 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

Revisió	Revisió	3292	N. revisió doc.:	Aprovat per	Disseny detallat de la solució
1.0	Accenture				
1.1	Accenture				


Registre de canvis del document

Revisió	Registre de canvis del document	Data Modificació	Revisió
1.0	Tots	12/11/2024	
1.1	Punt 3.3	17/12/2024	Es modifiquen errades al punt 3.3 de les capçaleres de seguretat


RESPONSABLE DEL DOCUMENT: Accenture / CTTI

Í N D E X

1. INTRODUCCIÓ	3
1.1 Propòsit	3
1.2 Abast	3
1.3 Definicions, acrònims i abreviatures	4
1.4 Referències	4
2. DISSENY DETALLAT (POLÍTIQUES)	4
2.1 Component Validar IP Origen (ctti-validate-ip) (no disponible de moment)	5
2.1.1 Disseny del component	5
2.1.2 Escenaris i algorismes	5
2.1.3 Altres vistes i informació addicional.....	6
2.2 Component Recuperació de variables (ctti-get-variables)	6
2.2.1 Disseny del component	6
2.2.2 Escenaris i algorismes	8
2.2.3 Altres vistes i informació addicional.....	8
2.3 Component Capçalera Arquitectura (ctti-header-arch)	8
2.3.1 Disseny del component	8
2.3.2 Escenaris i algorismes	10
2.3.3 Altres vistes i informació addicional.....	10
2.4 Component Logs (ctti-custom-log)	11
2.4.1 Disseny del component	11
2.4.2 Escenaris i algorismes	12
2.4.3 Altres vistes i informació addicional.....	12

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

2.5	Component Logs Invoke (ctti-invoke-log)	13
2.5.1	Disseny del component	13
2.5.2	Escenaris i algorismes	14
2.5.3	Altres vistes i informació adicional.....	14
2.6	Component Validació de request (ctti-validate-request)	15
2.6.1	Disseny del component	15
2.6.2	Escenaris i algorismes	21
2.6.3	Altres vistes i informació adicional.....	22
2.7	Component Validació de response (ctti-validate-response)	23
2.7.1	Disseny del component	23
2.7.2	Escenaris i algorismes	28
2.7.3	Altres vistes i informació adicional.....	29
2.8	Component Gestió d'errors (ctti-error-management)	29
2.8.1	Disseny del component	29
2.8.2	Escenaris i algorismes	43
2.8.3	Altres vistes i informació adicional.....	43
3.	DISSENY DETALLAT (EXTENSIONS)	44
3.1	Component Pre-Request - Generar TraceId (ctti-trace-id)	44
3.1.1	Disseny del component	44
3.1.2	Escenaris i algorismes	45
3.1.3	Altres vistes i informació adicional.....	46
3.2	Component (<i>Pre-Request</i>) - Validació CORS (ctti-request-cors)	46
3.2.1	Disseny del component	46
3.2.2	Escenaris i algorismes	48
3.2.3	Altres vistes i informació adicional.....	48
3.3	Component (<i>Post-Response</i>) - Gestió de capçaleres de seguretat (ctti-response-headers-secure)	49
3.3.1	Disseny del component	49
3.3.2	Escenaris i algorismes	51
3.3.3	Altres vistes i informació adicional.....	52
4.	ANNEXOS	52
4.1	<i>Procediment per importar polítiques a nivell de LTE</i>	52
4.2	<i>Procediment per desplegar polítiques i extensions sobre la plataforma</i>	53
4.3	<i>Errors custom definits en la política de gestió d' errors</i>	56

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

1. INTRODUCCIÓ

En aquest document, s' exposen els dissenys tècnics proposats per a les polítiques i extensions, concebuts com a millores dins de l' API Manager de CTTI. Aquesta proposta senta les bases per a un desenvolupament eficient i sostenible en proporcionar una guia detallada per a la implementació de polítiques i extensions a la plataforma de l'API Manager de CTTI.

1.1 Propòsit

En resposta a la necessitat d' enfortir i optimitzar l' entorn de l' API Manager de CTTI, s' ha generat aquest document que presenta els dissenys tècnics proposats per a les polítiques i extensions.

1.2 Abast

Abast del Document:

Aquest document se centra en la millora i optimització de l' API Manager de CTTI, abordant específicament els aspectes següents:

Polítiques:

- Disseny detallat de polítiques customitzades que possibiliti a CTTI de disposar de controls i noves funcionalitats a oferir als desenvolupadors d' APIs a la plataforma.
- Mecanismes per a la importació de polítiques que serveixi de manual per a us despleguis d'aquestes per a l'Oficina Tècnica.

Extensions:

- Definició detallada de l' arquitectura d' extensions que possibiliti a CTTI de disposar de controls i noves funcionalitats globals sobre les execucions d' APIs a la plataforma.

Aplicació del Document al 100%:

1. Nous Desenvolupaments:

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

2. Integració de Funcionalitats:

En incorporar noves funcionalitats o mòduls a l' API Manager de CTTI, el document servirà com a guia integral per assegurar una integració efectiva.

3. Actualitzacions de Polítiques:


Per adaptar i actualitzar les polítiques del sistema, aquest document proporcionarà les directrius necessàries, assegurant una gestió dinàmica i eficient.

Casos que Requereixen Adaptacions:

1. Canvis en Requisits de les polítiques personalitzades i extensions:

Si hi ha canvis substancials en els requisits durant el desenvolupament, el document podria requerir ajustaments per garantir l' alineació amb les noves necessitats.

2. Actualitzacions Tecnològiques:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

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.

1.3 Definicions, acrònims i abreviatures


Elements configurables pel desenvolupador que poden ser utilitzats per controlar el comportament de la política específica on s'han definit aquestes propietats. Els camps d'aquestes propietats són definits pel propi desenvolupador i els valors s'introdueixen a través de la interfície de la política dins l'assemlatge.

Acrònims i Abreviatures	Significat
API	És una interfície de programació d'aplicacions (API) que facilita la comunicació entre un usuari i un sistema o servei, canalitzant les sol·licituds de l'usuari cap a aplicacions que poden estar dins o fora de la xarxa interna del client, i transmetent les respostes corresponents de tornada a l'usuari.
Context	L'objecte context s'utilitza per accedir i manipular variables en el context API, el payload de missatges HTTP i l'encapçalat del missatge HTTP durant el processament de l'assemlatge a API Gateway.
Apim	L'objecte apim és un mòdul que comparteix la mateixa funcionalitat de l'objecte Context, amb la diferència que l'ús d'aquesta llibreria s'aconsella reduir-lo únicament per migrar APIs amb versió DataPower Gateway v5.
GatewayScript	Es tracta d'una política que es fa servir per executar un programa de "GatewayScript" de DataPower. El "GatewayScript" és un llenguatge de scripting utilitzat per personalitzar i manipular missatges HTTP, XML i altres tipus de missatges que passen pel Gateway.
DataPowers	Terme que s'utilitza dins de l'IBM API Connect, per designar o nomenar els components Gateways.
Propietats de l'API (Properties)	Elements configurables pel desenvolupador que poden ser utilitzats per controlar el comportament de les polítiques d'API Connect. Dins de les propietats es poden configurar el nom de propietat, el valor i, opcionalment, un catàleg específic al qual s'aplica un valor de propietat.
Propietats de les polítiques de l'API (Policy-properties)	Elements configurables pel desenvolupador que poden ser utilitzats per controlar el comportament de la política específica on s'han definit aquestes propietats. Els camps d'aquestes propietats són definits pel propi desenvolupador i els valors s'introdueixen a través de la interfície de la política dins de l'assemlatge.

1.4 Referències

Aquest punt no aplicaria en la definició i disseny de les polítiques i extensions custom abordat en aquest document.

2. DISSENY DETALLAT (POLÍTIQUES)

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

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 passarel·la 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 depenent de la funcionalitat de cada política. A diferència de les polítiques estàndard, que solen ser genèriques i aplicables a una àmplia varietat de situacions, les polítiques **USER DEFINED** ens brinden la capacitat d'emmotllar l'entorn tecnològic segons les necessitats particulars de la nostra arquitectura i projecte.

Les polítiques customitzades d'aquest tipus s'empren comunament amb el propòsit de proporcionar la flexibilitat necessària per adequar els paràmetres de seguretat, autorització i altres aspectes crítics a les característiques particulars del nostre projecte. A més, permeten un major control sobre l'accés a recursos i la configuració del sistema, assegurant que la infraestructura compleixi amb estàndards i requisits específics. De manera similar, possibiliten l'adaptació precisa de cada aspecte de l'entorn tecnològic a les necessitats individuals de cada component, aconseguint així una optimització efectiva del rendiment i la seguretat.

2.1 Component Validar IP Origen (ctti-validate-ip) (no disponible de moment)

2.1.1 Disseny del component

Es tracta d'una política que permet 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 reforça 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.

En el disseny de l'API tindrem una propietat amb el llistat d'IPs per entorn que es validaran amb la IP d'origen. Si aquesta IP d'origen no està en la propietat de l'Api, es mostrarà un missatge d'error. Aquest missatge d'error llançarà un **Throw** que permetrà controlar l'error al **Catch** de l'acoblament.

La política de validació d'IP recollirà com a dades d'entrada les propietats següents:

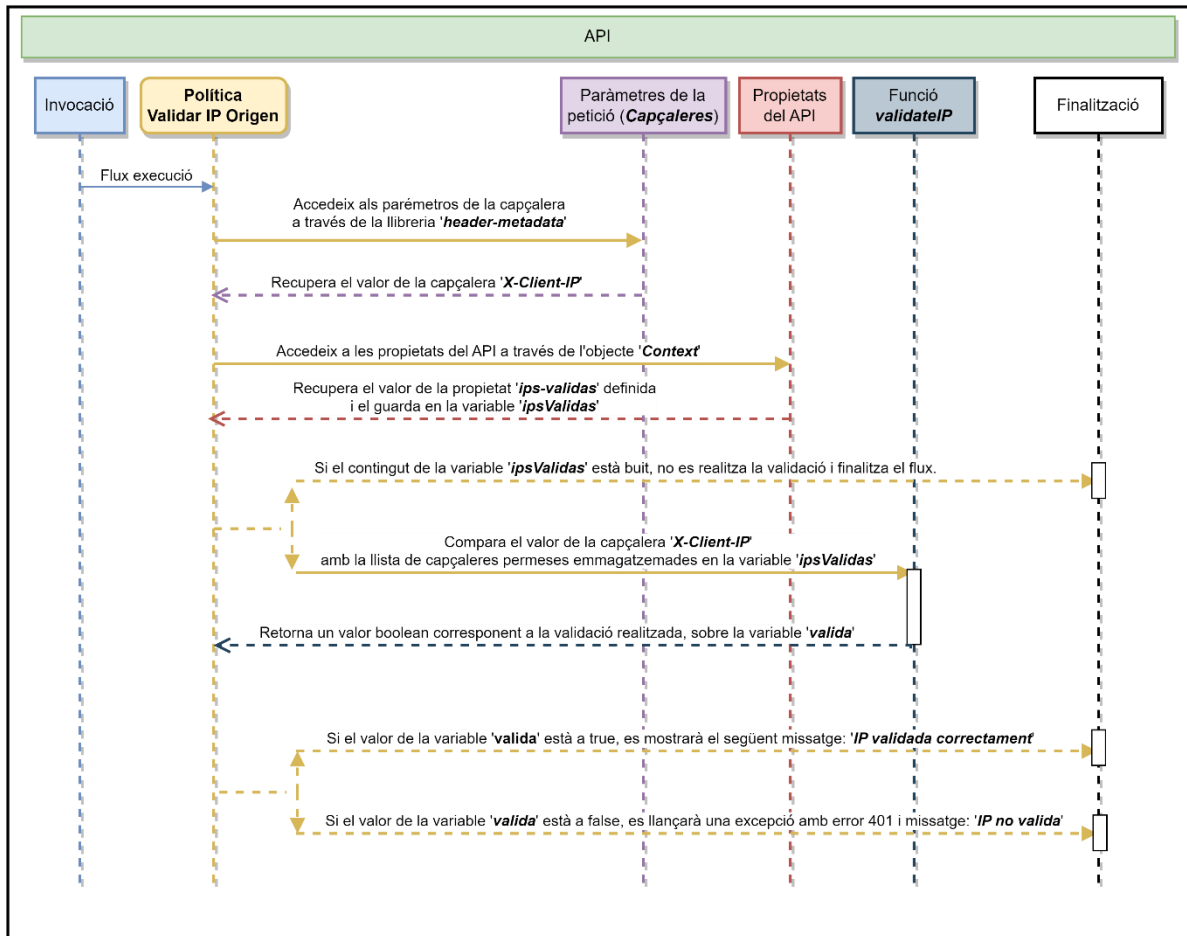
- **ipsValidas**: És una llista de les IPs permeses. Aquesta llista es definirà en una propietat de la pròpia API anomenada **ips-validas**. Serà el desenvolupador qui empleni aquesta propietat amb les IPs que s'han establert com a vàlides usant com a separador una coma. **Ex.:** 172.25.36.34,89.15.34.87,125.36.75.99. En cas que aquesta propietat no estigui informada, es retornarà un error indicant en el camp **moreInformation** que la propietat no està informada.
- **ipOrigen**: Direcció IP del client que ve en la variable de context (**request.headers.x-client-ip**).

