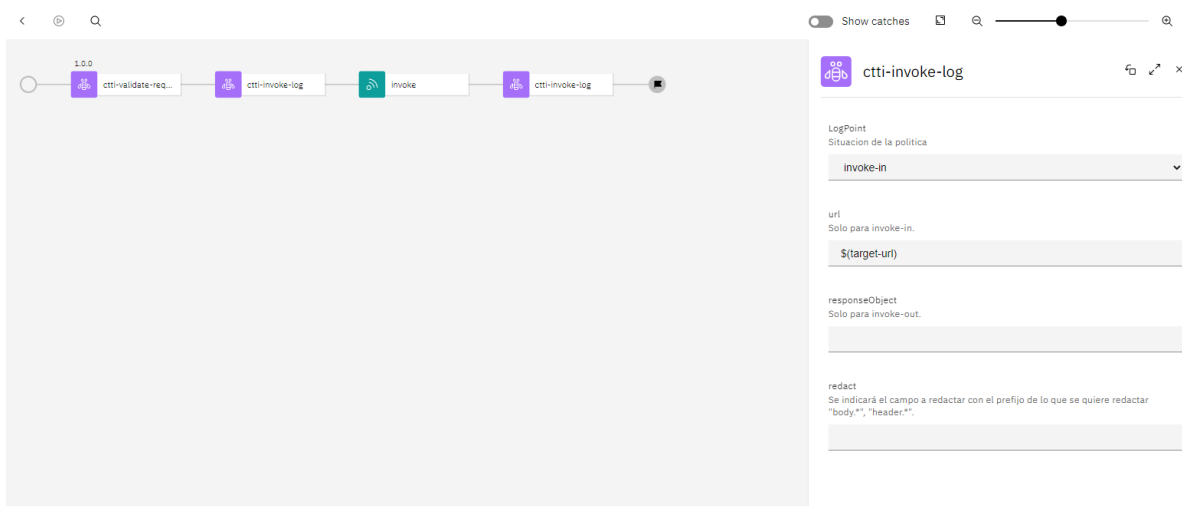
- **upperCaseList**
 - Llista de possibles valors en **majúscules** on haurà d' estar el valor que se li passi. Aquesta propietat **NO** exigeix que coincideixin la capitalització (les minúscules i majúscules) del valor informat en la petició amb l'esquema. Exemple:
 - Llista: *[EXCELENTE,ACEPTABLE,DEPLORABLE]*
 - Valor vàlid: *Excelente*
 - **maxLength**
 - Longitud màxima que permetrà el camp.
 - **minLength**
 - Longitud mínima que permetrà el camp.
 - **length**
 - Longitud exacta que haurà de tenir el camp.
 - **not**
 - Que no sigui el valor especificat.
 - **literal**
 - Valor literal que haurà de portar el camp que es valida.
 - **lengthDecimal**
 - S' amidarà la longitud de la part decimal d' un valor, i que el nombre de xifres decimals no superi el definit.
 - **array**
 - Validarà que un camp és un array amb elements simples, i els seus elements han de tenir el tipus especificat en aquest camp.
 - Valors possibles: **number**, **string** o **boolean**.
 - **arrayObject**
 - Validarà un array d' objectes i en l' esquema cal especificar quines propietats té cada objecte.
- **ctti-invoke-log:**

Amb aquesta política, es podrà guardar en el log la request i response de la política d'Invoke per a posteriorment ser enviats a l'**Analytics**, permetent d'aquesta manera registrar més etapes del flux d'execució, podent el desenvolupador revisar la invocació abans i després de ser enviada al backend, amb la qual cosa permetrà identificar si la lògica aplicada en l'API i la resposta del backend estan funcionant correctament, agilitant la identificació i resolució d' errors, i ampliant la traçabilitat.

Per a això, caldria enviar com a informació la **URL**, **capçaleres** i **body** de la petició per al cas **invoke-in** (abans de la política invoke), i les **capçaleres** i **body** de la resposta per al cas **invoke-out** (després de la política invoke). Per tant, aquesta política s'haurà d'afegir just abans i després d'una política de tipus Invoke.

La política disposarà de les propietats pròpies següents:

- **logPoint:** Propietat que indicarà si vam gravar a Analytics la trucada o la resposta del servei.
 - **invoke-in:** S'haurà d'utilitzar quan se situa la política just abans de l'invoke per enviar les dades de la request al log.
 - **invoke-out:** S'haurà d'utilitzar quan se situa la política just després de l'invoke per enviar les dades de la response al log.
- **url:** (Camp **opcional**) Només s'utilitzarà en el cas de **invoke-in**. Propietat que contindrà la URL que se li passarà a la política Invoke, serveix per a tenir la traçabilitat dels *pathparams* i *queryparams* que es manen en la request.
- **responseObject:** (Camp **opcional**) Només s'utilitzarà en el cas d'**invoke-out**. Nom del camp que contindrà l' objecte de resposta de l' invoke, només haurà de ser utilitzat si es modifica l' objecte de resposta dins la política Invoke. Si s'informés a buit, per defecte l'agafarà del message (cos de la resposta).
- **redact:** (Camp **opcional**) S'utilitzarà per ocultar/emmascarar el contingut dels camps del body o de capçaleres que s'indiquin en aquesta propietat. Per indicar el camp corresponent, s'hauria d'afegir el prefix "**body.**" o "**header.**" seguit del nom del camp que es volgués ocultar. Exemple: **body.data,header.Set-Cookies**



Com es pot veure en la imatge, en pulsar sobre la política "**ctti-invoke-log**", apareixeran a la part dreta les propietats internes de la política que s'han descrit a dalt per poder informar-les amb els valors requerits.

Nota: Els nous camps que s'afegiran al log seran "**invoke-in-{N}**" i "**invoke-out-{N}**", sent N el número de la invocació que correspon al contingut d'aquest log. El valor inicial és l' 1.

- **invoke:**

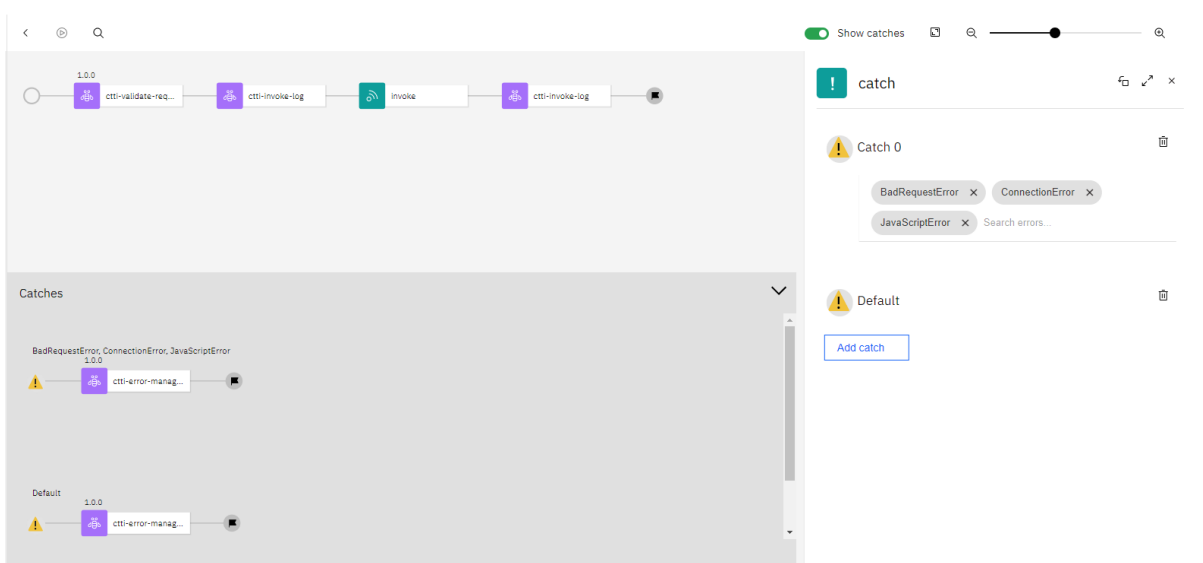
La política **"invoke"** que ofereix IBM API Connect és una política que permet invocar o cridar a un altre servei o API externa des d'una API definida. Aquesta política s'utilitza per fer trucades HTTP a endpoints externs i rebre respostes d'aquests endpoints.

En utilitzar la política **"invoke"**, es pot establir la URL de destinació, els mètodes HTTP (com GET, POST, PUT, DELETE, etc.), els encapçalats de sol·licitud i els paràmetres necessaris per realitzar la trucada al servei extern. A més, es poden configurar opcions addicionals, com l' autenticació, la manipulació de dades i la gestió d' errors, tot això a través de les seves propietats internes.

Com que no és una política custom, sinó que pertany a les que IBM API Connect disposa per usar-se en el seu catàleg de polítiques preconfigurat, no vam aprofundir en les seves propietats. De tota manera, a través del següent enllaç a la pàgina oficial d'IBM, es pot trobar tota la informació relacionada amb aquesta política per a la v10 instància reservada que té CTTI: ([Invoke - IBM Documentation](#)).

D'altra banda, a nivell de gestió d'errors, la plantilla de desenvolupament base establirà per defecte dos **"Catch"**, un per tractar els errors **"ConnectionError"**, **"JavaScriptError"**, **"BadRequestError"**, i un altre per al **"default"**. Tots dos estaran associats a la política de gestió d'errors **"ctti-error-management"**, la qual permetrà personalitzar els errors d'estat i el funcionament dels quals es detallarà més endavant.

Per fer ús dels **"Catches"** dins l'API Manager, s'haurà de fer click sobre **"Show catches"** i després, sobre **"Catches"**, apareixerà un menú al lateral dret. Dins d'aquest menú, s'haurà de fer click a **"Add catch"**, on s'hauria de veure una cosa similar a la imatge:



Igualment, es podria visualitzar i modificar a través de la definició del YAML de la plantilla de desenvolupament:

```
catch:
  - errors:
    - BadRequestError
    - ConnectionError
    - JavaScriptError
    execute:
      - ctti-error-management:
          version: 1.0.0
          title: ctti-error-management
          FormatoRespuesta: json
      - default:
          - ctti-error-management:
              version: 1.0.0
              title: ctti-error-management
              FormatoRespuesta: json
    finally: []
  x-customPaths:
```

A continuació, es detalla el funcionament de la política de gestió d'errors associada als "Catches", que s'inclou en la plantilla de desenvolupament d'APIs bàsica, i que permetrà personalitzar els errors.

- **ctti-error-management (Gestió d'errors):**

Aquesta política proporcionarà una gestió més efectiva i customitzada dels errors produïts dins l'assembly de l'API, en funció de les necessitats del projecte.

Capturarà l'error generat per les polítiques que formen part de l'acoblament de l'API, i generarà i assignarà el body i el codi d'estat HTTP de la resposta de l'error.

Els errors retornats no hauran d'exposar informació sensible dels sistemes, com piles amb la informació dels errors, dades sensibles d'usuaris, etc. Els codis d'error retornats per l'API seran codis d'error estàndard del protocol HTTP.

Amb aquesta política es podran realitzar les següents accions:

- En cas que no es requereixi un format d' error específic, la política gestionarà l' error donat per API Manager i el convertirà a l' estructura d' error **per defecte**.
- Permetrà configurar que l' error es mapeja a una **estructura definida pel desenvolupador**.

D' altra banda, a l' hora de fer ús d' aquesta política, s' hauran de tenir en compte les consideracions següents:

- Per als casos en què el consumidor de l' API requereixi una resposta d' error que la política no pugui realitzar, la gestió de l' error es realitzarà mitjançant un **GatewayScript custom**.
- Si es desitja modificar tant el **"body"** com el **codi de resposta** fora de les possibilitats que ofereix la política, s' haurà de fer en un **GatewayScript** ubicat **després** de la **política d' error**.
- En el cas que es vulgui modificar un **camp** o el **"status code"** d' un error concret, haurà de fer-se després de la política sempre que el API el necessiti.
- Per a qualsevol altre error d' estat que no sigui **"JavaScriptError"**, **"ConnectionError"** o **"BadRequestError"**, la política els interpretarà com a error genèric i retornarà la resposta d' error per defecte amb un codi HTTP 500:

```
{
  "statusCode": 500,
  "httpMessage": "Internal Server Error",
  "moreInformation": "Servicio no disponible momentáneamente"
}
```

Igualment, a banda dels **"Catches"** que vindran definits en la plantilla de desenvolupament, els quals estaran associats a aquesta política de gestió d' errors (**ctti-error-management**), es podran afegir nous o modificar els existents segons les necessitats del projecte, sempre tenint en compte les següents consideracions:

- Cada error podrà manejar-se amb una captura (**Catch**) diferent i cada captura podrà manejar múltiples errors d' estat, com **"ConnectionError"**, **"JavaScriptError"**, **"BadRequestError"** o qualsevol altre que es defineixi.
- Els errors retornats no hauran d' exposar informació sensible dels sistemes, com piles amb la informació dels errors, dades sensibles d' usuaris, etc. Els codis d' error retornats per l' API han de ser codis d' error estàndard del protocol HTTP.
- Per personalitzar els errors d' estat, es podrà fer ús d' aquesta política de gestió d' errors i, al seu torn, es podran utilitzar elements d' acoblament com **"GatewayScript"** o **"set variable"**.

1 - Gestió dels errors de l' API a través de l' estructura d' error per defecte

La política, per defecte, gestionarà l' error donat a l' API Manager, i generarà els errors seguint la següent estructura:

- **statusCode:** Codi HTTP de l' error.
- **httpMessage:** Missatge corresponent al codi HTTP.
- **moreInformation:** Descripció de l' error.

D' aquesta manera, n' hi haurà prou amb usar la política en els catches de l' API perquè els errors passin a ser generats amb aquesta estructura per defecte, facilitant la creació del missatge d' error, generant el body i assignant el valor del codi d' estat de la resposta.

Prenent com a exemple una invocació a un backend, el qual no està disponible, el missatge d' error estàndard que elevarà la política en el body de sortida quedaria de la següent manera:

```

{
  "statusCode": 500,
  "httpCode": "500",
  "httpMessage": "Internal Server Error",
  "moreInformation": "Internal Error"
}

```

2 – Gestió dels errors de l'API configurant una estructura personalitzada d'error

A. Definició de l' estructura custom de l' error

La política permetrà la customització del missatge d'error a mostrar, habilitant la possibilitat que el desenvolupador defineixi un esquema d'error custom en la definició del YAML de l'API, dins de la secció "*x-customDefinitions*".

Per definir un esquema d' error de sortida personalitzat, s' hauran de seguir els següents passos:

1. S' haurà d' indicar la definició de l' esquema d' error de sortida personalitzat dins l' etiqueta "*x-customDefinitions*" de la definició del YAML. Un exemple de definició de l'esquema d'error seria el següent, on es referencia "*\$ref*" a l'esquema "*tppMessagesCustom*" dins l'etiqueta "*outputError*":

```

x-customDefinitions:
  outputError:
    type: object
    properties:
      tppMessages:
        $ref: '#/x-ibm-configuration/x-customDefinitions/tppMessagesCustom'
  tppMessagesCustom:
    type: object
    properties:
      category:
        type: string
        key: valor=ERROR
      code:
        type: string
        key: codigo
      txt:
        type: string
        key: texto
      reason:
        type: string
        key: razon

```

Nota: Per indicar un valor literal, ja sigui string o number, dins d'un camp de l'esquema custom definit, s'haurà d'incloure a l'etiqueta "**key**" del camp el literal "**valor=**" davant del valor que es vol indicar ("**valor=<valor a incloure>**"). Amb això, la política interpretarà que no ha de fer cap mapatge, sinó només deixar el

valor literal indicat. Per a la resta dels camps que apareixen en l'estructura ("code", "txt", "reason"), en l'apartat key de cadascun d'ells, s'ha indicat el nom del camp informat en escalar l'error del qual la política haurà de recollir el seu valor.

2. A continuació, s'haurà de fer referència a l'esquema d'error de sortida personalitzat definit a dalt (**outputError**), en l'apartat de **x-customPaths** de la definició del YAML de l'API, associat al codi d'error en el qual es vulgui aplicar, quedant de la següent forma (en aquest exemple, es referencia com a esquema de sortida a utilitzar en la resposta a l'error 400 per al verb "post"):

```
x-customPaths:
  /gestion-consulta:
    post:
      responses:
        '200':
          description: success
          schema:
            type: string
        '201':
          description: 201 Create
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputData'
        '400':
          description: 400 Bad Request
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
```

3. Un cop s'hagi realitzat la definició de l'esquema d'error de sortida personalitzat i se n'hagi configurat la referència dins la definició del YAML, ja es podria utilitzar en la lògica de la nostra API.

B. Com escalar l'error a través d'un GatewayScript.

Per a poder escalar un error a través d'un GatewayScript, el qual serà gestionat posteriorment per la política de gestió d'errors, el desenvolupador haurà de fer ús de la llibreria context de API Connect, en concret de la funció "**context.reject**".

El format de la funció "**context.reject**" és el següent:

```
context.reject(errorName, errorMessage);
context.message.statusCode = "|Status code| |Status reason|";
```

- En el camp **errorName**, s'haurà d'indicar el valor "**JavaScriptError**" per indicar que és un error generat en una política GatewayScript.
- En el camp **errorMessage**, s'haurà d'incloure un string amb el missatge d'error o una estructura JSON en format string (fer ús de la funció JSON.stringify) amb les dades que es desitgin escalar sobre l'error.

- Per a indicar el valor dels camps “*Status code*” i “*Status reason*”, es realitzarà a través de la sentència “*context.message.statusCode*” després de l'execució de “*context.reject*” on, seguint el format de “*code*” i “*reason*”, s'actualitzaran els camps “*Status code*” i “*Status reason*” del missatge d'excepció. Per exemple, per a indicar un “*Status code*” i “*Status reason*” de “*501 Custom Error*”, s'haurà de posar la següent línia de codi:

```
context.message.statusCode = "501 Custom Error";
```

Un exemple complet d'ús de la funció **context.reject()** per a escalar un error seria el següent:

```
context.reject("JavaScriptError", "No está autorizado para ejecutar este servicio");
context.message.statusCode = "401 Unauthorized";
```

Perquè la política de gestió d'errors pugui manejar els errors escalats/aixecats en els GatewayScripts a través de la funció “*context.reject*”, indicada a dalt, es requerirà el següent:

- Per tenir tota la informació necessària per gestionar l'error, la política de gestió d'errors requerirà que, en el paràmetre “*errorMessage*” de la llibreria “*context*”, s'introdueixi una estructura JSON amb, almenys, els següents camps:

```
{
  "tipoError" : <tipo de error a escalar>,
  "errorCode" : "<El código de error a escalar>",
  "moreInformation": "<Información adicional del error a mostrar>"
}
```

- La política té a disposició una taula d'errors personalitzats ja definits, en els quals es contempen els més comuns, i que podran ser referenciats a l'hora d'aixecar l'excepció ([vegeu el punt de l'annex 6.2](#)).

Un exemple d'escalat d'error, utilitzant la llibreria “*context.reject()*” i l'estructura definida de missatge per al paràmetre “*errorMessage*” indicada a dalt, seria el següent:

Format del codi a utilitzar

```
let excepcion = {
  "tipoError" : "tipoError",
  "errorCode" : "errorCode",
  "moreInformation": "No está autorizado para ejecutar este servicio"
}
context.reject(errorMessage, errorName);
context.message.statusCode = "|Status code| |Status reason|";
```

Exemple del codi

```
let excepcio = {
  "tipoError" : "controlled",
  "errorCode" : "401",
  "moreInformation": "No está autorizado para ejecutar este servicio"
}
context.reject("JavaScriptError", JSON.stringify(excepcio));
context.message.statusCode = "401 Unauthorized";
```

Nota: En el cas que l'error a aixecar sigui un dels definits a la taula d'errors ([vegeu el punt de l'annex 6.2](#)), o s'hagi inclòs a través de la propietat de la política de gestió d'errors **"errorsParticularsCtti"** (vegeu detall de les propietats usades per la política més endavant), s'ha d'indicar en el camp **"tipoError"** el valor **"controlled"** (**"tipoError": "controlled"**). D'aquesta manera, la política anirà a buscar la informació en la taula d'errors que conté a través del **"errorCode"** indicat. En el cas de no venir informat com a **"tipoError"="controlled"**, s'indicarà la informació del missatge per defecte **"500" "Internal Server Error"**, ja que no es realitzaria la recerca a la taula d'errors. Això succeiria igual en el cas d'indicar un **"errorCode"** erroni o que no existeixi a la taula d'errors, tot i haver indicat com a **"tipoError"="controlled"**, ja que aniria a buscar-lo a aquesta taula i no el trobaria.