La política seguirà els següents passos per validar la IP:

1. S'importa la biblioteca necessària **'header.metadata'** per a l'execució de la política.
2. Obtenir l'adreça IP del client de la capçalera X-Client-IP de la petició mitjançant l'objecte **'Context'** i assignar-lo a la variable **ipOrigen**.
3. Validar que **ipOrigen** és una IP permesa mitjançant la funció **validateIP (ipOrigen, ipsValidas)**. Aquesta funció, ens retornarà un **true** o un **false** de manera que:
 - **True**: La IP és vàlida es permet continuar el flux de l'API.
 - **La IP no és vàlida**, es genera un error amb els valors **httpCode 401** i **httpMessage Unauthorized** perquè sigui tractat en la política de **Gestió d' Errors** interrompent l'execució de l'API. La funció **validateIP (ipOrigen, ipsValidas)** valida si **ipOrigen** es troba a la llista d'IPs permeses (**ipsValidas**) i retorna **true** o **false** en funció de si la IP és vàlida o no.

2.1.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política per realitzar la validació de la IP d'origen:



Aquest document s'ha basat en la plantilla publicada al MQS Disseny Detallat V1.2

2.1.3 Altres vistes i informació addicional

En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:

- Si la propietat **"ips-validas"** està definida però no se li assigna cap IP com a valor dins de la propietat, aleshores es donarà per fet que no hi ha restricció quant a les IPs d'origen que realitzen les peticions.


2.2 Component Recuperació de variables (ctti-get-variables)

2.2.1 Disseny del component

L'objectiu d'aquesta política és recuperar el valor de variables emmagatzemades en diferents arxius a DataPower (Gateways). Això permet 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 és visible al YAML de l'API, sinó que es recupera 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:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```

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, caldrà 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.
- **fichero-variables:** Contindrà la ruta del fitxer sobre el qual buscar les variables. L'estructura del seu valor és la següent: **espai/nombrefichero.js** (**Ex:** *cd3292/fichero1.js*)

El disseny de la política de ctti-get-variables no contempla dades d' entrada com a tal, sinó que agafarà els paràmetres de les propietats de l' API anteriorment descrites.

Es definiran en el codi les variables internes següents:

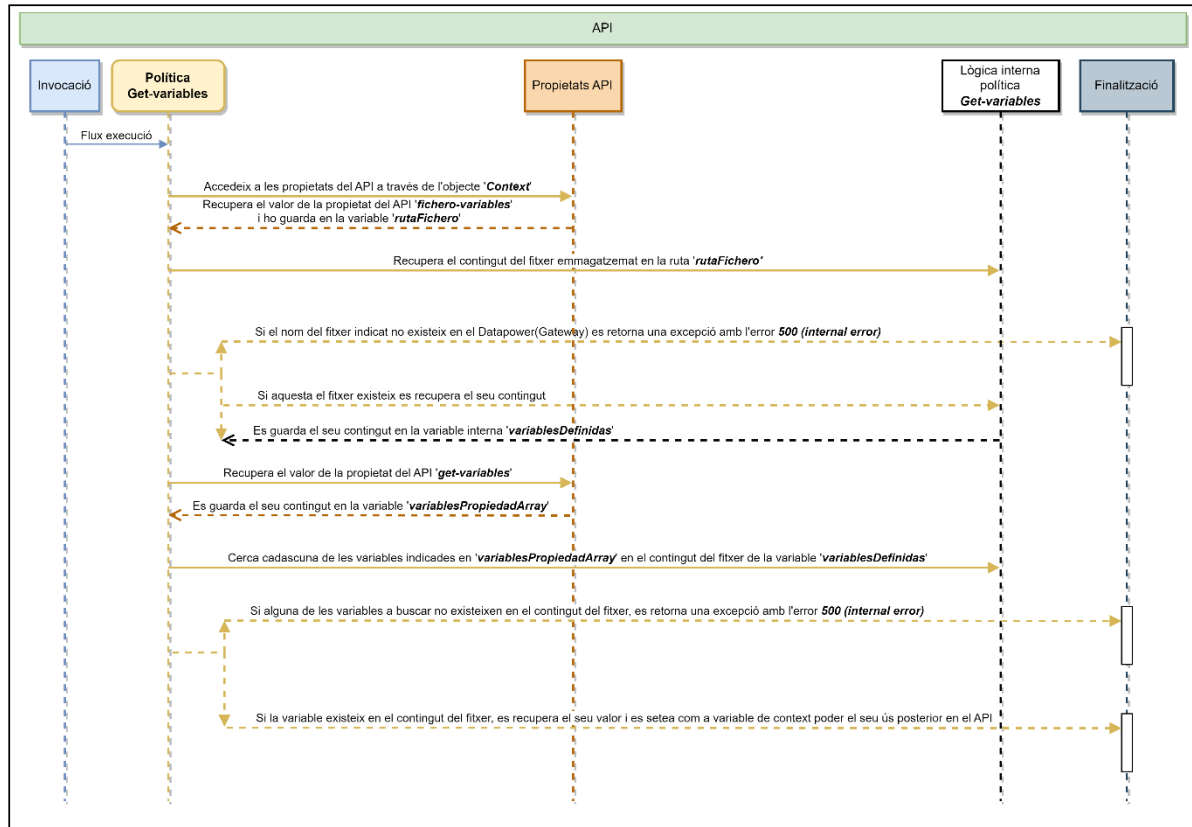
- **rutaFichero:** És el nom del fitxer que es vol consultar al DataPower. Es recollirà mitjançant una propietat de l'API (**'fichero-variables'**).
- **variablesPropiedad:** Aquí s'indicaran les variables que es volen recuperar mitjançant una propietat (**get-variables**), que es definirà a l'API. Sera el desenvolupador qui empleni aquesta propietat amb les variables que vulgui recuperar del fitxer del Datapower. Cadascuna de les variables que vulgui recuperar hauran d'anar separades per comes i sense espais (**Exemple:** *url1, usuari1, password2*). A més, hauran de tenir el mateix nom que tenen al fitxer per poder recuperar-les. En cas que aquesta propietat no està informada, es retornarà un error indicant que la propietat no està informada.
- **variablesDefinidas:** Aquí es carregarà el contingut del fitxer ubicat en **'rutaFichero'** que es troba a la carpeta **local** del Datapower.

La política seguirà els següents passos:

1. S'accedeix a les propietats de l'API utilitzant la llibreria de context **"Context"**.
2. Es recupera el contingut de la propietat **"fichero-variables"** de l'API, i el guardem en la variable **"rutaFichero"**, per posteriorment obtenir la ruta del fitxer a consultar.
3. A continuació, s'obté el contingut del fitxer, la ruta del qual està en la variable **"rutaFichero"** i el guardem en la variable interna **"variablesDefinidas"**. En el cas que s'hagi informat malament el nom del fitxer i no existeixi en el Datapower (Gateway), es mostrarà el missatge d'error **500 internal Server Error**, i en el camp **"moreInformation"** un missatge indicant que el fitxer amb aquest nom no existeix en el Datapower".
4. Es recuperen les dades de la propietat **"get-variables"** definida a l'API, separant les variables a recuperar i guardant-les en la variable **"variablesPropiedadArray"**.
5. Busca cadascuna de les variables recuperades en el pas 4 en la informació recuperada del fitxer en el pas 3. En el cas que no trobi alguna de les variables indicades, es mostrés el missatge d'error **500 internal Server Error**, i al camp **"moreInformation"** un missatge indicant que la variable no es troba al fitxer. En cas contrari, si la variable existeix, es recupera el valor i se setea com a variable de context de l'API a través de l'objecte **"context"**, per posteriorment poder accedir-hi si és necessari.

2.2.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política per recuperar les variables necessàries d'un fitxer ubicat als Datapowers (Gateways):



Aquest document s'ha basat en la plantilla publicada al MCS Disseny Detallat V1.2

2.2.3 Altres vistes i informació adicional

En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:


- En els passos descrits en el punt anterior es realitza la consulta del fitxer de variables retornant el valor de les variables demanades. Si el valor introduït en la propietat no està en el fitxer de variables, es generarà un error perquè sigui tractat en la política de Gestió d'Errors interrompent l'execució de l'API.

2.3 Component Capçalera Arquitectura (ctti-header-arch)

2.3.1 Disseny del component

Es defineix una capçalera d'arquitectura per a 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 l'eficiència de les comunicacions.

Cada capçalera tindrà un propòsit específic i contribueix a la funcionalitat i seguretat de les aplicacions. Un cas d'ús podria ser en aquells casos en què, des de l'API, es facin trucades als microserveis. La política emplenarà els camps de capçalera d'arquitectura necessaris per a una correcta comunicació amb els microserveis, evitant que el desenvolupador ho hagi de fer en cadascuna de les APIs que hagin d'utilitzar aquesta política.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

A través d'aquesta política, es disposarà d'una capçalera custom amb nous camps que s'inclouran en totes les peticions. Això 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'autenticació per a la seva validació en nivells inferiors del sistema.

Les capçaleres d'arquitectura a implementar en la solució seran les següents*:

- **Trace-Id**: Identificador únic de la petició. Es manarà el valor de la capçalera de **X-Ctti-Traceld**.
- **Authorization**: És el token d'autenticació contra els recursos específics del backend. L'agafarem del camp **Authorization** de la capçalera.
- **X-Forwarded-For**: Identifica l'adreça IP del client que va realitzar la sol.licitud. Es manarà el valor del paràmetre de capçalera **X-Client-IP**.
- **Accept-Language**: Indica el llenguatge en el qual es vol obtenir el resultat. Per defecte, s'informarà amb el valor **"ca-ES"**.
- **User-Agent**: Client que està fent la sol.licitud. Es mana a la capçalera de la petició amb el mateix nom (**User-Agent**). Se li assignarà el valor de **"NOINFO"** en cas que no es vingui informat en la petició.