Amb això, la política comptaria amb la informació necessària per gestionar l'error, retornant el següent missatge d'error per a l'exemple que estem comentant:

```
{
  "status": {
    "code": 401,
    "reason": "Unauthorized"
  },
  "name": "JavaScriptError",
  "message": "{ \"tipoError\": \"controlled\", \"errorCode\": \"401\", \"moreInformation\": \"No está autorizado para ejecutar este servicio\" }",
  "policyTitle": "gatewayscript"
}
```

C. Mapeig de l' error escalat amb GatewayScript a una estructura d' error personalitzada.

Perquè la política pugui realitzar el mapatge de l' error escalat a una estructura personalitzada definida, s' ha de complir que s' hagi configurat una estructura d' error en **"x-customDefinitions"** per a un dels errors definits en el YAML de l' API, i es tracti aquest error a través d' un GatewayScript, com s' ha indicat anteriorment.

La política, en processar l'error escalat, comprovarà si hi ha una estructura d'error personalitzada definida, i anirà muntant el missatge de resposta d'error mapejant cadascun dels camps de l'estructura amb els valors que s'hagin informat en l'escalat de l'error a través de la funció **"context.reject"**.

Prenent la definició de l'esquema d'error següent, per a l'error 400 del verb **"post"**, on es fa referència a l'etiqueta **"outputError"** i, posteriorment, a l'estructura **"tppMessagesCustom"**:

```
x-customPaths:
  /gestion-consulta:
    post:
      responses:
        '200':
          description: success
          schema:
            type: string
        '201':
          description: 201 Create
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputData'
        '400':
          description: 400 Bad Request
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
        '401':
          description: 401 Unauthorized'
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
```

```
x-customDefinitions:
  outputError:
    type: object
    properties:
      tppMessages:
        $ref: '#/x-ibm-configuration/x-customDefinitions/tppMessagesCustom'
  tppMessagesCustom:
    type: object
    properties:
      category:
        type: string
        key: valor=ERROR
      code:
        type: string
        key: codigo
      txt:
        type: string
        key: texto
      reason:
        type: string
        key: razon
```

La lògica del GatewayScript per aixecar l'excepció quedaria de la següent forma:

```
let error= {"tipoError": "controlled", "errorCode": "E0401", "moreInformation": "Servicio no autorizado", "codigo": "13",
"razon": "Permisos insuficientes", "texto": ".....", "campoExterno1": "valorExterno1", "campoExterno2": "valorExterno2"}
context.reject("JavaScriptError",error);
context.message.statusCode = "401 Unauthorized";
```

Nota: Com es pot veure en l'exemple, la variable **"error"** té els tres camps base de l'estructura definida de missatge per al paràmetre **"errorMessage"** (**"tipoError"**, **"errorCode"** i **"moreInformation"**) i la resta dels camps desitjats a mapejar, corresponents als camps definits en l'esquema anterior (**"codigo"**, **"razon"** i **"text"**).

També s'inclouen més camps d'exemple, com **campoExterno1** i **campoExterno2** que, en no estar en l'esquema custom definit "**tppMessagesCustom**" i no pertànyer als requerits per la política, no es mapejaran.

El camp "**tipoError**" té el valor de "**controlled**" per determinar si l'error escalat es troba dins dels definits a la taula d'errors base de la política i, per tant, ha estat un error autoescalat, o no és així i, per tant, s'escalaria l'error per defecte (500).

Els camps que s'incloguin dins la variable **error** que no corresponguin a cap dels camps definits en l' esquema no es mapejarán en l' objecte de sortida com, per exemple: **campoExtern1**, **campoExtern2**.

Finalment, després d'executar-se la lògica de la política amb aquesta informació d'exemple, el resultat del missatge d'error en el body, havent-se realitzat el mapatge sobre l'estructura definida de "**tppMessagesCustom**", quedaria de la següent manera:

```
{
  "tppMessages": {
    "category": "ERROR",
    "code": "13",
    "txt": ".....",
    "reason": "Permisos insuficientes"
  }
}
```

D' aquesta manera, la política permetrà personalitzar el missatge d' error de sortida de l' API, en funció de la informació que es requereixi mostrar dins del projecte en el qual s' estigui aplicant.

3 – La política requerirà fer ús de dues propietats al YAML

- **FormatoRespuesta:** Propietat interna de la política que determinarà el format de sortida del missatge d'error (body), podent ser: (**json/xml/soap**).

La plantilla de desenvolupament de l'API vindrà amb el valor per defecte "**json**" per a aquesta propietat. Aquest es podrà canviar a un altre valor dels permesos a través de la definició del YAML de la plantilla.

```
catch:
  - errors:
    - BadRequestError
    - ConnectionError
    - JavaScriptError
  execute:
    - ctti-error-management:
      version: 1.0.0
      title: ctti-error-management
      FormatoRespuesta: json
    - default:
      - ctti-error-management:
        version: 1.0.0
        title: ctti-error-management
        FormatoRespuesta: json
  finally: []
```

Nota: En el cas que s' indiqui com a format de missatge de sortida soap, la política retornarà com a missatge una estructura SOAP personalitzada predefinida, tenint la següent estructura:

- **faultCode:** Contindrà el valor del missatge de l' error escalat.

- **faultString**: Contindrà un string amb la concatenació dels camps de l'objecte d'error "moreInformation" i l'"errorCode".

Un exemple seria el següent, prenent com a objecte de resposta d' error generat:

- **httpCode**: "400"
- **httpMessage**: "Bad Request"
- **moreInformation**: "Error de validación de los campos de entrada"

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xsi:type="xsd:string">Bad Request</faultcode>
      <faultstring xsi:type="xsd:string">Error de validación de los campos de entrada (400)</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- En el cas que no es vulgui seguir aquesta plantilla SOAP per necessitats del projecte, el desenvolupador haurà de modificar l' estructura SOAP a través de la implementació d' un **GatewayScript** o **XSLT** personalitzat a continuació de la política.
- **erroresParticularesCtti**: Aquesta propietat estarà definida al YAML de l' API i servirà per afegir errors personalitzats, en funció de les necessitats del projecte, en el mapa d' errors que té la política de gestió d' errors definit de base.

```
erroresParticularesCtti:
  value: >-
    {"QBE0190" : {"httpCode": 400, "httpMessage": "Bad Request",
    "moreInformation": "No se dispone de información en el sistema sobre el
    elemento a buscar."}}
  description: >-
    **Policy: ctti-error-management** Json con los errores particulares de
    este API. **Si no se va a añadir un error particular eliminar esta
    property**
```

El format de la definició de l'error a seguir dins de la propietat "**erroresParticularesCtti**" seria el següent:

```
{
  "ErrorCodeCustom": {
    "httpCode": "<código del error http>",
    "httpMessage": "<razón del error>",
    "moreInformation": "<descripción del error>"
  }
}
```

Un exemple de la configuració d' un error dins la propietat seria el següent:

```

properties:
  errorsParticularesHost:
    value: >-
      {"CTE0190" : {"httpCode": 410, "httpMessage": "Bad Request",
"moreInformation": "No se dispone de información en el sistema sobre el
elemento a buscar."}}
    description: >-
      **Policy: ctti-error-management** Json con los errores particulares de
este API. **Si no se va a añadir un error particular eliminar esta
property**

```

Pel que fa als errors per defecte, la plantilla inclourà la definició de les etiquetes personalitzades "*x-customPaths*" i "*x-customDefinitions*" per als errors 400, 401 i 500, així com l'objecte "*outputError*" referenciat. Això s' inclou per poder proporcionar al desenvolupador un exemple d' una estructura d' error personalitzada.

```

x-customPaths:
  /path:
    get:
      responses:
        '200':
          description: success
          schema:
            type: string
        '400':
          description: Bad Request
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
        '401':
          description: Unauthorized
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
        '500':
          description: Internal Server Error
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
      consumes: []
      produces: []
      request:
        params:
          param1:
            obligatorio: true
            type: string
            maxLength: 30
        headers:
          header1:
            obligatorio: true
            type: string
            maxLength: 30
x-customDefinitions:
  outputError:
    type: object
    properties:
      httpCode:
        key: httpCode
      httpMessage:
        key: httpMessage
      errorCode:
        key: errorCode
      moreInformation:
        key: moreInformation

```

A continuació, es procedeix a detallar cadascuna de les *extensions* de les quals es podrà fer ús amb la plantilla de desenvolupament bàsica d' APIs, a través de les propietats que inclou.

- *Generar TraceID (ctti-trace-id) – Pre-Request:*

Aquesta extensió tindrà com a objectiu generar un *Traceld* en cas que no vingui informat. D'aquesta manera, es garantirà que es generi i emmagatzemi un *Traceld* únic per a cada petició a una API, la qual cosa permetrà un seguiment eficient i una millor traçabilitat de les peticions.

L' extensió no farà ús de cap propietat pròpia o definida a l' API. Verificarà si ve o no informada la capçalera '*X-Ctti-Traceld*' en la invocació a l'API i, en funció d'això, generarà un *Traceld* nou o no.

En qualsevol cas, el valor del *Traceld*, ja sigui nou o el que hagi arribat en la petició a l'API, l'emmagatzemarà en la variable de context "*cttiTraceld*", perquè pugui utilitzar-se si es requereix al llarg del flux de l'API.

- **Validació CORS (*ctti-request-cors*) – Pre-Request:**

Aquesta funció(extensió) estarà enfocada a validar els CORS que li arriben per part del sistema origen de la petició amb els CORS que l'API tindrà configurat.

A diferència de la validació CORS d' IBM, aquesta extensió custom permetrà principalment realitzar les següents accions addicionals.

- Permetrà la possibilitat que l'error per CORS sigui transformat a l'error per defecte establert en la política d'errors (*ctti-error-management*).
- Oferirà la possibilitat d' establir validacions d' una manera més custom. La política de CORS d'IBM pren tota la informació del YAML, cosa que obliga a deixar tot definit en aquest punt. L'extensió permetrà configurar cada set de validacions mitjançant una property, podent definir, per exemple, les capçaleres que permetem sense necessitat de definir-les totes dins del YAML.

Not permitted

HttpCode	HttpMessage	Descripción
403	Forbidden	Not permitted

Per part de l' aplicació d' origen haurà d' arribar, a la capçalera, la informació al camp *Origin* i *Referer*.

L' extensió requerirà tenir configurades en la definició del YAML de l' API una sèrie de propietats, les quals contindran els valors a validar contra els CORS que arribaran a la capçalera per part del client de l' API.

La plantilla de desenvolupament base de l' API inclourà aquestes propietats, que hauran de ser modificades per part del desenvolupador en funció de les necessitats del seu projecte i el que es requereixi validar. Igualment proveirà per a cada propietat una descripció del seu funcionament i valors d' exemple per defecte.