*** 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

Aquests són els passos que es realitzen en la política de Capçaleres d'Arquitectura:

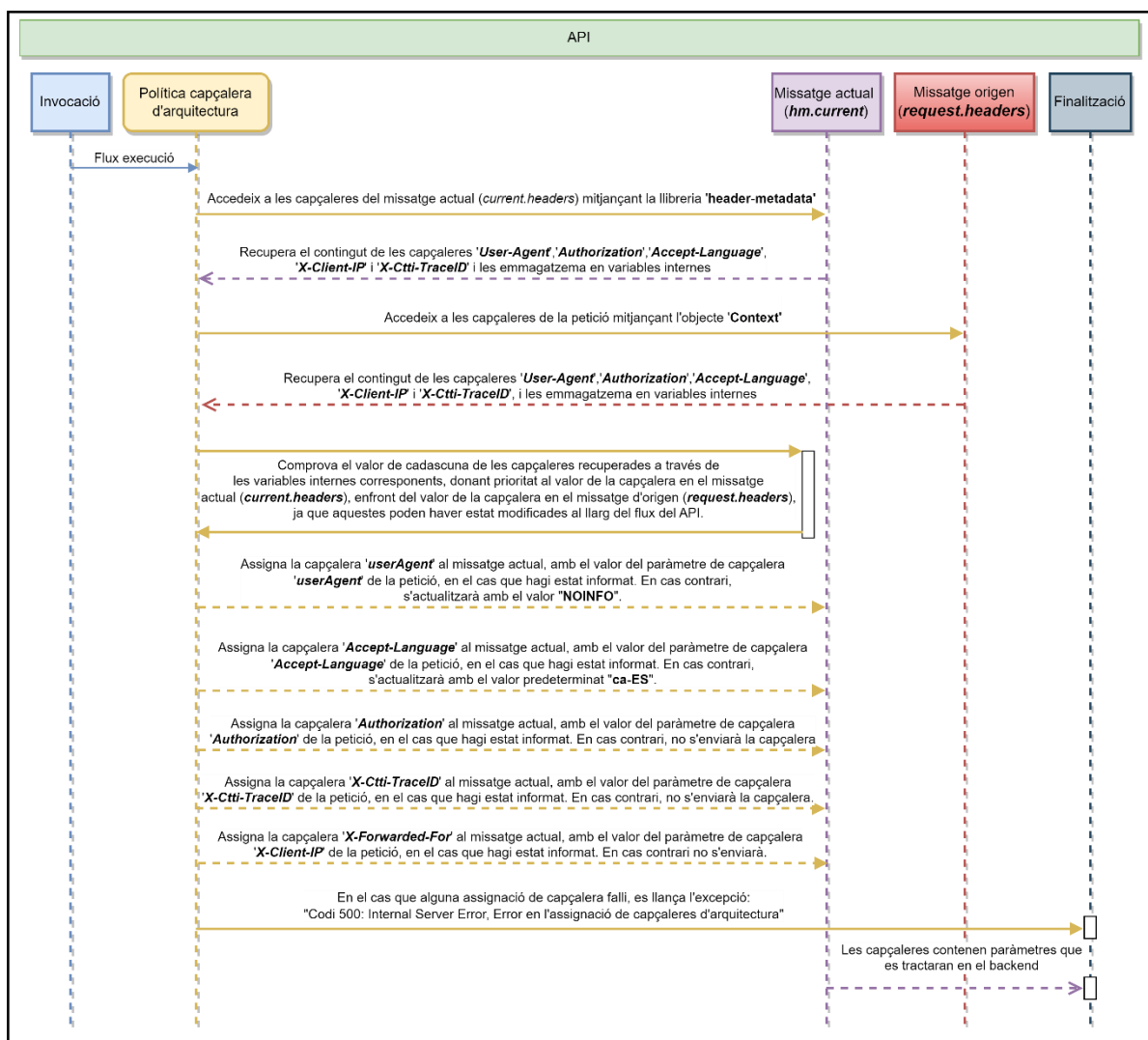
1. Accedeix al contingut de les capçaleres del missatge actual (**current.headers**) a través de la llibreria **"header-metadata"**.
2. Recupera el valor de les capçaleres actuals(current), **'User-Agent'**, **'Authorization'**, **'Accept-Language'**, **'X-Client-IP'** i **'X-Ctti-Traceld'** del missatge actual, i l'emmagatzema en variables internes.
3. S'accedeix a les capçaleres origen de la petició (**request.headers**) mitjançant l'objecte **"Context"**.
4. Recupera el contingut de les capçaleres origen de petició(**request.headers**), **'User-Agent'**, **'Authorization'**, **'Accept-Language'**, **'X-Client-IP'**, **'X-Ctti-Traceld'**, i l'emmagatzema en variables internes.
5. Comprova el valor de cadascuna de les capçaleres recuperades a través de les variables internes corresponents, donant prioritat al valor de la capçalera en el missatge actual (**current.headers**), enfront del valor de la capçalera en el missatge d'origen (**request.headers**), ja que aquestes poden haver estat modificades al llarg del flux de l'API.
6. Assigna la capçalera **'userAgent'** al missatge actual, amb el valor del paràmetre de capçalera **'userAgent'** de la petició, en el cas que hagi sigut informat. En cas contrari, s'actualitzarà amb el valor **"NOINFO"**.
7. Assigna la capçalera **'Accept-Language'** al missatge actual, amb el valor del paràmetre de capçalera **'Accept-Language'** de la petició, en el cas que hagi sigut informat. En cas contrari, s'actualitzarà amb el valor predeterminat **"ca-ES"**.
8. Assigna la capçalera **'Authorization'** al missatge actual, amb el valor del paràmetre de capçalera **'Authorization'** de la petició, en el cas que hagi sigut informat. En cas contrari, no s'enviarà la capçalera.
9. Assigna la capçalera **'X-Ctti-Traceld'** al missatge actual, amb el valor del paràmetre de capçalera **'X-Ctti-Traceld'** de la petició, en el cas que hagi estat informat. En cas contrari, no s'enviarà la capçalera.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

10. Assigna la capçalera **'X-Forwarded-For'** al missatge actual, amb el valor del paràmetre de capçalera **'X-Cliënt-IP'** de la petició, en el cas que hagi estat informat. En cas contrari no s'enviarà.
11. En el cas que alguna assignació anterior falli, llavors llança l'excepció **"Codi 500: Internal Server Error, Error en la assignación de cabeceras de arquitectura"**
12. A les capçaleres del missatge actual hi ha continguts els paràmetres que es tractaran al backend.

2.3.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política que configuren els paràmetres necessaris per interactuar amb els microserveis de CTTI:




Aquest document s'ha basat en la plantilla publicada al MCS Disseny Detallat v1.2

2.3.3 Altres vistes i informació adicional

En fer ús d'aquesta política s'ha de tenir en compte les consideracions següents:

- 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 **'User-Agent'** en la petició, se li assignarà el valor predeterminat (**"NOINFO"**) a la capçalera d'arquitectura que s'envia al backend.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

- En el cas que no vingui informada la capçalera '**X-Client-IP**', no es tractarà aquesta capçalera 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.
- En el cas que no s'envii la capçalera '**X-Ctti-Traceld**' en la petició, s'autogenerarà en l'extensió **ctti-trace-id**.

2.4 Component Logs (ctti-custom-log)

2.4.1 Disseny del component

Aquesta política s'encarregarà de guardar el contingut d'una variable 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 dos paràmetres d'entrada:

- **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_variable)**.

```
message
Message del log.
```

```
$(nombre_variable)
```

- Combinació de text pla i el contingut d'una variable com, per exemple, "**La persona que va realitzar la prova és: \$(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.

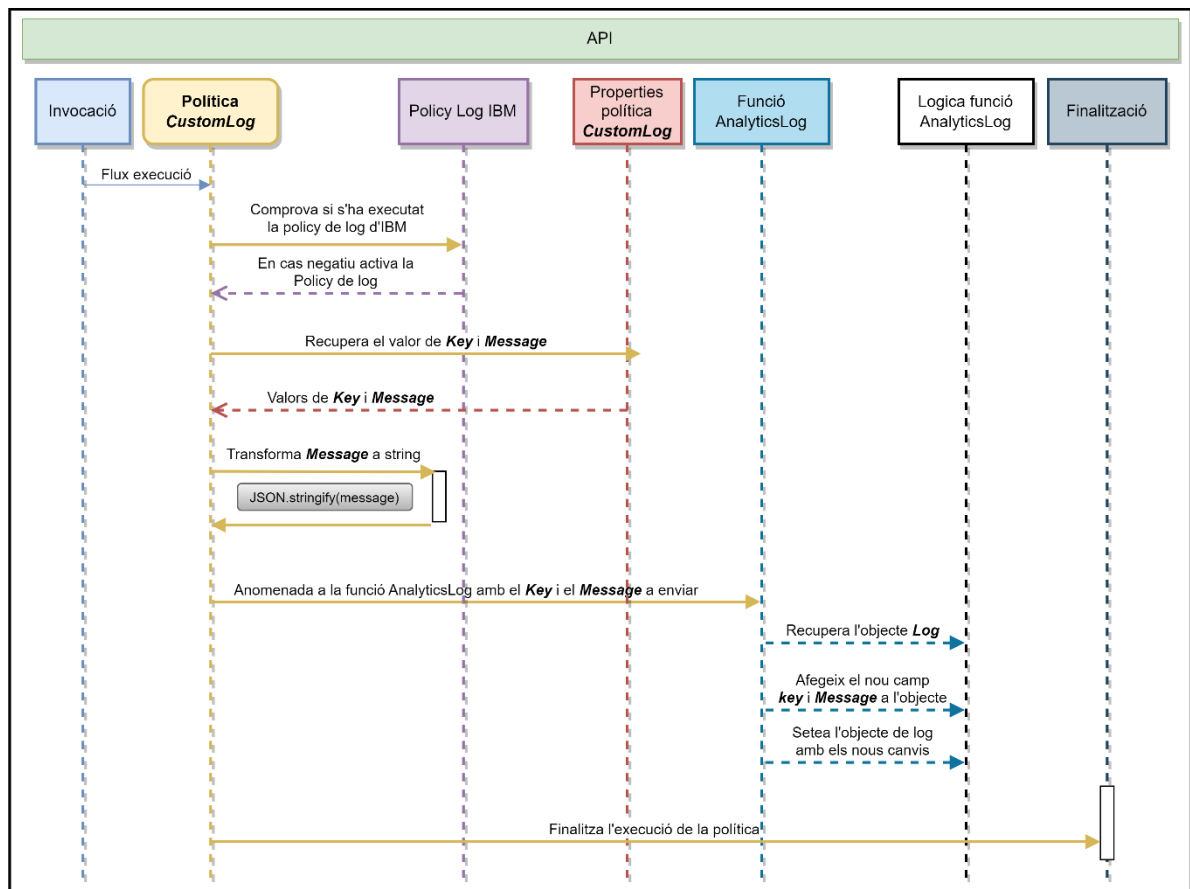
```
{"nombre":"Antonio", "apellido":"Sanchez"}
```

La política seguirà els següents passos per afegir els camps a l' objecte de log:

1. Comprova si s' ha executat la policy de log d' IBM, de no ser així l' executa.
2. Prendrà el valor de **key** i **message**.
3. En cas que **message** sigui un objecte el convertirà en un **String**.
4. Trucarà a la funció **analyticsLog** indicant-li la **key** i el **message** que es volen enviar.
5. La funció analyticsLog realitzarà la següent lògica:
 - a. Recuperarà de la variable de context l' objecte de log.
 - b. Introduirà en aquest objecte un nou camp, el nom del qual serà la clau i el valor del qual serà el missatge.
 - c. Setearà en la variable de log l' objecte amb el camp afegit.


2.4.2 Escenaris i algorismes

A continuació, es mostra a través d' un diagrama els passos que seguirà la política per afegir els camps per tal de aconseguir:



2.4.3 Altres vistes i informació addicional

En fer ús d' aquesta política s' ha de tenir en compte la consideració següent:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

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

2.5 Component Logs Invoke (ctti-invoke-log)

2.5.1 Disseny del component

Amb aquesta política podrem 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 el backend estan funcionant correctament, agilitant la identificació i resolució d'errors, i ampliant la traçabilitat.