The screenshot shows the API Manager interface with the following elements:

- Top navigation: "Plantilla API" dropdown, "Open API v3.0.0" button, "Offline" toggle, "Validate" and "Save" buttons.
- Tabbed interface: "Design", "Gateway" (selected), "Test", "Endpoint", "Explorer".
- Search bar: "Search" with a magnifying glass icon.
- Properties list (left sidebar):
 - Properties (11) with a plus icon.
 - target-url
 - erroresParticularsCtti
 - ips-validas
 - get-variables
 - fichero-variables
 - cors-enabled** (highlighted with a red box)
 - cors-origins (highlighted with a red box)
 - cors-headers (highlighted with a red box)
 - cors-methods (highlighted with a red box)
 - cors-credentials (highlighted with a red box)
- Main content area: "Gateway and portal settings / Properties / cors-enabled"
 - Section title: "cors-enabled"
 - Property Name: "cors-enabled" with an "Update" button.
 - Property Value (optional): "false".
 - Checkbox: "Property Value is Base64 Encoded" (unchecked).
 - Property Description (optional): empty.
 - Editor toolbar: "Write", "Preview", "TT", "B", "I", "↶", "🔗", "🗨", "📄", "🔍", "⌵", "⌶", "⌷".

```

cors-enabled:
  description: >-
    **Extension: ctti-request-cors** Indica si se realiza la extension
    ctti-request-cors. **Si se deja a true se ejecutará la extension de
    CORS, si no se va a ejecutar la extension CORS se puede borrar**
  value: 'false'
cors-origins:
  value: https://origen1,https://origen2,https://origen3
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Origin permitidos para las peticiones a la API. Los
    valores deben ir separados por comas **Si no se va a ejecutar la
    extension CORS se puede borrar**
cors-headers:
  value: Origin,X-Requested-With,Content-Type,Accept,Authorization
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Allow-Headers permitidos para las peticiones a la API.
    Los valores deben ir separados por comas y sin espacios **Si no se va a
    ejecutar la extension CORS se puede borrar**
cors-methods:
  value: POST,GET
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene los valores de
    la cabecera HTTP Allow-Methods permitidos para las peticiones a la API.
    Los valores deben ir separados por comas **Si no se va a ejecutar la
    extension CORS se puede borrar**
cors-credentials:
  value: 'false'
  description: >-
    **Extension: ctti-request-cors** Esta propiedad contiene el valor de la
    cabecera HTTP Allow-Credentials que devolverá la API. Los posibles
    valores son true o false **Si no se va a ejecutar la extension CORS se
    puede borrar**

```

La descripció de cadascuna de les propietats necessàries seria la següent:

- **cors-enabled**. Indicador per saber si cal fer la validació de CORS. Prendrà els valors **true** (fa validació) o **false** (desactiva la validació de CORS). En el cas que no s'informi amb cap valor o no s'estableixi aquesta propietat, **per defecte** la política prendrà el valor **false** d'aquest camp i no procediria a realitzar la validació de CORS.
- **cors-origin**. Propietat on s'indican els valors permesos de la capçalera **Origin** per a les peticions a l'API, separats per comes.
Exemple: <https://origen1>,<https://origen2>,<https://origen3>

- ***cors-headers***. Propietat on s' indicaran els valors permesos de la capçalera ***Allow-Headers*** per a les peticions a l' API, separats per comes. **Exemple:** Origin, X-Requested-With, Content-Type, Accept, authorization.
- ***cors-methods***. Propietat on s' indicaran els valors permesos de la capçalera ***Allow-Methods*** per a les peticions a l' API, separats per comes. **Exemple:** POST,GET
- ***cors-credentials***. Propietat on s' indicarà el valor permès de la capçalera ***Allow-Credentials***. Tindrà el valor ***true*** o ***false***.

Per fer ús d' aquesta política, el desenvolupador haurà de revisar les propietats necessàries incloses en la plantilla de desenvolupament d' APIs base i adequar-les a les necessitats del seu projecte. Per modificar-les, es podrà realitzar tant des de la part visual del toolkit, com a través de la definició del YAML, atenent el descrit anteriorment.

Nota: Si es fa ús de la política, indicant-se a través de la propietat ***"cors-enabled=true"***, i es deixa buit algun dels valors de la resta de propietats necessàries per la política, aquesta retornarà un error indicant ***"mensaje de CORS no permitido"***.

- ***Gestió de capçaleres de seguretat (ctti-response-headers-secure) – Post-Response:***

Aquesta extensió s' executarà en la fase ***Post-Response*** del processament de l' API, o en la fase ***Post-Error***, en el cas que es produeixi un error. Seguint l'estàndard ***OWASP Secure Headers Project***, també anomenat ***OSHP***, aquesta funció (Extensió), si està activada, s'encarregarà de:

- Eliminar, per defecte, totes les capçaleres de resposta que hi hagi en aquest punt per controlar la seguretat i integritat de les sol·licituds, assegurant que no s'envii de tornada als sol·licitants cap informació sensible o no desitjada en la resposta, com una adreça IP. El desenvolupador podrà preservar les que es considerin necessàries, afegint el nom d'aquestes capçaleres a la propietat ***"cabeceras-seguridad"***. El valor de les capçaleres de resposta que es preservin serà el que tinguin en aquell moment, sense modificar.
- Generar una sèrie de capçaleres de resposta de seguretat adequades en base a l' estàndard ***OSHP***. Els valors d'aquestes capçaleres es podran configurar en la propietat ***"cabeceras-seguridad"***, comptant, al seu torn, amb valors per defecte en cas de no configurar-se.

Les capçaleres de seguretat que es generaran per part de l' extensió en la resposta de l' API seran les següents:

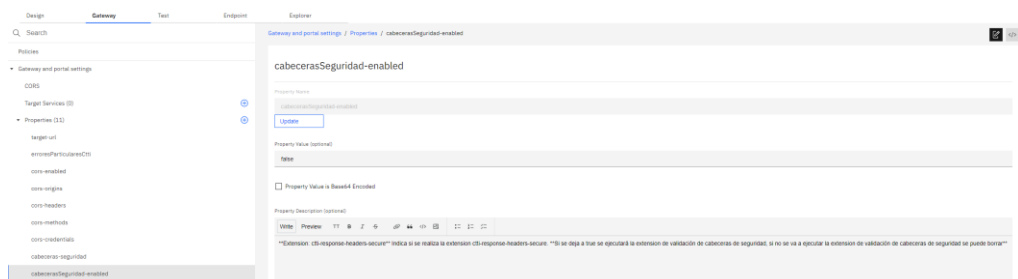
- ***Strict-Transport-Security***
És una característica de seguretat que permet a un lloc web indicar als navegadors que només s'ha de comunicar amb HTTPS en lloc d'usar HTTP.

- **X-Frame-Options**
Podrà ser usat per indicar si se li deu permetre-li a un navegador renderitzar una pàgina en un <frame>, <iframe>, <embed> o <object>.
- **X-Content-Type-Options**
Aquest encapçalat va ser introduït per Microsoft en IE 8 perquè els webmasters bloquegin el rastreig de contingut, podent transformar tipus MIME no executables en tipus MIME executables.
- **Content-Security-Policy**
Permetrà als administradors d'un lloc web controlar els recursos que l'User-Agent pot carregar a una pàgina.
- **Location**
Aquest encapçalat s'utilitzarà per redirigir l'usuari a una altra pàgina web. És important assegurar-se que només es permet redirigir a pàgines web de confiança, per evitar els atacs de phishing i altres formes de suplantació d'identitat.
- **Access-Control-Allow-Headers, Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Credentials**: Aquests encapçalats són la capçalera de CORS de resposta.
- **X-Ctti-TraceId**
Aquest encapçalat inclourà el TraceId únic de la petició de l' API, permetent un seguiment eficient i una millor traçabilitat de les peticions. Aquest TraceId serà generat en l'extensió "**ctti-trace-id**" documentada anteriorment.

La resposta de l'API, per tant, inclourà les capçaleres de seguretat esmentades en la llista anterior, juntament amb les capçaleres de resposta del backend que decideixi el desenvolupador configurar en la propietat de "**cabeceras-seguridad**".

Per fer ús d'aquesta extensió, la plantilla de desenvolupament d'APIs base tindrà inclosa les propietats "**cabecerasSeguridad-enabled**" i "**cabeceras-seguridad**", necessàries i requerides per a això.

- **cabecerasSeguridad-enabled**: Indicador per saber si cal fer la gestió de les capçaleres de seguretat. Pren els valors **true** (fa validació) o **false**. Si no s'informa amb valor o no s'estableix aquesta propietat, la política prendrà el valor false d'aquest camp i no s'executarà l'extensió de capçaleres de seguretat.



```
cabecerasSeguridad-enabled:
  description: >-
    **Extension: ctti-response-headers-secure** Indica si se realiza la
    extension ctti-response-headers-secure. **Si se deja a true se ejecutará
    la extension de validación de cabeceras de seguridad, si no se va a
    ejecutar la extension de validación de cabeceras de seguridad se puede
    borrar**
  value: 'false'
```

- **cabeceras-seguridad**: Tindrà un format JSON que permetrà la conservació de les capçaleres de resposta indicades pel desenvolupador i la customització del valor de cadascuna de les capçaleres de seguretat descrites anteriorment.



```
cabeceras-seguridad:
  description: >-
    **Extension: ctti-response-headers-secure** JSON con las cabeceras que
    se debe permitir su transmisión en la respuesta del API. Si se indican
    con valor vacío se transmitirá un valor por defecto contenido en la
    extensión **La propiedad no puede borrarse y debe contener las cabeceras
    que se desean transmitir**
  value: "{ \"Strict-Transport-Security\": \"max-age=86400; includeSubDomains\", \"X-Content-Type-Options\": \"nosniff\", \"X-Frame-Options\": \"deny\", \"Content-Security-Policy\": \"default-src 'self'\", \"Location\": \"\", \"Access-Control-Allow-Headers\": \"Content-Type, Authorization\", \"Access-Control-Allow-Origin\": \"https://example.com\", \"Access-Control-Allow-Methods\": \"GET, POST\", \"Access-Control-Allow-Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\" }
```

La propietat contindrà els següents valors d'exemple: *Strict-Transport-Security*, *X-Content-Type-Options*, *X-Frame-Options*, *Content-Security-Policy*, *Location*, *Access-Control-Allow-Headers*, *Access-Control-Allow-Origin*, *Access-Control-Allow-Methods*, *Access-Control-Allow-Credentials* i *Cabecera-Custom-Example*. Aquesta última seria un exemple de capçalera que el desenvolupador desitgi mantenir en les capçaleres de resposta del API.

```
{ \"Strict-Transport-Security\": \"max-age=86400; includeSubDomains\", \"X-Content-Type-Options\": \"nosniff\", \"X-Frame-Options\": \"deny\", \"Content-Security-Policy\": \"default-src 'self'\", \"Location\": \"\", \"Access-Control-Allow-Headers\": \"Content-Type, Authorization\", \"Access-Control-Allow-Origin\": \"https://example.com\", \"Access-Control-Allow-Methods\": \"GET, POST\", \"Access-Control-Allow-Credentials\": \"true\", \"Cabecera-Custom-Example\": \"\" }
```