Per a això s'enviarà 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 següents:

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

Aquests són els passos que es realitzen en la política d'invoke-log:

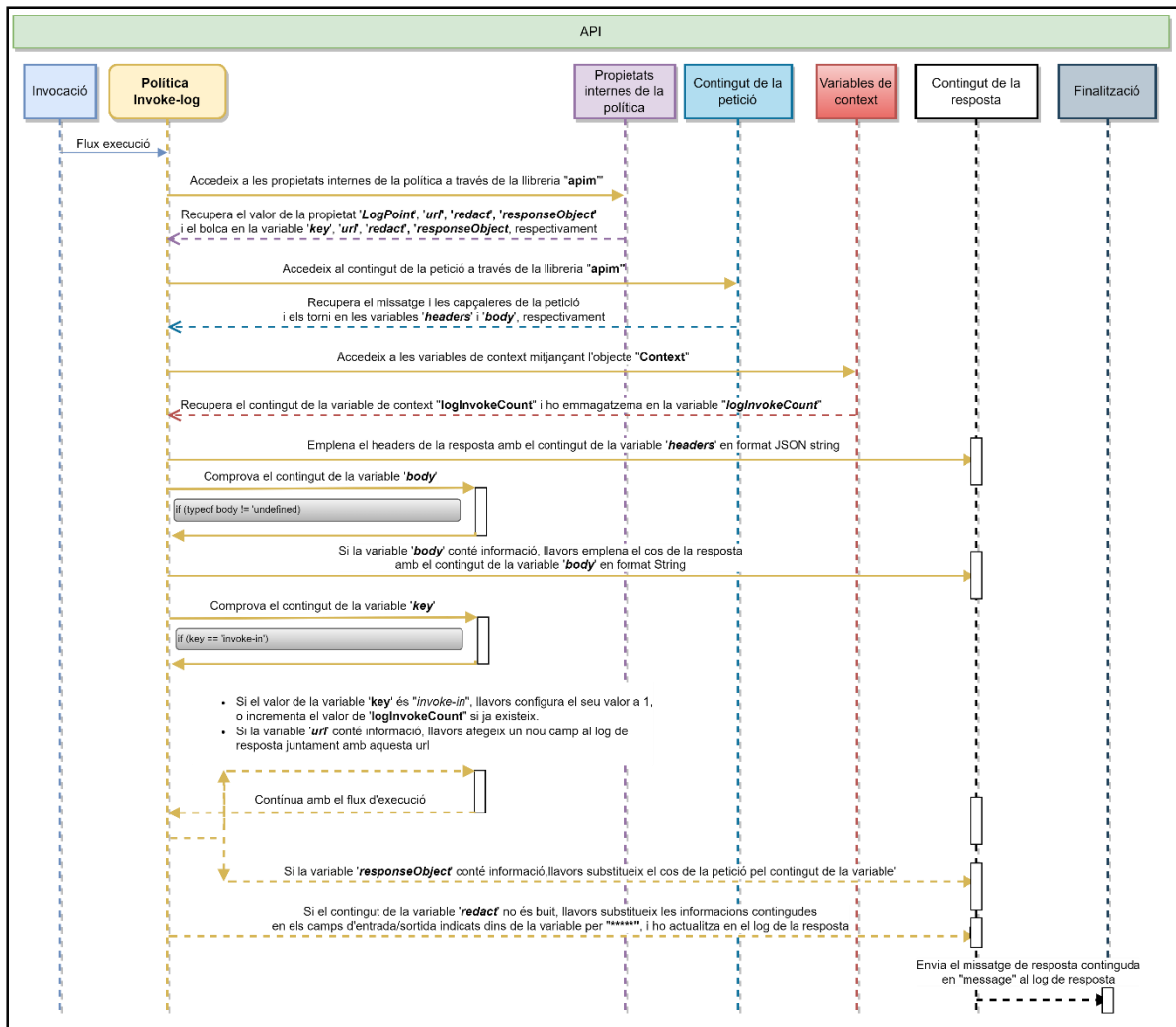
1. Primer, comprova si està declarat la variable de context **"log"** a l'API; si no es troba definit, llavors ho declara, i en cas contrari, se saltarà aquest pas.
2. En el cas que la variable **"Activity Log"** estigui activat, es procedeix a accedir a les propietats internes de la política amb la llibreria "apim".
3. Guarda els valors de les propietats **"LogPoint", "url", "redact", "responseObject"** i els emmagatzema en les variables internes de la política per al seu posterior ús.
4. S'accedeix al cos i les capçaleres de la petició mitjançant la llibreria **"apim"**.
5. Guarda el contingut del cos i les capçaleres de la petició en les variables **"body"** i **"headers"**, respectivament.
6. S'accedeix a la propietat de l'API denominada **"logInvokeCount"** amb l'objecte **"Context"**, i emmagatzema el seu valor en una variable interna.
7. Comprova si el cos de la petició NO és buit, i en cas afirmatiu, omple el seu contingut en la variable **"message"** en format **String**.
8. Comprova si el valor de **"key"** és **'invoke-in'**, en cas que sí que ho sigui:
 - Comprova si el contingut de la propietat **'url'** està buit, en cas negatiu, omple el contingut en la variable del missatge a enviar, que es denominarà **"message.url"**.

- Comprova el contingut de la propietat **"logInvokeCount"**; si no està definit, l'inicialitza amb el valor 1; si ja està definit, incrementa el seu valor en 1.

- Comprova si el valor de **"responseObject"** NO és buit, i en cas afirmatiu, substitueix el contingut del **"message"** pel contingut que es trobi en el camp de la petició definit en **"responseObject"**.
- Comprova si el valor de **"redact"** NO és buit, i en cas afirmatiu, substitueix el contingut del camp especificat per **"*****"** i l'emmagatzema en **"message"**.
- Finalment, envia el contingut de la variable **"message"** al log d'Analytics, sota el nom **invoke-in-{N}** i **invoke-out-{N}**.

2.5.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política per afegir els camps addicionals al log d'Analytics:




Aquest document s'ha basat en la plantilla publicada al MQS Disseny Detallat V1.2

2.5.3 Altres vistes i informació addicional

En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:

- Els nous camps que s'afegeixen 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.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

2.6 Component Validació de request (ctti-validate-request)

2.6.1 Disseny del component

El disseny d'aquesta nova política està enfocada en què s'utilitzi sempre al principi de l'assembly de les APIs en què es vulgui validar la request. Aquesta política realitzarà la validació del missatge d'entrada de la invocació a l'API 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 les trucades entrants compleixin amb el format esperat (paràmetres, queries, etc.) definit al YAML. Això evita que peticions mal formades o amb contingut maliciós arribin als backends. A més, es garanteix la qualitat de les APIs en exigir als desenvolupadors que segueixin els estàndards i requisits establerts en el procés de desenvolupament.

Per validar l'entrada s'utilitzarà la definició d'objectes que s'inclouen en el YAML. És en aquesta definició on el desenvolupador haurà d'incloure totes aquelles validacions que s'hagin de realitzar.

Nota: Cal destacar que la política només realitzarà la validació sobre el cos (**body**) de la petició, si s'ha enviat en format **JSON**, altrament, la política només realitzarà validacions sobre els paràmetres (**parameters**) i les capçaleres (**Headers**) de la petició.

La política no comptarà amb cap paràmetre d'entrada.

Els elements que consultarà la política per veure les dades a validar estaran continguts dins l'etiqueta del YAML **x-customPaths** → **path** → **verb** → **request** que es troba dins de l'etiqueta **x-ibm-configuration**. A continuació, veiem un exemple de **x-customPaths**:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```

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 **x-customDefinitions** del YAML 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:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0


```

{
  "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"
        }
      ]
    }
  ]
}

```

Aquest document s'ha basat en la plantilla publicada al MQS Disseny Detallat v1.2

Podria ser validada amb la següent definició de YAML:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0


```

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', 'K0']
      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 poden 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 "**arrayDeObjectes**" (**\$ref: '#/x-ibm-configuration/x-customDefinitions/arrayDeObjectes**), la 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'ha d'incloure l'etiqueta "**additionalProperties**" amb el seu valor booleà corresponent dins de la definició referenciada.


 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

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, per als **headers**, **params** i **body**, seran les següents:

- **obligatorio**
 - Indica 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 ha de complir el camp. Al YAML s' ha 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 permet el camp
- **minLength**
 - Longitud **mínima** que permet el camp
- **length**

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

- 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**
 - Valida un **array d' objectes** i en l' esquema cal especificar que propietats té cada objecte

La lògica a aplicar per la política es realitzarà en una funció interna dins de la llibreria "**cttiValidateRequest**", la qual estarà allotjada en els Gateways de CTTI en la ruta "**local:/lib/libCttiValidaRequest.js**", ja que en haver-hi funcionalitats compartides amb la política de validació de response s'agruparà el codi en aquesta llibreria, per aprofitar la reutilització.


La funció interna rebrà el següent paràmetre d' entrada:

- **bodyPetición:** Body de la request, en cas d' existir. Si no hi ha body s' haurà d' enviar un **null**. El body l'agafarà del YAML de l'API com ja ho fa la política i es farà referència a la funció des del YAML.

La resta de les dades necessàries per a l' execució es recuperaran dins de la pròpia funció.

Per a la validació de la request seguirà la següent lògica:

1. S'importa la llibreria ubicada al Gateway "**libCttiValidaRequest.js**" a través del comando '**require**', i es guarda en una variable interna per al seu posterior ús.
2. Es recupera el '**body**' de la petició i guarda el seu valor en una variable interna, a través del comando '**request.body**'.
3. Es valida el contingut de la variable que conté el cos de la petició. Si no està informat, es para el procés de la política i es retorna un error **400 'Bad Request'**. En cas contrari, es reprèn l' execució.
4. A continuació, s'invoca a la funció inclosa en la llibreria '**libCttiValidaRequest.js**', que valida l'estructura i el contingut de la petició, per procedir amb la lògica de la política.
5. Recupera els elements del YAML a validar
 - a. Realitza la recerca per path i verb dins de **customPath**.
 - i. Si no troba els valors, no es realitza la validació.
 - ii. Si troba els valors, continua amb l' execució.
 1. Si hi ha paràmetres definits, els recupera.
 2. Si hi ha headers definits, els recupera.
 3. Si hi ha definició de body, la recupera de **x-customDefinitions**.
 - iii. Es van realitzant les validacions en l' ordre definit a continuació. Si es produeix un error de validació en algun punt no continua i escala l'error corresponent (**E0400**):
 1. Realitza la validació dels paràmetres recuperats "**params**" contra una funció específica inclosa a la llibreria "**libCttiValidaRequest.js**".
 2. Realitza la validació dels "**headers**" recuperats contra una funció específica inclosa a la llibreria "**libCttiValidaRequest.js**".
 3. Realitza la validació del "**body**" contra una funció específica inclosa a la llibreria "**libCttiValidaRequest.js**".

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

6. La funció de validació a la qual s'ha anat invocant per a cada element ("**params**", "**headers**" i "**body**") continguda a la llibreria "**libCttiValidaRequest.js**", realitzarà les següents operacions:
- a. Es valida el contingut de la petició contra l' esquema definit amb el procediment següent:
 - i. Primer, transforma el JSON d'entrada a una estructura de JSON lineal (**{"usuario": "A", "datosAcceso.clave": "miclave"...}**).
 - ii. Després, es realitzaran les següents validacions:
 1. Si l' objecte JSON transformat està buit, significa que porta camps amb valor **null**, i per tant llançarà un error.
 2. Si l' objecte a validar conté propietats addicionals, s' invoca una funció específica que comprova si l' objecte té altres propietats no contemplades en l' esquema; en cas afirmatiu, s' aixeca una excepció indicant l' error i la primera propietat incongruent detectada.
 3. Posteriorment es validaran les diferents propietats que s' hagin especificat en l' esquema per a cada determinat camp.

En el cas que l' algun dels objectes a validar de l' esquema no aquest informat en el body, es comprovarà si és obligatori, en cas afirmatiu es retornarà un error, indicant que el camp "x" és obligatori. En cas contrari es continuarà amb validació de la propietat del camp.

Si en realitzar la validació de les propietats del camp, d' entre les possibles descrites anteriorment, no es compleix, es retornarà un error d' acord amb la validació realitzada. En cas contrari, continuarà amb la resta de les validacions.

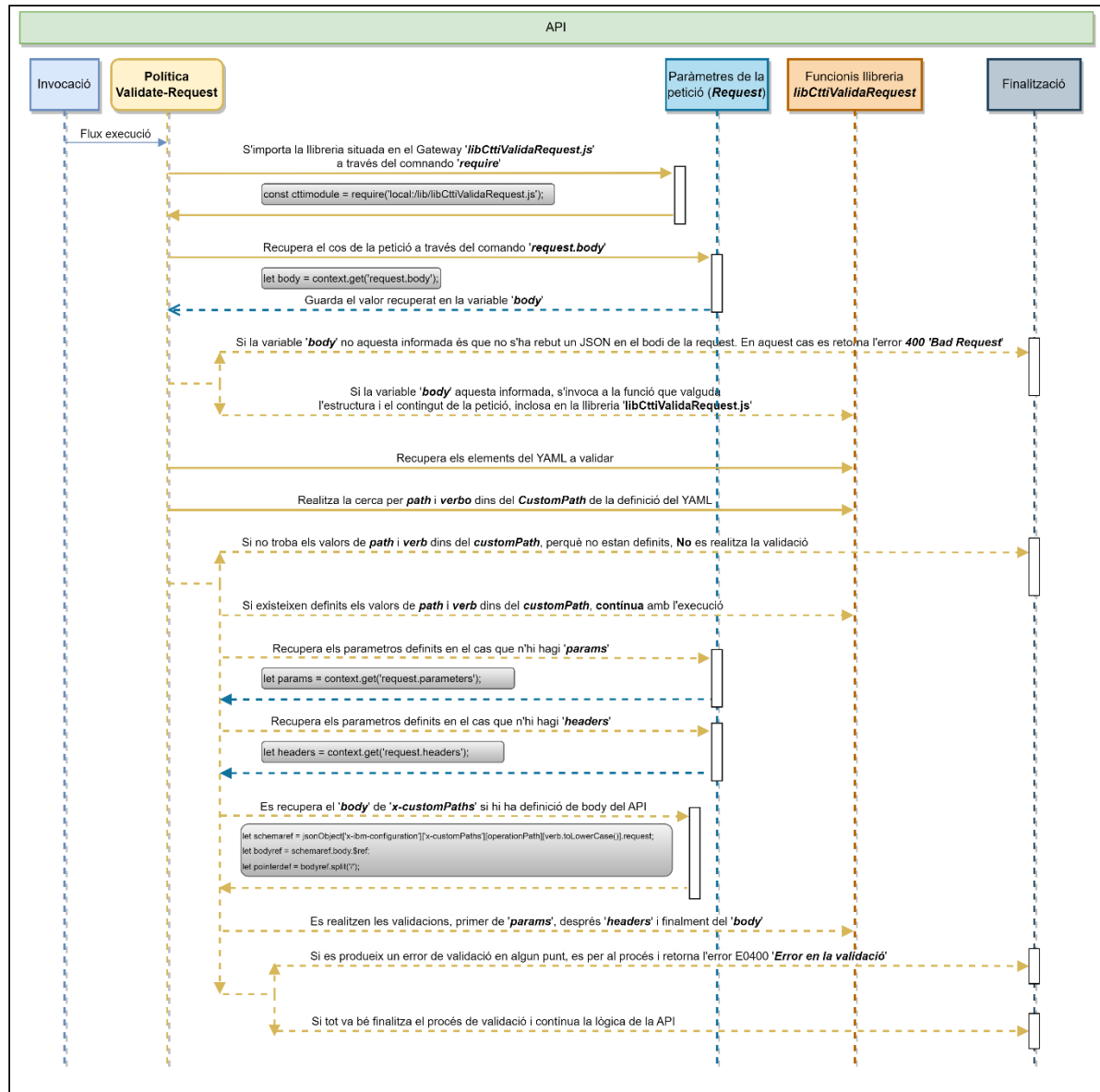
- b. Un cop s' han realitzat totes les validacions corresponents, es retornarà la següent estructura de dades:
 - **resultado**: Guardarà l'estat de la validació, **true** si ha estat tot OK i **false** si hi ha hagut un KO.

En el cas que es produeixi un error (KO) en algun punt de la lògica, es retornarà la següent estructura de dades, amb la informació corresponent:

- **nombreObjetoValidado** → l'objecte que s'ha validat
- **campo** → Nom del camp que s' ha validat, dins l' objecte
- **moreInformation** → Missatge d' Error
- **tipoError** → Tipus d'error generat, en aquest cas contindrà el valor "Controlled"
- **errorCode** → Codi d' error generat, en aquest cas contindrà el valor E0400
- **propiedad** → Propietat que ha estat validada

2.6.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política per realitzar les validacions de la request:




Aquest document s'ha basat en la plantilla publicada al MQS
Disseny Detallat v1.2

2.6.3 Altres vistes i informació adicional

En fer ús d'aquesta política s'ha de tenir en compte les consideracions següents:

- Si la petició llançada sobre l'API que està utilitzant aquesta política no conté **'body'**, es retornarà un error **400 'Bad request'**, ja que al fer ús d'aquesta política ha de venir informat en les peticions.
- Si no s'han especificat els elements a validar dins del YAML de l'API, segons l'especificat a dalt, no es realitzarà la validació corresponent, ja que, en no trobar la definició, encara que vingui el camp informat en la request, no es validarà.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

2.7 Component Validació de response (ctti-validate-response)

2.7.1 Disseny del component

El disseny d'aquesta política serà similar a la descrita anteriorment (*Validació de request*), però en aquest cas el disseny està enfocat que s'utilitzi sempre al final de l'assembly. Aquesta política realitzarà 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, querys, etc.) definit al YAML. Això assegura la qualitat de les APIs i dels backends. Per exemple, es pot detectar falsos positius que puguin retornar-se com a resposta per part del backend.

Per validar la resposta utilitzarem la definició d' objectes que s' inclouen en el YAML. És en aquesta definició on el desenvolupador haurà d' incloure totes aquelles validacions que s' hagin de realitzar.

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.

La política no comptarà amb cap paràmetre d'entrada.

Els elements que consultarà la política per veure les dades a validar estaran continguts dins l'etiqueta del YAML **x-customPaths** → **path** → **verb** → **response** → **httpCode** → **schema** → **\$ref** que es troba dins de l'etiqueta **x-ibm-configuration**. A continuació, veiem un exemple de com podria ser el contingut de **x-customPaths**:


 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
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'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 el que s'ha 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 seves 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:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```


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 poden definir objectes dins d'objectes a través d'una referència a la definició del mateix 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'ha 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:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
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 es permetran amb aquesta política per a cadascun dels camps que s'especifiquin en la definició del YAML, seran les següents:

- **obligatorio**
 - Indica 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 ha de complir el camp. Al YAML s' ha 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 permet el camp
- **minLength**
 - Longitud **mínima** que permet el camp..
- **length**
 - Longitud **exacta** que haurà de tenir el camp.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0


- **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**
 - Valida un **array d' objectes** i en l' esquema cal especificar que propietats té cada objecte.

La lògica a aplicar per la política es realitzarà en una funció interna dins de la llibreria "**libCttiValidaRequest.js**", la qual estarà allotjada en els Gateways de CTTI en la ruta "**local:/lib/libCttiValidaRequest.js**", ja que en haver-hi funcionalitats compartides amb la política de validació de request s'agruparà el codi en aquesta llibreria.

Per a la invocació a l' esmentada funció no seran requeriran paràmetres d' entrada, ja que les dades necessàries per a la seva execució es recuperaran dins de la pròpia funció.

Per a la validació de la response seguirà la següent lògica:

1. S'importa la llibreria ubicada al Gateway '**libCttiValidaRequest.js**' a través del comando '**require**', i es guarda en una variable interna per al seu posterior ús.
2. A continuació, s'invoca a la funció interna inclosa a la llibreria '**libCttiValidaRequest.js**', per procedir amb la lògica de la política.
3. Recupera els elements del YAML a validar
 1. Realitza la recerca per **path** i **verb** dins la definició de **x-customPaths**.
 - a. Si no troba els valors **No** es realitza la validació.
 - b. Si troba els valors continua l' execució.
 1. Si hi ha definició de body la recupera de **x-customDefinitions**.
4. Llança la validació del body contra la funció interna comuna amb la política de validació de request, que es troba a la llibreria "**libCttiValidaRequest.js**". Si es produeix un error de validació en algun punt No continua i escala l'error corresponent (**E0406**).
5. La funció de validació a la qual s'invoca en el punt anterior per validar el "body" de resposta, continguda a la llibreria "**libCttiValidaRequest.js**", és la mateixa que s'ha indicat en la lògica de la política validació de request, i realitzarà les següents operacions:
 - a. Es valida el contingut de la petició contra l' esquema definit amb el procediment següent:
 - i. Primer, transforma el JSON d'entrada a una estructura de JSON lineal (**{"usuario":"A", "datosAcceso.clave":"miclave"...}**).
 - ii. Després, es realitzaran les següents validacions:
 1. Si l' objecte JSON transformat està buit, significa que porta camps amb valor **null**, i per tant llançarà un error.
 2. Si l' objecte a validar conté propietats addicionals, s' invoca una funció específica que comprova si l' objecte té altres propietats no contemplades en l' esquema; en cas afirmatiu, s' aixeca una excepció indicant l' error i la primera propietat incongruent detectada.
 3. Posteriorment es validaran les diferents propietats que s' hagin especificat en l' esquema per a cada determinat camp.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

En el cas que l' algun dels objectes a validar de l' esquema no aquest informat en el body, es comprovarà si és obligatori, en cas afirmatiu es retornarà un error, indicant que el camp "x" és obligatori. En cas contrari es continuarà amb validació de la propietat del camp.

Si en realitzar la validació de les propietats del camp, d' entre les possibles descrites anteriorment, no es compleixen, es retornarà un error d' acord amb la validació realitzada. En cas contrari, continuarà amb la resta de les validacions.

- b. Un cop s' han realitzat totes les validacions corresponents, es retornarà la següent estructura de dades:
- **resultado:** Guardarà l'estat de la validació, **true** si ha estat tot OK i **false** si hi ha hagut un KO.

En el cas que es produeixi un error (KO) en algun punt de la lògica, es retornarà la següent estructura de dades, amb la informació corresponent:

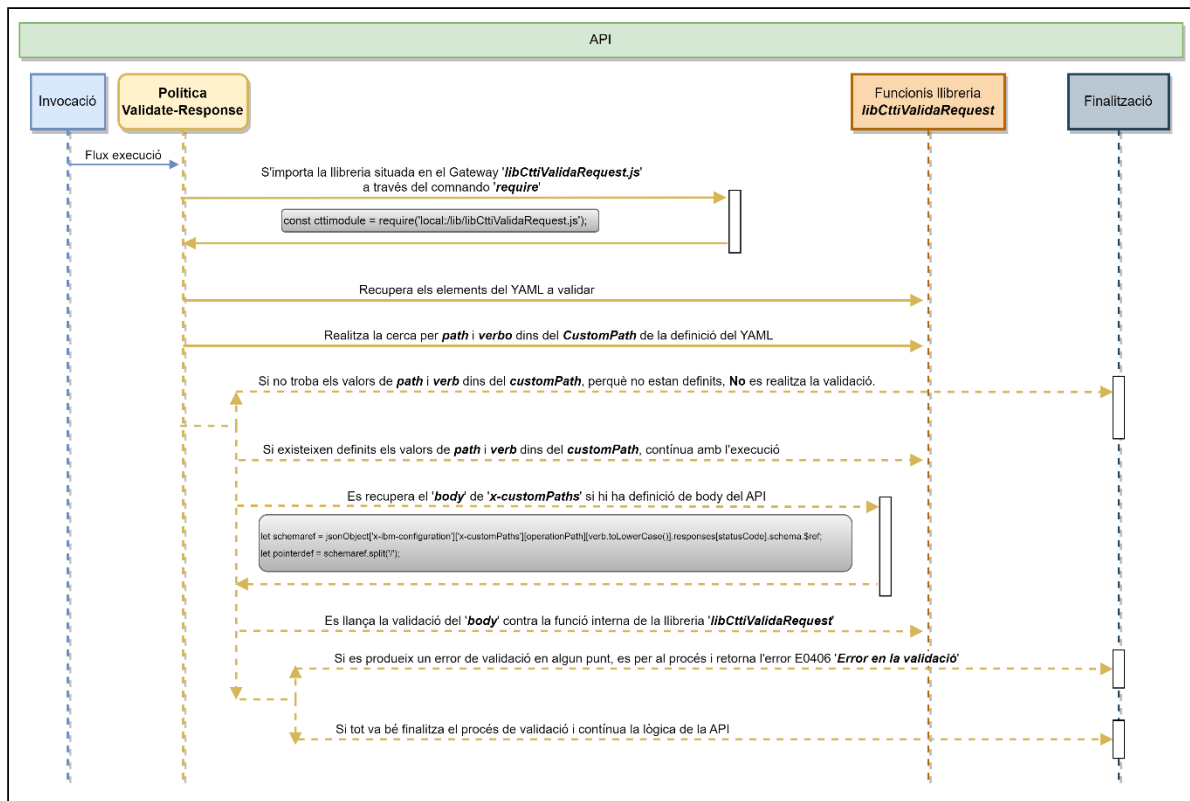
- **nombreObjetoValidado** → Nom de l'objecte que s'ha validat
- **campo** → Nom del camp que s' ha validat, dins l' objecte
- **moreInformation** → Missatge d' Error
- **tipoError** → Tipus d'error generat, en aquest cas contindrà el valor **"Controlled"**
- **errorCode** → Codi d' error generat, en aquest cas contindrà el valor **E0400**
- **propiedad** → Propietat que ha estat validada

La definició dins del mapa d'errors de l'error **"E0406"** ([vegeu el punt de l'annex 4.3](#)), seria la següent:

```
"E0406": {httpCode: 406, httpMessage: "Not Acceptable", moreInformation: "Error de validación de la respuesta del servicio de backend"}
```

2.7.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà la política per realitzar les validacions de la response:



Aquest document s'ha basat en la plantilla publicada al MQS Disseny Detallat V1.2

2.7.3 Altres vistes i informació adicional

En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:

- Si no s'han especificat els elements a validar dins del YAML de l'API, segons l'especificat a dalt, no es realitzarà la validació corresponent, ja que, en no trobar la definició, encara que vingui el camp informat en la request, no es validarà.

2.8 Component Gestió d'errors (ctti-error-management)


2.8.1 Disseny del component

El disseny d'aquesta política està enfocat a proporcionar una gestió més efectiva dels errors produïts dins l'assembly de l'API, de manera més customitzada, en funció de les necessitats del projecte. S'indica l'estructura del missatge d'error per defecte a retornar, com fer ús de la política d'errors, i la customització d'errors per a aquells canals que requereixen una estructura diferent.

Els errors retornats no han 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.

Com a contingut de l'error per defecte, es retornarà la següent informació:

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

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
{
  "statusCode": 400,
  "httpMessage": "Bad Request",
  "moreInformation": "El campo no tiene un formato válido"
}
```

Facilitant la creació del missatge d' error, la política generarà el body i assignarà el valor del codi d' estat de la resposta.

A continuació, s' ha subdividit la informació en diversos apartats, atès que, per aplicar aquesta política, s' han de conèixer i utilitzar components existents al Toolkit API Connect per al disseny de l' API, i per tant considerem que s' ha de conèixer el seu funcionament i com s' integren amb la política. La definició dels passos que realitzarà la política es troba a l' [apartat 2.8.1.2](#).

2.8.1.1 Escalat de l'error a GatewayScript

Perquè la política de gestió d'errors pugui reconèixer correctament els errors custom llançats des d'una política "**GatewayScript**", han de ser aixecats mitjançant la funció error (**context.reject**) de la llibreria context.

Per a fer ús d'aquesta funció se li han d'enviar totes les dades de l'error en particular que s'està aixecant, ja que és l'objecte que recuperarà la política d'error per a escriure la resposta.

```
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.
- **context.message.statusCode = "501 Custom Error";status code" i "status reason"**, es realitza 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, "**501 Custom Error**".

```
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";
```

Tots els codis d'error (status.code) que s'hagin d'escalar han d'estar definits com a resposta de l'operació al YAML.

Com s'ha citat anteriorment, per a l'escalat d'errors s'ha d'utilitzar l'objecte error que proporciona IBM (versió v10 del API Connect), on l'escalat de l'error no es realitza a través d'un throw exception, sinó que cal fer ús de la llibreria **context.reject()**.

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

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

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

Nota: En el cas que l'error a aixecar sigui un dels definits a la taula d'errors ([vegeu l'annex 4.3](#)), o inclòs a través de la propietat de la política de gestió d'errors "**erroresParticularesCtti**", 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 a través de l'errorCode. 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.

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:

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


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

Amb això la política comptarà amb la informació necessària per gestionar l' error:

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

La política posa a disposició una taula d'errors custom ja definits, en els quals es contempen els més comuns, i que poden ser referenciats a l'hora d'aixecar l'excepció ([vegeu el punt de l'annex 4.3](#)). Igualment, la política, a través de la propietat "**erroresParticularesCtti**", inclosa en la plantilla de desenvolupament d'APIs, et permet afegir errors a la taula d'errors.

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

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
{
  "ErrorCodeCustom" : {
    "httpCode": "<codigo 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:
  erroresParticularesCtti:
    value: >-
      EXAMPLE: {"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**
```

2.8.1.2 Política de gestió d' errors

Aquesta política ha de ser utilitzada en el catch de l' assembly de l' API.

La política de gestió d' errors 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.

Aquesta política realitzarà les següents accions:

- Captura de l' error.
- Recuperació de properties.
- Generació de l' objecte **estructuraErrorDevolver** en base a la definició en el YAML.
- Crida a una funció interna de la política per generar l'objecte d'error a retornar.
- Crida a una altra funció interna de la política per donar format a l'objecte d'error segons el format requerit en la resposta (**json/xml/soap**).
- Retornarà com a *status.code* i *status.reason* els valors escalats amb l' error i indicarà a la capçalera **ContentType** el format de missatge retornat.

Si es vol modificar el codi i estat de http de la resposta s' haurà de realitzar en un **GatewayScript** posterior a la política de gestió d' error.

2.8.1.2.1 Dades d' entrada a la política


Les dades d' entrada que requerirà la política de gestió d' errors seran les següents:

- **Objeto Error**
 - La política recuperarà l'objecte d'error escalat amb **context**.

- **Properties**

A nivell de propietats dins de l' API s' afegirà la següent:

- **erroresParticularesCtti**
 - Contindrà aquells errors que s' hauran d' afegir o modificar dins del mapa d' errors depenent de l' API. Un exemple seria el següent:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```

{"CTE0720": {"httpCode": 400, "httpMessage": "PARAMETER_NOT_CONSISTENT",
"moreInformation":
"El código del trámite indicado no se encuentra en el sistema."},
"CTE0672":
{"httpCode": 400, "httpMessage": "PARAMETER_NOT_CONSISTENT",
"moreInformation": "El
servicio de consulta no se encuentra disponible."}}

```

- **Entrada gestión errors**
 - En la pròpia política es disposarà un camp per indicar el format de la resposta. Els formats suportats són:
 - **json**
 - **xml**
 - **soap**

2.8.1.2.1.1 Definició de la política

La política de gestió d' error comptarà amb el següent camp d' entrada:

- **FormatoRespuesta**

En aquest camp s'ha d'indicar si la resposta ha d'estar escrita en "**xml**", "**json**" o "**soap**", enviant el que es requereixi.

La política de gestió d' errors seguirà els següents passos per arribar a generar el missatge d' error.

1- Captura de l' error:

La política recuperarà l'error que ha estat aixecat [**context.get('error')**] i extraurà els camps name(**nombreExcepcion**) i message(**detalleExcepcion**) per posteriorment ser utilitzats en funcions internes de la política.

2- Recuperació de propietats:


Es recuperarà l' **erroresParticularesCtti** i es prepararà l'objecte amb la informació per ser posteriorment utilitzat en funcions internes de la política.

3- Generació de l' objecte estructuraErrorDevolver:

L' objecte **estructuraErrorDevolver** es generarà a partir de l' esquema que estigui referenciat en el codi de resposta corresponent per a l' operació dins del YAML. Si no es troba la referència dins del YAML, es generarà l' estructura d' error per defecte.

S' obtindrà l' esquema de resposta seguint la lògica següent:

- Del context de l' API s' obtindrà el **path** i **verb** de l' operació.
- Aquesta estructura de **path** i **verb** haurà d'estar dins de l'etiqueta **YAML "x-customPaths"** dins de "**x-ibm-configuration**".
- En el YAML de l' API es buscarà la referència de la definició de l' error seguint la següent lògica. Si no es troba la referència es retornarà l' objecte d' error bàsic anteriorment generat. **Path** → **verb** → **responses** → **httpCode** → **schema** → **\$ref**.
- La definició de l'error s'haurà de trobar dins de l'etiqueta "**x-customDefinitions**" que està dins de "**x-ibm-configuration**". Un exemple seria el següent:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
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'
```

Aquest esquema de resposta necessita posseir un camp **key** dins d'aquells camps que tinguin un mapatge dins del **estructuraErrorDevolver**. És sobre la base d'aquest key que es generarà l'objecte

En aquesta política l'objecte d'error que s'elevàretornarà amb la informació definida, podrà tenir dues estructures depenent de si s'ha definit o no, un esquema de sortida d'error custom en la definició del YAML a tenir en compte, per incloure'l en l'objecte d'error.

L'estructura de l' objecte d' error en el cas que no s' hagi definit un esquema d' error de sortida custom a incloure' s, seria la següent:

- **statusCode:** Codi d' error httpcode que s' assignarà a la resposta com, per exemple: 400, 500, etc.
- **body:** Cos del missatge de resposta, que per defecte seguirà la següent estructura:


```
{
  "statusCode": "<código del error>",
  "httpMessage": "<razón del estado>",
  "MoreInformation": "<Descripción del error>"
}
```

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",
  "httpMessage": "Internal Server Error",
  "moreInformation": "Internal Error"
}
```

D'altra banda, en el cas que s'hagi definit un esquema d'error de sortida custom dins de "**x-customDefinitions**", l'objecte d'error que aquesta política elevaria en el body tindria una estructura d'acord amb el que s'hagi definit al YAML.

Per realitzar aquesta definició s' han de seguir els següents passos:


 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

1. S'ha d'indicar la **definició** de l'esquema d'error de sortida custom, dins l'etiqueta "**x-customDefinitions**" de la definició del YAML. Un exemple seria la següent (**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 de realitzar incloent en 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.

2. A continuació, es fa **referència** a l'esquema d'error de sortida custom definit anteriorment, dins de la definició del YAML, quedant de la següent forma (en aquest cas es referència com a esquema de sortida a utilitzar en la resposta a l'error 400 per al verb "**post**"):
 -

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
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'
```


- Un cop s'ha realitzat la definició de l'esquema d'error de sortida custom i configurada la seva referència dins la definició del YAML, ja podríem utilitzar-lo en la lògica de la nostra API. Per a això, en el **"GatewayScript"** corresponent, abans d'aixecar l'excepció, s'ha de definir una variable que contindrà un JSON amb els camps i valors que es desitgen mapejar per a cadascun dels **"key"** (camps), definits en l'esquema d'error de sortida, així com els valors bàsics que requereix la política (**"tipoError"**, **"errorCode"** i **"moreInformation"**).

Mostrant un exemple del comentat a dalt i continuant amb l'exemple anterior de l'esquema custom definit **"tppMessagesCustom"**, 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 mapegessin.

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**).