Aquesta propietat haurà de ser modificada pel desenvolupador per indicar els valors corresponents de les capçaleres de seguretat i els noms de les capçaleres de resposta que es requereixin mantenir en funció de les necessitats del projecte. En el cas que no se li assigni cap valor a alguna capçalera de seguretat, la política la generarà amb el corresponent valor per defecte:

- *Strict-Transport-Security* → "max-age=86400; includeSubDomains"

- *X-Content-Type-Options* → "deny"
- *X-Frame-Options* → "nosniff"
- *Content-Security-Policy* → "default-src 'none'"
- *Location* → En el cas que no s'informi, s'emplenarà amb valor buit
- *Access-Control-Allow-Headers* → Si no se li ha informat cap valor per a aquesta capçalera a la propietat, la política no la tindrà en compte.
- *Access-Control-Allow-Origin* → Si no se li ha informat cap valor per a aquesta capçalera a la propietat, la política no la tindrà en compte.
- *Access-Control-Allow-Methods* → Si no se li ha informat cap valor per a aquesta capçalera a la propietat, la política no la tindrà en compte.
- *Access-Control-Allow-Credentials* → Si no se li ha informat cap valor per a aquesta capçalera a la propietat, la política no la tindrà en compte.
- *X-Ctti-TraceId* → Valor de TraceId que s'hagi generat en l'extensió "*ctti-trace-id*"

Aquesta propietat podrà ser modificada a través de la interfície visual del toolkit, *API*→*Gateway*→*Properties*→*cabeceras-seguridad*, o modificant directament la definició del YAML, ja sigui des del toolkit o des d'un altre programa d'editor de textos.

La plantilla d' APIs base també inclourà una pre-configuració, a tall d' exemple, del valor assignat a certes propietats, tant pròpies de l' API, com de les requerides per les polítiques i extensions, per a cadascun dels catàlegs de CTTI en els quals es pot desplegar l' API.

En concret, vindran com a exemple els valors de la propietat *target-url* i de les propietats requerides per l'extensió de validació de CORS (*cors-enabled*, *cors-origins*, *cors-headers* i *cors-credentials*), i les requerides per l'extensió de validació de capçaleres de seguretat (*cabeceras-seguridad* i *cabecerasSeguridad-enabled*).

```

catalogs:
  public_pre:
    properties:
      target-url: https://exampledesa.com/test
      cors-enabled: 'false'
      cors-origins: https://origen1,https://origen2,https://origen3
      cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
      cors-credentials: 'false'
      cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400; include
      cabecerasSeguridad-enabled: 'false'
  public:
    properties:
      target-url: https://exampleint.com/test
      cors-enabled: 'false'
      cors-origins: https://origen1,https://origen2,https://origen3
      cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
      cors-credentials: 'false'
      cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400; include
      cabecerasSeguridad-enabled: 'false'
  privat_pre:
    properties:
      target-url: https://examplepro.com/test
      cors-enabled: 'false'
      cors-origins: https://origen1,https://origen2,https://origen3
      cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
      cors-credentials: 'false'
      cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400; include
      cabecerasSeguridad-enabled: 'false'
  privat:
    properties:
      target-url: https://examplepro.com/test
      cors-enabled: 'false'
      cors-origins: https://origen1,https://origen2,https://origen3
      cors-headers: Origin,X-Requested-With,Content-Type,Accept,Authorization
      cors-credentials: 'false'
      cabeceras-seguridad: "{ \t\"Strict-Transport-Security\": \"max-age=86400; include
      cabecerasSeguridad-enabled: 'false'

```

Nota: Els valors que s'indiquen a la plantilla són només d'exemple, hauran de ser modificats pel desenvolupador en base a les necessitats del seu projecte.

El desenvolupador, d'aquesta manera, podrà configurar per al seu API el valor de les propietats que consideri necessàries o requereixi el seu projecte, en funció del catàleg de CTTI en el qual s'hagi de desplegar.

Quant a com s'haurà d'omplir la informació de les etiquetes "*servers*" o "*basePath*" (depenent de si es fa servir OpenAPI 3.0 o 2.0), on es definirà la ruta base o URL principal que s'utilitzarà per accedir a l'API de forma inequívoca dins de l'API Manager de CTTI, està detallat en el punt anterior [3.1.2 Via d'accés Base \(Base Path\)](#).

REF: 2023-00001

- OpenAPI 3.0:

```
servers:
  - url: /codi/basepath-api
```

- OpenAPI 2.0:

```
basePath: /codi/basepath-api
```

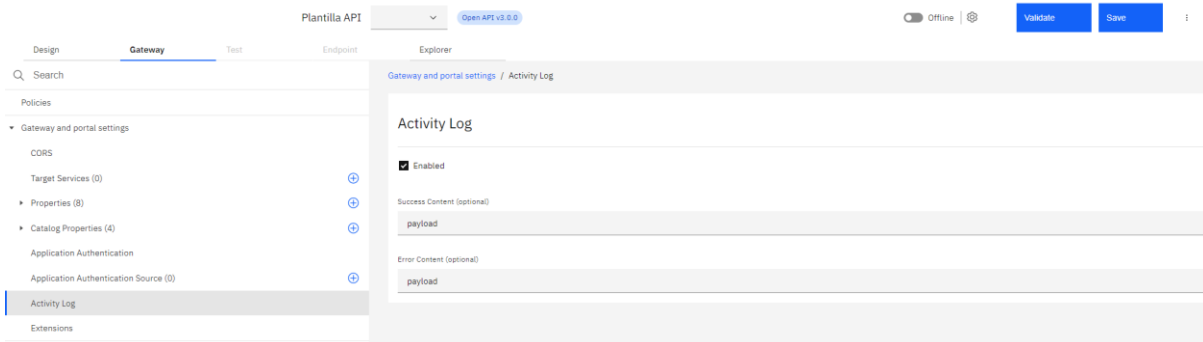
A nivell de seguretat, a la plantilla base d'APIs estarà inclosa també la configuració de la seguretat mínima d'API requerida a l'entorn de CTTI, que seria l'autenticació mitjançant el *ClientId*, de manera que qualsevol petició que es realitza a aquesta API necessitarà informar la capçalera "Client Id" perquè es pugui processar correctament.

```
security:
  - clientID: []
components:
  securitySchemes:
    clientID:
      type: apiKey
      name: X-IBM-Client-Id
      in: header
```

El nivell de seguretat es podrà augmentar segons la necessitat i característiques del projecte. Per saber quin tipus de seguretat s'ha d'utilitzar, revisar la normativa de seguretat d'API del CTTI.

Finalment, la plantilla també inclourà en l'acoblament l'activació de la política de log pròpia d'IBM API Connect, la qual permetrà configurar les preferències de registre de l'activitat d'API que s'emmagatzema en les dades analítiques d'IBM® API Connect.

```
activity-log:
  enabled: true
  success-content: payload
  error-content: payload
```



Nota: Si es vol aprofundir més en el funcionament i en les propietats de la política "activity-log" d'IBM API Connect, s'indica el link a la pàgina oficial d'IBM (https://www.ibm.com/docs/es/api-connect/10_reserved_instance?topic=policies-activity-log).

4.4 Aplicació i ús de la plantilla estesa

Partint de la plantilla base comentada en el punt anterior, a continuació, es descriurà una possible expansió de mínims de la plantilla, mostrant l'ús de la resta de polítiques a aplicar en la definició del YAML de l'API, així com les seves dades necessàries per al seu ús.

El llistat de les polítiques addicionals que es trobaran a la disposició dels desenvolupadors és el següent:

- ***ctti-validate-ip (no disponible de moment):***

Es tracta d'una política que permetrà controlar qui invoca o no l'API en funció de la IP que ve informada a la capçalera de la petició "***x-client-ip***". Amb això es reforçarà la capa de seguretat en les invocacions realitzades sobre les APIs, permetent controlar i restringir les IPs des de les quals s'invoca una API concreta.

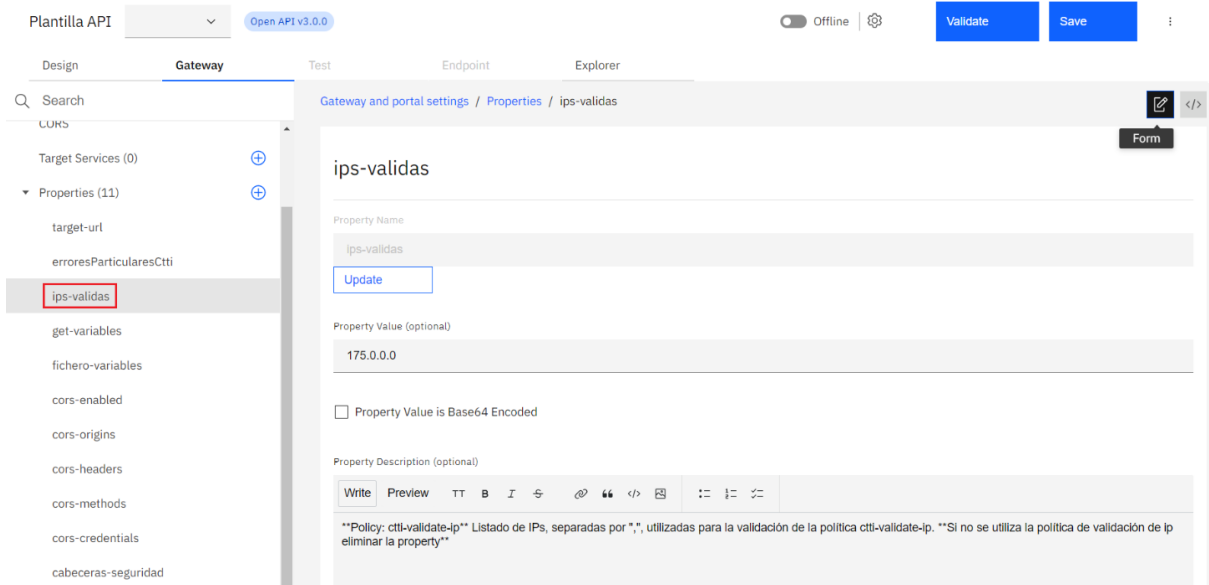
Per al seu funcionament, la política requerirà la següent informació:

- **ipsValidas:** Sera una llista blanca de les IPs permeses per realitzar trucades a l' API. Aquesta llista es definirà en una propietat de la pròpia API anomenada ***ips-validas***. Serà el desenvolupador qui creï i informi aquesta propietat amb les IPs que s' han establert com a vàlides, usant com a separador una coma. Per exemple: 172.25.36.34,89.15.34.87,125.36.75.99.
En cas que aquesta propietat no s'hagi creat dins de la definició del YAML de l'API, la Política retornarà un error indicant que la propietat no està informada, en el camp "***moreInformation***".
- **ipOrigen:** Adreça IP del client que ve a la capçalera en el paràmetre ***X-Client-IP***.

En el cas que la propietat "***ips-validas***" s'hagi creat en la definició del YAML de l'API però estigui buida, la política no realitzarà cap validació sobre les IPs d'origen, permetent d'aquesta manera totes les trucades d'API provinent de qualsevol IP.

D'altra banda, si la ***IPOrigen*** no es troba dins de la llista blanca d'IPs vàlides de la propietat "***ips-validas***", es generarà un error amb els valors ***statusCode 401*** i ***errorMessage Unauthorized*** perquè sigui tractat en la política de **Gestió d'Errors (ctti-error-management)**, interrompent l'execució de l'API.

La propietat es podrà visualitzar i modificar a través de la interfície visual del toolkit, com en la definició del YAML de l' API.



```
ips-validas:
value: 175.0.0.0
description: >-
  **Policy: ctti-validate-ip** Listado de IPs, separadas por ",",
  utilizadas para la validación de la política ctti-validate-ip. **Si no
  se utiliza la política de validación de ip eliminar la property**
```

- *ctti-get-variables*

L'objectiu d'aquesta política serà recuperar el valor de les variables emmagatzemades en diferents arxius a DataPower (Gateways). Això permetrà tenir un control sobre el procés de retornar valors que no es volen exposar al YAML de les APIs. D'aquesta manera, la informació sensible no serà visible al YAML de l'API, sinó que es recuperarà de forma segura.

Els fitxers allotjats als DataPowers podran contenir tot tipus de variables (configuracions, usuaris, endpoints, etc.). D' aquesta manera, el desenvolupador de l' API podrà accedir al fitxer i al valor de les variables que es necessitin, indicant ambdós en les propietats de l' API.

Per assegurar que el contingut es pugui recuperar de manera correcta, els fitxers que contenen les variables hauran de definir-se seguint la següent estructura:

```
exports.variables = Object.freeze({
  string1: 'valorString1',
  string2: 'valorString2',
  number1: valorNumber1,
  number2: valorNumber2,
  ...
});
```

Per poder dur això a terme i accedir a les diferents variables emmagatzemades, el desenvolupador necessita configurar dues propietats a l' API, de les quals farà ús la política:

- **get-variables:** Contindrà el llistat de variables, separades per "," que es volen recuperar amb la política.

```
get-variables:
  value: var1,var2
  description: >-
    **Policy: ctti-get-variables** Listado de variables, separadas por ",",
    que se desean recuperar con la política ctti-get-variables. **Si no se
    utiliza la política eliminar la property**
```

- **fichero-variables:** Contindrà la ruta del fitxer sobre el qual buscar les variables. El seu valor ha de seguir la següent estructura: *espacio/nombrefichero.js* (Ex: *cd3292/fichero1.js*)

```
fichero-variables:
  value: espacio/nombrefichero.js
  description: |-
    **Policy: ctti-get-variables** Ruta del fichero sobre el que buscar las
    variables con la política ctti-get-variables. **Si no se utiliza la
    política eliminar la property**
```

La política de ctti-get-variables comprovarà l'existència de cadascuna de les variables introduïdes en la propietat "**get-variables**", en el fitxer corresponent indicat en la propietat "**fichero-variables**". Si el valor de la variable introduïda no està en el fitxer de variables, es generarà un error perquè sigui tractat en la política de Gestió d'Errors (**ctti-error-management**), interrompent l'execució de l'API.

Nota: Per poder desplegar o modificar fitxers allotjats als Gateways, el desenvolupador haurà d' obrir [un tiquet JIRA d' ACOAPIM](#) a l' OFT, indicant-li tota la informació relativa al canvi a realitzar, com el motiu i una descripció breu del canvi, adjuntar els fitxers necessaris, etc. Després de la validació per part de l'OFT de la informació aportada al tiquet JIRA, aquesta l'escalarà a l'equip del CPD a través d'un tiquet Remedy, perquè es dugui a terme el canvi.

- **ctti-header-arch**

A través d'aquesta política, es disposarà d'unes capçaleres d'arquitectura amb nous camps que s'inclouran en totes les peticions que passen per API Connect. Aquestes capçaleres es

faran servir per controlar, negociar i millorar la interacció entre clients i servidors, així com per garantir la seguretat i eficiència de les comunicacions.

Aquesta funcionalitat resultarà útil en diversos contextos, com a l'hora d'informar l'identificador únic de la petició (**UUID**) a les capes d'integració subsegüents, o incloure el token utilitzat en l'autentiació per a la seva validació en nivells inferiors del sistema. La política emplenarà els camps de capçaleres d'arquitectura necessaris per a una correcta comunicació, evitant que el desenvolupador ho hagi de fer en cadascuna de les APIs en les quals s' hagi d' utilitzar aquesta política.

Per a l' ús d' aquesta política no es requerirà la configuració de propietats addicionals a l' API, per la qual cosa el desenvolupador podria implantar directament aquesta política en l' acoblament, preferiblement a l' inici de l' execució.

Les capçaleres d'arquitectura que implementarà la política seran les següents*:

- **Trace-Id**: Identificador únic de la petició. Es manarà el valor de la capçalera de "**X-Ctti-Traceld**". Aquest valor es manarà sempre ja que, de no venir informat a la capçalera, es genera en l'extensió "**ctti-trace-id**".
- **Authorization**: Serà el token d' autenticació contra els recursos específics del backend. Es recollirà del camp **Authorization** de la capçalera.
- **X-Forwarded-For**: Identificarà l' adreça IP del client que va realitzar la sol·licitud. Es manarà el valor del paràmetre de capçalera **X-Client-IP**.
- **Accept-Language**: Indicarà el llenguatge en el qual es vol obtenir el resultat. Per defecte, s'informarà amb el valor "**ca-ES**".
- **User-Agent**: Identificarà el client que està fent la sol·licitud. Es manarà a la capçalera de la petició amb el mateix nom ("**User-Agent**").

*** Totes les capçaleres d'arquitectura indicades seran opcionals.**

Perquè es pugui implementar aquestes capçaleres d'arquitectura, a la capçalera origen de la petició s' hauria d' informar els camps següents:

- X-Ctti-Traceld
- Authorization
- X-Client-IP
- Accept-language
- User-Agent

Notes:

- En el cas que no vingui informada la capçalera '**X-Client-IP**', la política ometrà aquesta capçalera i no se l'informarà al backend.
- En el cas que no s'envii la capçalera '**Authorization**', al backend no se l'informarà de cap token de verificació.
- En el cas que no s'envii la capçalera '**X-Ctti-Traceld**' en la petició, s'autogenerarà en l'extensió "**ctti-trace-id**".

- En el cas que no s'envii la capçalera '**User-Agent**' en la petició, se li assignarà el valor predeterminat ("**NOINFO**") a la capçalera d'arquitectura que s'envia al backend.
- En el cas que no s'envii la capçalera '**Accept-Language**' en la petició, se li assignarà el valor predeterminat ("**ca-ES**") a la capçalera d'arquitectura que s'envia al backend.

- **ctti-validate-response**

Aquesta política és similar a la de validació de request (**ctti-validate-request**), indicada en [l'apartat 4.3](#), però, en aquest cas, el disseny està enfocat al fet que s'utilitzi sempre al final de l'assembly. L'ús d'aquesta política està enfocat a realitzar la validació del missatge de sortida de l'API (amb la resposta del backend) contra l'esquema generat en les operacions del disseny de l'API.

Amb aquesta mesura, s'enforteix la capa de seguretat dels desenvolupaments en requerir que els missatges sortints de l'API compleixin amb el format esperat (paràmetres, queries, etc.) definit al YAML. Això assegura la qualitat de les APIs i dels backends. Per exemple, es podrien detectar falsos positius que puguin retornar-se com a resposta per part del backend.

Per validar la resposta, la política es basarà en la definició dels objectes de resposta que s'inclouran en la definició del YAML de l' API. És en aquesta definició on el desenvolupador haurà d' incloure totes aquelles validacions que s' hagin de realitzar. Si no s' especifiquen els elements a validar dins del YAML de l' API, la política no realitzarà cap validació a l' objecte de resposta.

Aquesta política no requerirà l' ús d' una propietat definida a l' API però, com s' ha indicat a dalt, exigirà al desenvolupador definir uns esquemes custom de validació que posteriorment es validaran amb el missatge de resposta de l' API.

Nota: Cal destacar que la política només realitzarà la validació sobre el cos (**body**) de la resposta, si s'ha enviat en format **JSON**, altrament, la política no procedirà a realitzar la validació del body.

Per poder definir els esquemes custom, s' ha de seguir les directrius següents:

L'esquema que consultarà la política per veure les dades a validar ha d'estar contingudes dins l'etiqueta del YAML **x-customPaths** → **path** → **verb** → **response** → **httpCode** → **schema** → **\$ref** que es troba dins de l'etiqueta **x-ibm-configuration**.

Un exemple de com podria ser el contingut de **x-customPaths** seria el següent:

```
x-customPaths:
  /gestion-consulta:
    post:
      responses:
        '200':
          description: success
          schema:
            type: string
        '201':
          description: 201 Create
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputData'
        '400':
          description: 400 Bad Request
          schema:
            $ref: '#/x-ibm-configuration/x-customDefinitions/outputError'
      request:
        params:
          param1:
            obligatorio: true
            type: string
            maxLength: 30
        headers:
          header1:
            obligatorio: true
            type: string
            maxLength: 30
        body:
          $ref: '#/x-ibm-configuration/x-customDefinitions/inputData'
```

Per incloure dins de *x-customDefinitions* l' esquema de validació de resposta que es requereixi, el qual s' hauria referenciat en *x-customPaths* anteriorment, i que tindrà tots els camps i les seves propietats de validació a realitzar per la política, caldria seguir els següents passos:

- Dins de *x-customDefinitions* es crearà una nova etiqueta, que haurà de tenir el mateix nom que el que s' hagi especificat en la referència dins de *x-customPaths*, com a resposta a una operació de l' API. En el nostre cas d'exemple exposat anteriorment, seria *"outputData"*.
- L'objecte de l'esquema a crear haurà de tenir el camp *"type"*, per indicar el tipus d'objecte que és (object, array, etc.), i el camp *"properties"*, on s'inclouran tots aquells camps que han de ser validats per la política.
- Dins de *"properties"* s'inclouran tots aquells camps i les corresponents propietats de validació que la política ha de realitzar sobre cadascun d'ells.

Per a l'exemple de **"outputData"**, una definició del YAML que seguiria aquestes directrius seria la següent:

```
x-customDefinitions:
  outputData:
    type: object
    properties:
      descripcion:
        obligatorio: false
        type: string
        maxLength: 3200
      flag:
        obligatorio: true
        type: boolean
      estado:
        obligatorio: false
        type: string
        list: ['OK', 'KO']
      arrayDeObjetos:
        $ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'
    arrayDeObjetos:
      type: array
      properties:
        descripcion:
          obligatorio: false
          type: string
          maxLength: 3200
        flag:
          obligatorio: true
          type: boolean
```

Nota: Com es pot veure en l'exemple, també es podran definir objectes dins d'objectes a través d'una referència a la definició d'aquest amb el comando **"\$ref"**, com passa per a l'objecte **"arrayDeObjetos"** (**\$ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'**), la qual es troba dins de la mateixa estructura sota **x-customDefinitions**.

Per indicar si l'esquema admet propietats addicionals, s'haurà d'incloure l'etiqueta **"additionalProperties"** amb el seu valor booleà corresponent dins de la definició referenciada.

Un exemple de definició del YAML seria el següent:


```
x-customDefinitions:
  outputData:
    type: object
    properties:
      descripcion:
        obligatorio: false
        type: string
        maxLength: 3200
      flag:
        obligatorio: true
        type: boolean
      estado:
        obligatorio: false
        type: string
        list: ['OK', 'KO']
      arrayDeObjetos:
        $ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjetos'
    additionalProperties: true
```

Les propietats de validació que permetrà aquesta política per a cadascun dels camps que s' especifiquin en la definició del YAML seran les següents:

- **obligatorio**
 - Indicarà si un camp és obligatori o no.
 - Valors possibles: **true** o **false**.
- **type**
 - Tipus del camp indicat.
 - Valors possibles: **string**, **number**, **boolean**, **object** o **array**.
 - En el cas que sigui un arrelament d' objectes o un conjunt d' objectes, s' haurà de definir com a **object**.
 - En el cas que sigui un **array** simple, s'ha de definir com **array**
- **regexp**
 - Expressió regular que haurà de complir el camp. Al YAML s' haurà d' incloure entre cometes. **Exemple:** `"/^d{2}/d{2}/d{2,4}$/"`.
- **list**
 - Llista de possibles valors on haurà d' estar el valor que se li passi. Aquesta propietat **SI** exigeix que el valor informat en la petició coincideixi amb el definit en l' esquema, incloent-hi les majúscules i minúscules. **Exemple:**
 - Llista: `[MuyBien,AlgoDesgastado,RecienFabricado]`
 - Valor vàlid: `MuyBien`
 - Valor NO vàlid: `muybien`
- **upperCaseList**
 - Llista de possibles valors en majúscules on haurà d' estar el valor que se li passi. Aquesta propietat **NO** exigeix que coincideixin la capitalització (les minúscules i majúscules) del valor informat en la petició amb l'esquema. **Exemple:**

- Llista: *[EXCELENTE,ACEPTABLE,DEPLORABLE]*
- Valor vàlid: *Excelente*
- **maxLength**
 - Longitud **màxima** que permetrà el camp.
- **minLength**
 - Longitud **mínima** que permetrà el camp.
- **length**
 - Longitud **exacta** que haurà de tenir el camp.
- **not**
 - Que **no** sigui el valor especificat.
- **literal**
 - Valor **literal** que haurà de portar el camp que es valida.
- **lengthDecimal**
 - S' amidarà la **longitud** de la part **decimal** d' un valor, i que el nombre de xifres decimals no superi el definit.
- **array**
 - Validarà que un camp és un **array** amb elements simples, i els seus elements han de tenir el tipus especificat en aquest camp.
 - Valors possibles: *number, string* o *boolean*.
- **arrayObject**
 - Validarà un **array d' objectes** i, en l' esquema, caldrà especificar que propietats té cada objecte.

- **ctti-custom-log**

Aquesta política permetrà guardar el contingut d' una variable custom que el desenvolupador consideri que és útil per poder monitoritzar la funcionalitat de l' API.

La política guardarà els camps com un clau-valor dins del missatge que es mana per defecte a l'**Analytics**.

La política tindrà els següents paràmetres (propietats internes) d'entrada que el desenvolupador haurà de configurar per poder dur a terme el funcionament d'aquesta política:

- **key**: Contindrà el nom del camp en el qual es vol guardar el log dins de l' objecte que s' envia a l' Analytics.
- **message**: Contingut que es vol transmetre a l' Analytics, podent ser:
 - Un text pla com, per exemple: *"Texto de prueba"*.

message
Message del log.

Texto de prueba

- El valor d' una variable que hagi estat definida abans de l' execució de la política. Per a això s'ha d'indicar de la següent forma: ***\$(nombre de la variable)***.

message
Message del log.

\$(nombre_variable)

- Combinació de text pla i el contingut d'una variable com, per exemple, ***"La persona que realizó la prueba es: \$(nombre)"***. On ***"nombre"*** és el nom de la variable de context que es va crear anteriorment en el flux de l'API.

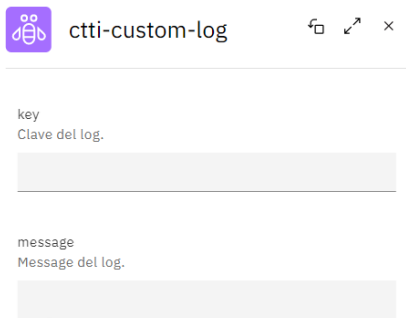
message
Message del log.

La persona que realizó la prueba es: \$(nombre)

- Un objecte json, el qual serà transformat a string com, per exemple, ***"{"nombre":"Antonio", "apellido":"Sanchez"}"***.

message
Message del log.

{"nombre":"Antonio", "apellido":"Sanchez"}



Cal destacar també que, si es posen diverses polítiques *ctti-custom-log* en l'assembly amb el mateix valor en el camp *'key'*, es trepitjarà el seu valor, enviant-se a l'Analytics únicament el valor recuperat en l'última política.

5 Consideracions a tenir en compte per a l' ús de la plantilla

A continuació, s' indiquen algunes consideracions que s' han de tenir en compte en fer ús d' aquesta plantilla:

- Es podran eliminar les propietats predefinides a la secció **"Properties"** de l'API Designer, sempre que el desenvolupador estigui segur que no es vagi a usar aquestes propietats. **Nota:** Les propietats que són imprescindibles, les quals no s' haurien d' esborrar de la plantilla base, estan vinculades a les polítiques i extensions especificades a [l' apartat 4.3](#).
- L'acoblament definit a la plantilla base d'API podrà ser modificat pel desenvolupador en funció de les necessitats del seu projecte, sempre que s'inclouin totes les polítiques, i les seves propietats, especificades a la plantilla base ([apartat 4.3](#))
- En l' entorn de CTTI es requereix un nivell mínim de seguretat, que és l' autenticació mitjançant el **ClientId**. Per defecte, a la plantilla base s'inclourà la configuració d'aquesta mesura de seguretat però, segons la necessitat/característiques de cada projecte, es podria reforçar aquesta capa de seguretat amb l'ús de, per exemple, les combinacions de **ClientId+Client Secret** o **ClientId+OAuth2** per part del desenvolupador. Per conèixer quina seguretat hauria de tenir el teu API, pots revisar la normativa de seguretat d'APIs de CTTI.

Un exemple seria el següent:

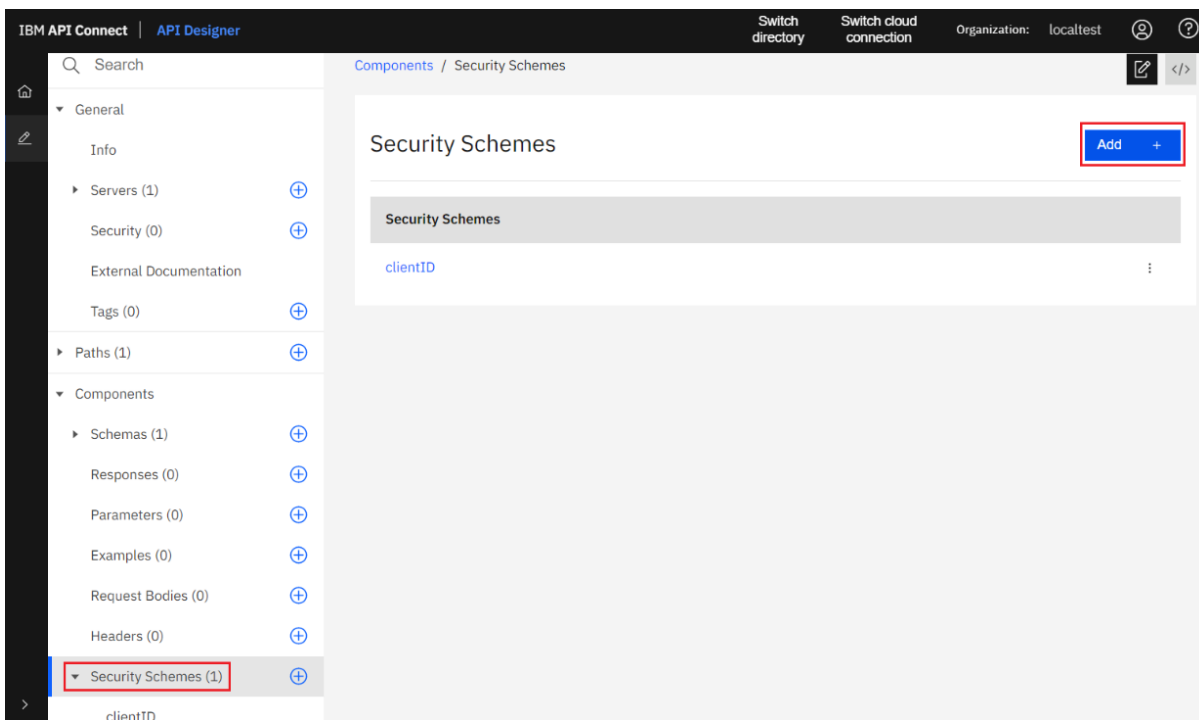
Required Security Schemes	Security Scheme Type
clientID	apiKey ↑ ↓ ✎ 🗑️
gicar 2 × Scopes ▼	oauth2

Combinació de seguretat "Client Id + OAuth2 (Gicar)"

Required Security Schemes	Security Scheme Type	
clientID	apiKey	↑ ↓ ✎ 🗑
clientSecret	apiKey	

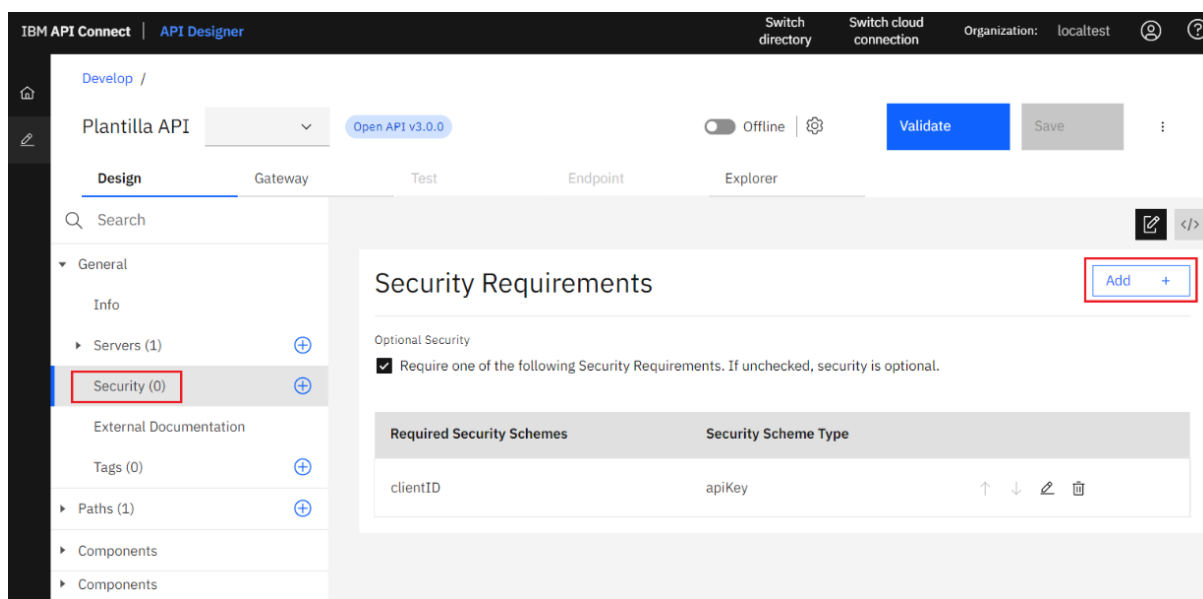
Combinació de seguretat "Client Id + Client Secret"

Per fer ús d' aquestes combinacions de seguretat, caldria definir els respectius components d' esquema de seguretat d' autenticació corresponents a Client Secret i OAuth2. Per a això, s'haurà d'accedir a la secció **"Security Schemes"** de l'apartat **"Components"** dins de l'API Designer (per a APIs amb versió **OpenAPI 3.0**). En el cas que l'API tingui la versió OpenAPI 2.0, s'accedirà directament a la secció **"Security Schemes"** (si es fa servir la interfície gràfica de l'API Designer, atès que el nom de la secció al YAML seria **"securityDefinitions"**) per a la definició dels esquemes de seguretat.



The screenshot shows the IBM API Designer interface. The top navigation bar includes 'Switch directory', 'Switch cloud connection', and 'Organization: localtest'. The main content area is titled 'Components / Security Schemes'. On the left, a sidebar menu shows various components, with 'Security Schemes (1)' highlighted. The main panel displays a table with one entry: 'clientID'. A red box highlights the 'Add +' button in the top right corner of the main panel.

Un cop s' hagin definit els components de l' esquema de seguretat, es procedirà a implantar aquestes mesures de seguretat a l' API. Això s'haurà de configurar a la secció **"Security"** dins de l'API Designer.



- Atès que el Toolkit API Designer no disposa de la funció de guardat automàtic, després de qualsevol modificació que s'hagi realitzat sobre aquesta plantilla, s'haurà de salvar donant al botó "Save", de cara a no perdre els canvis realitzats.

6 Annexos

6.1 Fitxer de plantilla

S'adjunta la definició YAML de la plantilla bàsica a importar a **OpenAPI 3.0**, per poder treballar dins de l' API Manager de CTTI. Adicionalment, també s'inclou la plantilla en **OpenAPI 2.0**, per si es requereix la seva utilització a causa dels requisits del projecte.



plantillaAPI.yaml



plantilla-api_2.0.yaml

6.2 Errors base definits en la política de gestió d' errors

A continuació, s'inclou una taula amb la correspondència de codi, missatge i descripció, per a cadascun dels errors custom que la política de gestió d'errors (**ctti-error-management**) inclou per defecte, que podran ser referenciats a l'hora d'aixecar una excepció des dels GatewayScripts.

Error	httpCode	Mensaje	Descripción
E0400	400	Bad Request	Error de validación de los campos de entrada
E0401	401	Unauthorized	No está autorizado para ejecutar el servicio
E0403	403	Forbidden	No tiene los privilegios adecuados para ejecutar el servicio
E0404	404	Not Found	Recurso no encontrado
E0405	405	Method Not Allowed	Método HTTP no encontrado
E0409	409	Conflict	Se ha detectado un conflicto en el recurso
E0429	429	Too Many Request	Se ha detectado demasiadas peticiones a un recurso
E0500	500	Internal Server Error	Servicio no disponible momentáneamente
E0503	500	Service Unavailable	Servicio no disponible momentáneamente
E0600	500	Internal Server Error	Error en el servicio de backend
E0800	500	Internal Server Error	Error en política de usuario
E0900	500	Internal Server Error	Error de conexión con backend
E0XXX	500	Internal Server Error	Error en el servicio de backend

Nota: La política permetrà incloure nous errors custom per a poder ser referenciats en aixecar una excepció, en funció de les necessitats del projecte.

Per poder dur a terme això, s'haurà de realitzar a través de la propietat **"erroresParticularsCtti"**, la qual estarà inclosa en la plantilla de desenvolupament d'APIs, on s'especificarà l'error custom a tenir en compte seguint el següent format:

```
{
  "ErrorCodeCustom" : {
    "httpCode": "<código del error http>",
    "httpMessage": "<razón del error>",
    "moreInformation": "<descripción del error>"
  }
}
```

Un exemple seria el següent:

```
properties:
  erroresParticularesCtti:
    value: >-
      {"CTE0190" : {"httpCode": 410, "httpMessage": "Bad Request",
        "moreInformation": "No se dispone de información sobre en el sistema sobre el
        elemento a buscar."}}
    description: >-
      **Policy: ctti-error-management** Json con los errores particulares de
      este API. **Si no se va a añadir un error particular eliminar esta
      property**
```

Per mantenir la concordança amb la nomenclatura del codi d'error definit a la taula d'errors de la política, en el cas que el codi d'error custom indicat a la propietat no comenci per **"EO"**, la política internament, en recollir-lo de la propietat i bolcar-lo a la taula d'errors, l'afegirà.

Prenent l' exemple exposat anteriorment, quedaria de la manera següent:

- Error custom indicat a la propietat: **"CTE0190"**
- Error que carregarà la política en la taula d' errors per a la seva utilització: **EOCTE0190**

Es recomana que els errors custom que es defineixin mantinguin, dins del que es pugui, la nomenclatura **EOXXX**.

Nota: L'error **EOXXX** representa l'error que es retornaria en el cas que, en escalar-se un error, s'hagi informat un codi d'error (**EOAAA**) que no es trobi a la taula d'errors o hagi estat afegit a través de la propietat proporcionada per la política.

6.3 Plantilla de producte

Per al disseny d' un producte, es disposa també de la següent plantilla genèrica, de cara a poder desenvolupar el seu propi producte realitzant les modificacions necessàries sobre la següent plantilla base:

```
info:
  version: 1.0.0
  title: XXXX Project name
  name: XXXX-project_name
  categories:
  - XXXX
  - XXXX/project_name
gateways:
```



```
- datapower-api-gateway
plans:
  default-pla:
    title: Default Pla
    approval: true
    description: Default Plan
    rate-limits:
      default:
        value: 100/1hour
        hard-limit: true
    burst-limits:
      default:
        value: 10/1minute
    apis:
      apiname-apiversion: {}
  bronze:
    title: Bronze
    approval: true
    rate-limits:
      default:
        value: 1000/1hour
        hard-limit: true
    burst-limits:
      default:
        value: 100/1minute
    apis:
      apiname-apiversion: {}
  silver:
    title: Silver
    approval: true
    rate-limits:
      default:
        value: 2000/1hour
        hard-limit: true
    burst-limits:
      default:
        value: 200/1minute
    apis:
      apiname-apiversion: {}
  gold:
    title: Gold
    approval: true
    rate-limits:
```

```

    default:
      value: 4000/1hour
      hard-limit: true
    burst-limits:
      default:
        value: 400/1minute
  apis:
    apiname-apiversion: {}
visibility:
  view:
    type: authenticated
    orgs: []
    tags: []
    enabled: true
  subscriu:
    type: authenticated
    orgs: []
    tags: []
    enabled: true
product: 1.0.0
apis:
  apiname-apiversion:
    $ref: nombreficheroapi.yml

```

Els components claus del producte, subjecte a canvis que realitzin els desenvolupadors segons les seves necessitats, són els següents:

- **Title:** Títol del producte. S'ha de seguir la següent estructura: *{Codi_Projecte Nom_Projecte}*
Ex: 3292 Projecte Alpha
- **Name:** Identificador únic del producte. S'ha de coincidir amb el títol, seguint la següent estructura: *{codi_projecte-nom_projecte}*
Ex: 3292-projecte-alpha
- **Categories:** Les categories estableixen una visualització jeràrquica de productes API en el portal de desenvolupadors. Segons la nomenclatura establerta per CTTI, s' ha de seguir l' estructura següent:
 - '[codi]'
 - '[codi]/[project_name]'

On **codi** és el codi del projecte, i **project_name** és el nom del projecte (separades per guió).

Ex:

- '1234'

REF: 2023-00001

- '1234/projecte-alpha'
- **Plans/{plan}/apis:** En aquest camp s'assignen els apis que es vulgui associar a cadascun dels plans definits en el producte. Els valors d' aquest camp es passaran com a referències definides sota l' etiqueta **apis**.
- **Apis:** En aquesta secció es defineixen les referències de les apis, usant el nom del fitxer **.yaml** de l'API.

Ex:

Projecte Alpha-1234

\$ref: proyecto-alpha.yaml

S'adjunta la definició YAML de la plantilla bàsica del producte a importar a l'API Designer (toolkit):