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

*Els camps que s' incloquin dins la variable error, que no corresponguin a cap dels camps definits en l' esquema, no es mapejaran en l' objecte de sortida com, per exemple: **campoExterno1**, **campoExterno2**.*

4. 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, quedaria de la següent manera:

```

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

```

D' aquesta manera, la política permet customitzar 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.

4- **Generació de l' objecte de resposta de l' error:**

S' anomenarà una funció interna de la política que realitzarà la lògica per generar l' objecte d' error.

A aquesta funció se li introduiran per paràmetre els objectes generats prèviament:

- **erroresParticularesCtti**
- **nombreExcepcion**
- **detalleExcepcion**
- **estructuraErrorDevolver**


A continuació, aquesta funció realitzarà les següents accions per generar l' objecte d' error:

- En el cas que s'hagin indicat errors particulars a tenir en compte (**erroresParticularesCtti**), a través d'una funció definida, s'afegiran en el mapatge d'errors genèrics de la política per fer-ne ús al llarg de la lògica.
- Es verificarà el nom de l'excepció elevada (**nombreExcepcion**), per determinar si es tracta d'un error per defecte, o es tracta d'un error custom de sortida definit al YAML i s'ha de realitzar el corresponent mapatge.

Per a això, en el cas que el nom de l'excepció sigui "**ConnectionError**", o "**BadRequestError**", es retornarà l'objecte de missatge de sortida estàndard (**httpCode**, **httpMessage** i **moreInformation**), amb la informació respectiva de cada error:

- **ConnectionError:**
 - **httpCode:** "500"
 - **httpMessage:** "Internal Server Error"
 - **moreInformation:** "Error de conexión con backend"
- **BadRequestError:**
 - **httpCode:** "400"
 - **httpMessage:** "Bad Request"
 - **moreInformation:** "Error de validación de los campos de entrada"

Per contra, si el nom de l'excepció és "**JavaScriptError**", es considerarà que és un error autoescalat, i per tant per generar l'objecte de resposta d'error, s'invocarà a una funció interna que verificarà, dins de l'estructura de "**errorMessage**" de l'excepció aixecada, si el camp "**tipoError**" té el valor "**controlled**". Si és així, es buscarà el codi

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

de l'error escalat "**errorCode**" dins del mapatge d'errors de la política per recuperar la seva informació, generant el següent objecte de resposta d'error:

- **httpCode**: httpCode indicat en el mapatge d' errors
- **httpMessage**: httpMessage indicat en el mapatge d' errors
- **moreInformation**: moreInformation indicat en el mapatge d' errors

En cas contrari, si NO ve informat el valor "**controlled**" en el camp "**tipoError**", o no s'ha trobat el codi d'error aixecat "**errorCode**" en el mapatge d'errors de la política, es retornarà l'estructura d'objecte de resposta d'error següent:

- **httpCode**: "500"
 - **httpMessage**: "Internal Server Error"
 - **moreInformation**: "Servicio no disponible momentáneamente"
- Si s'ha informat una estructura custom d'error en el paràmetre d'entrada "**estructuraErrorDevolver**", s'invocarà a una funció interna, la qual recorrerà cadascun dels camps indicats en l'estructura a retornar, i realitzarà el mapatge amb els seus respectius valors indicats en el camp "**errorMessage**" a l'hora d'elevat l'extensió, construint l'estructura de l'objecte de resposta d'error indicat. Un exemple seria l'indicat en el [punt 3](#) dins d'aquest mateix punt del document (**Generació de l'objecte estructuraErrorDevolver**)

finalment ens retornarà l' objecte que hem d' incloure en la resposta d' error en format JSON.

5- **Generació del missatge de resposta de l' error.**

S' invocarà a una funció interna de la política que realitzarà la lògica per generar el missatge de resposta de l' error.

A aquesta funció se li introduiran per paràmetre l'objecte de resposta del pas anterior i el format en el qual es desitja la resposta (**json/xml/soap**).


A continuació, aquesta funció validarà el tipus de format de sortida indicat, i en funció d' aquest es realitzarà la següent lògica:

- **json**: En ser el format origen de l' objecte d' error, no es realitzarà transformació i es retornarà tal qual.
- **xml**: Es realitzarà una conversió del contingut de l' objecte d' error, el qual està en format JSON, a XML.
- **soap**: En aquest cas es retornarà com a missatge una estructura SOAP custom predefinida tenint en compte les següents consideracions:
 - En aquest cas no es realitza una conversió del format JSON de l' objecte d' error a SOAP, ja que no procedeix, en el seu lloc es recollirà la informació necessària de l' objecte i es bolcarà en l' estructura SOAP.
 - L'estructura (**body**) definida per al format de resposta SOAP, seria la següent:
 - **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 el següent:

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

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

```
<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 vol seguir aquesta plantilla **SOAP** per necessitats del projecte, s'haurà de modificar a través d'una implementació en lloc d'usar la política.*

Per a qualsevol dels formats anteriors, també s'inclourà en la resposta el camp "**type**", corresponent a l'**applicationType** indicat ("**application/json**", "**application/xml**" i en el cas de "**soap**" ("**application/xml**").

En la resposta ens retornarà l' objecte que hem d' incloure en la resposta d' error en el seu format corresponent.

6- **Assignació de la resposta:**

Com a últim pas la política establirà els valors de la resposta:

- **Body de resposta:** S' introduirà l' objecte resultant del pas anterior.
- **Http status code:** Com a codi http s' introduirà l' elevat des de l' error en el camp code.
- **Http status reason:** Com a reason http s' introduirà l' elevat des de l' error en el camp status.reason.
- **Content Type:**
 - **JSON:** "application/json"
 - **XML:** "application/xml"
 - **SOAP:** "application/xml"

Si es requereix modificar algun d' aquests valors es farà mitjançant un GatewayScript a continuació de la política.

2.8.1.3 *Generació d' error customitzat mitjançant GatewayScript*


Per generar un missatge d'error diferent d'aquells que la política pot realitzar, s'haurà de crear un GatewayScript customitzat per mostrar un missatge a mida segons els requeriments.

Per recuperar el missatge d'error dins del GatewayScript customitzat utilitzarem les següents sentències:

```
let exception = context.get('error');
```

Amb això la variable excepció contindrà el JSON amb la informació de l' error. El format del missatge que conté és el següent:

```
{
  "name": "JavaScriptError",
  "message": "Internal Error",
  "status": {
    "code": 500,
    "reason": "Internal Server Error"
  },
  "policyTitle": "gatewayscript"
}
```

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

Aquest missatge conté la informació de l' error.

Hi ha casos en què no se'ns informarà del body que ha retornat el backend, com en el cas d'invoke. Per poder obtenir aquesta informació, es requereix que aquestes dades siguin mapejades en l' excepció.

Un exemple d'utilització del GatewayScript amb el tractament d'un error seria el següent:

```
const hm = require('header-metadata');
hm.current.set('Content-Type', 'application/json');
let exception = context.get('error');
context.set('message.status.code', exception.status.code);
context.set('message.status.reason', exception.status.reason);
let salidaError = {};
salidaError.httpCode = exception.status.code;
salidaError.httpMessage = exception.status.reason;
salidaError.errorCode = "E0401"
let customMessage = [{"tipoError": "controlled", "errorCode": "401", "moreinformation": "No está autorizado para ejecutar este servicio", "field": "API"}];
salidaError.moreInformation = customMessage;
session.output.write(salidaError);
```


Això ens donarà una sortida com la següent:

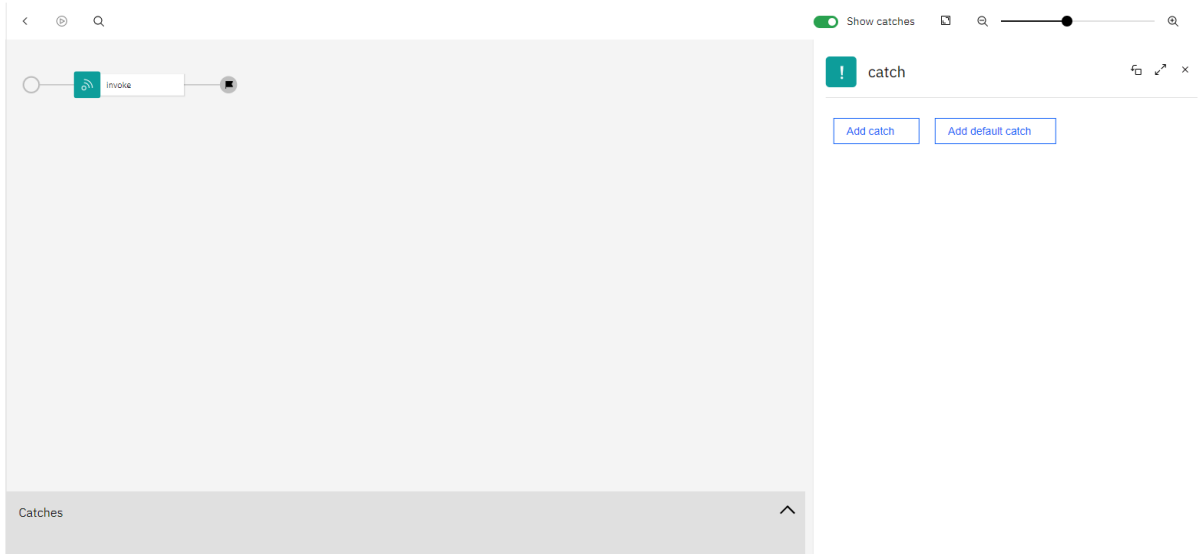
```
{
  "httpCode": 500,
  "httpMessage": "Service Error",
  "errorCode": "E0500",
  "moreInformation": [
    {
      "field": "API",
      "value": "Target url not reachable."
    }
  ]
}
```

2.8.1.4 Captura d' errors

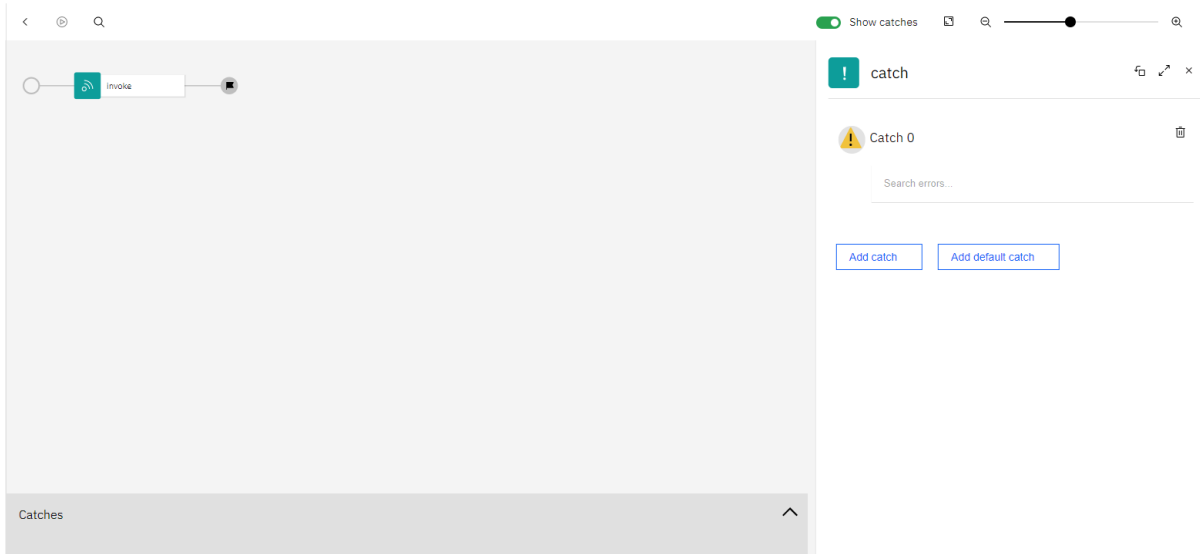
També conegut com a **Catch** de l' acoblat, s' utilitza per implementar els fluxos en la instància en la qual es produeix un error durant l' execució. Per exemple, l'acoblament podria contenir un component de llançament, la persona que crida l'API podria no autenticar-se o una política podria fallar en executar-se correctament. Cada error es pot manejar amb una captura diferent i cada captura pot manejar múltiples errors d' estat.

Per començar farem doble click esquerre sobre **"catch"** o **"captura"**. Es desplegarà un menú al lateral dret.


 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

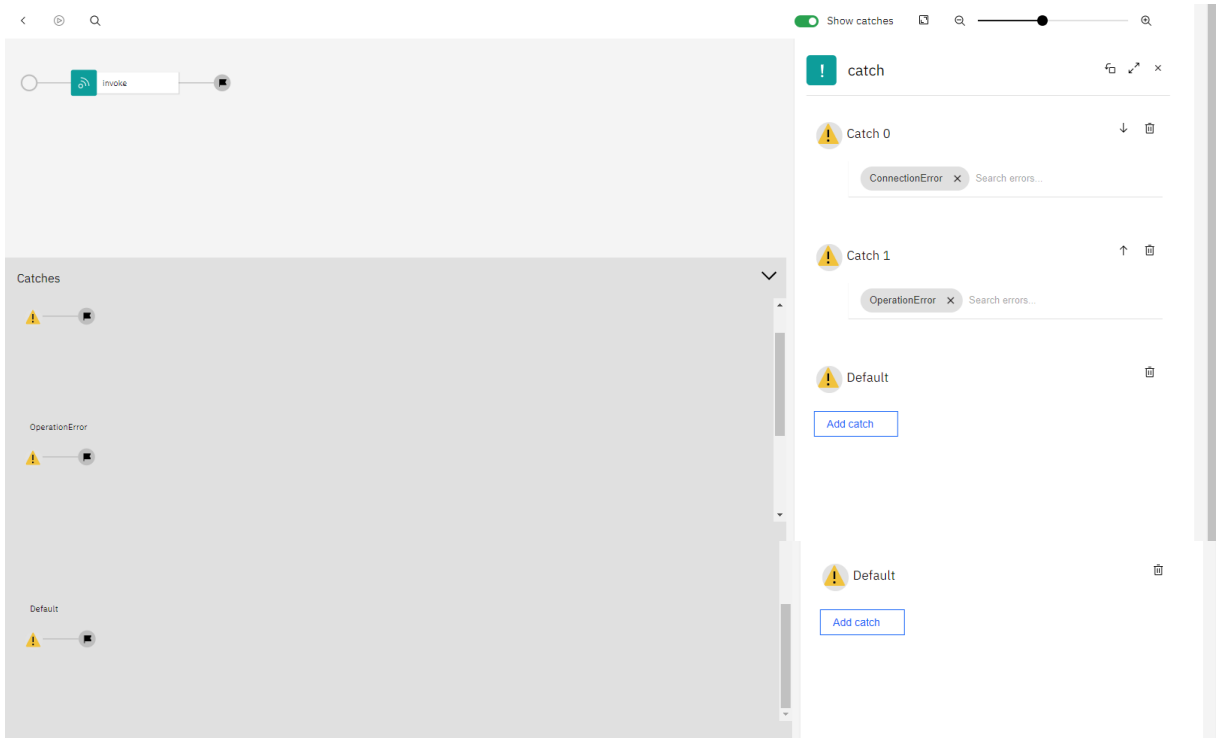


Dins d'aquest menú farem click a **"Add catch"** i hauríem de veure una cosa similar a la imatge:



Com podem observar tenim la possibilitat de crear tants catch com tipus d'errors volem contemplar. En la següent imatge veiem un catch creat amb els següents tipus d'error: **ConnectionError**, **OperationError** i **Default (Predeterminat)**.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0



Com a mínim en els catch s' hauran de capturar els errors de **ConnectionError**, **JavaScriptError**, **BadRequestError** i **Default**.

2.8.1.4.1 Casos d' error suportats pel catch d' acoblament

A continuació, s' indiquen els Catches per a errors produïts en l' acoblament.

- **JavaScriptError**
S'ha produït un error en executar JavaScript o GatewayScript en una política.
- **PropertyError**
S' ha produït un error a causa d' una propietat incorrecta durant una trucada d' invocació o durant l' execució d' una política set-variable quan una acció no era set, add o clear.
- **RedactionError**
S' ha produït un error durant la redacció d' un camp com a part d' una política de redacció.
- **TransformError**
S' ha produït un error durant una política de transformació.
- **RuntimeError**
S' ha produït un error no especificat.
- **ValidateError**
S' ha produït un error en intentar validar l' esquema d' un payload.
- **BadRequestError**
S'ha produït un error en intentar accedir a la sol·licitud. A partir de la versió 10, aquests errors no estan continguts en el catch **"Default"**; per capturar aquest tipus d' errors s' ha de generar aquest catch.
- **Default**
Qualsevol error que no es pugui classificar com un dels catches definits en l' acoblat es tractarà dins d' aquest catch com a error genèric.

2.8.1.4.2 Consideracions a tenir en compte

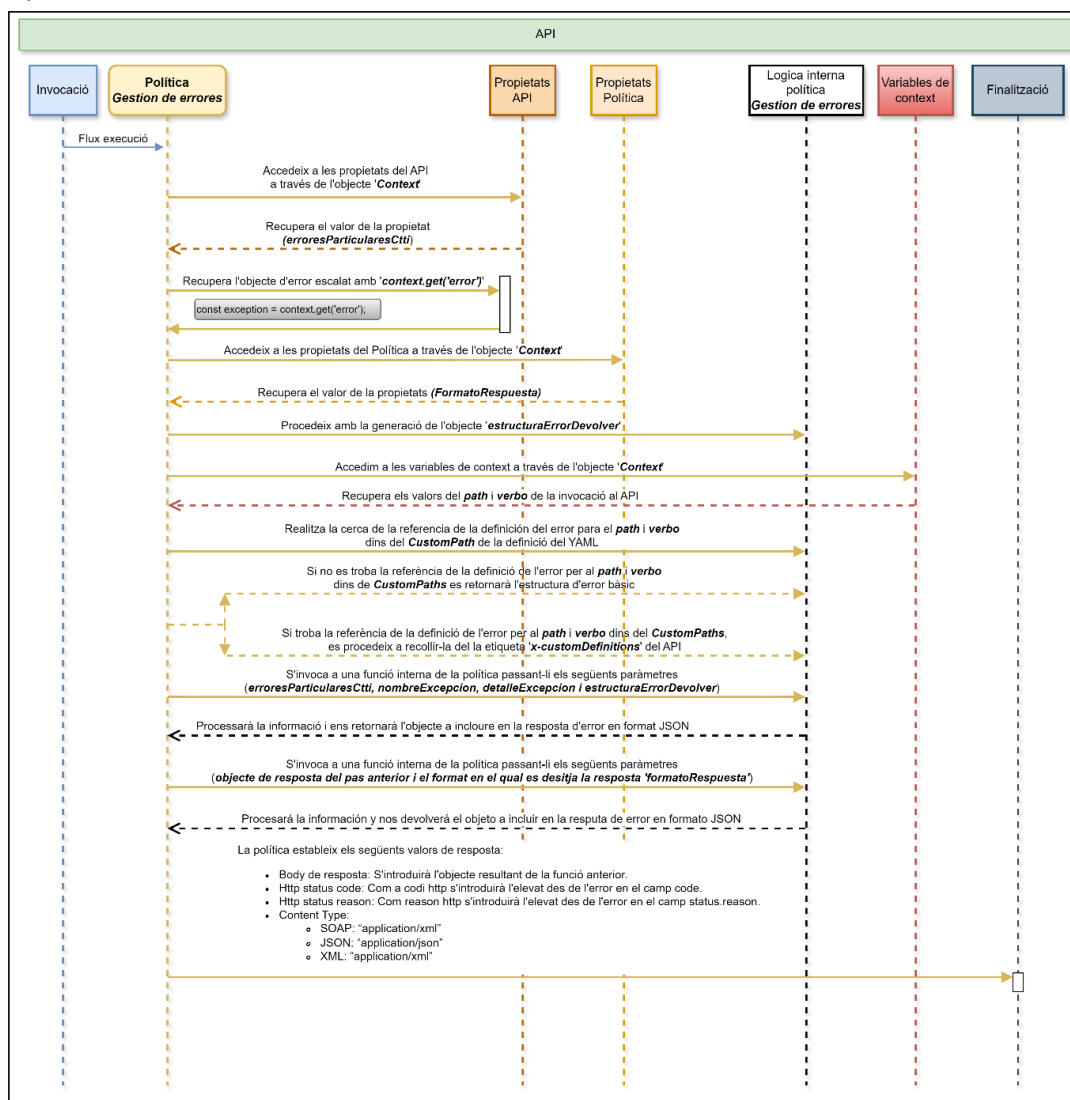
Si es vol modificar tant el body com el codi de resposta fora de les possibilitats de la política, s' haurà de realitzar en un GatewayScript ubicat després de la política d' error.

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. ([Veure apartat 2.8.1.3](#))

En el cas que es vulgui modificar un camp o l' estatus code d' un error concret, s' haurà de fer després de la política sempre que l' API ho necessiti.

2.8.2 Escenaris i algorismes


A continuació, es mostra a través d' un diagrama els passos que seguirà la política per realitzar la captura d' errors:



Aquest document s' ha basat en la plantilla publicada al MQS Disseny Detallat v1.2

2.8.3 Altres vistes i informació addicional

En fer ús d' aquesta política s' ha de tenir en compte les consideracions següents:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

- A la plantilla de desenvolupament base de l'API s'inclouran dos Catch, un per capturar els errors "**BadRequestError**", "**ConnectionError**" i "**JavaScriptError**", i un altre Catch per capturar l'error "**default**". Ambdós Catches estaran associats a la política de gestió d'errors, la qual permet personalitzar els errors d'estat.
- La política ofereix una sèrie d'errors base per poder ser referenciats a l'hora d'aixecar les excepcions. Igualment, també permet afegir més errors als ja inclosos, a través de la propietat "**erroresParticularsCtti**". [Per a més detall vegeu el punt de l'annex 4.3.](#)

3. DISSENY DETALLAT (EXTENSIONS)

Les extensions són mòduls de programari (lògica i codi) dissenyats per intervenir en diferents fases del cicle de vida de les sol·licituds i respostes API, permetent-nos influir en el comportament de la nostra infraestructura de manera dinàmica. En la fase "**pre-request**", aquestes extensions possibiliten la validació anticipada, la personalització de paràmetres i l'adaptació de dades abans que assoleixin el servidor de destinació. D'altra banda, en la fase "**post-response**", ens ofereixen la capacitat d'analitzar, transformar i enriquir les respostes generades, millorant així la qualitat i rellevància de la informació lliurada al consumidor final.

Les extensions, igual que les polítiques, dins de l'API Connect es desplegaran als Gateways (Datapowers), però a diferència d'aquestes últimes, les extensions sempre s'executaran de forma global per a qualsevol tipus d'API. Per contra, l'execució i ús de les polítiques quedarà supeditat al desenvolupador, ja que podrà utilitzar-les o no, en funció de les necessitats del seu projecte.

Els tipus d'extensions customitzades que es dissenyaran i implementar estaran compreses en les següents etapes:

- **Pre-Request:** es refereix a la fase primerenca de processament d'una sol·licitud API, just abans que s'envii al servidor de destinació. 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 pel servidor de destinació. 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.

3.1 Component Pre-Request - Generar Traceld (ctti-trace-id)


3.1.1 Disseny del component

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

L'extensió no tindrà paràmetres d'entrada, però si verificarà si ve o no informada la capçalera '**X-Ctti-Traceld**' en la invocació a l'API.

Aquests són els passos que es realitzen en el codi principal:

1. S'obté el valor de la capçalera '**X-Ctti-Traceld**' utilitzant la llibreria de context "**Context**".

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

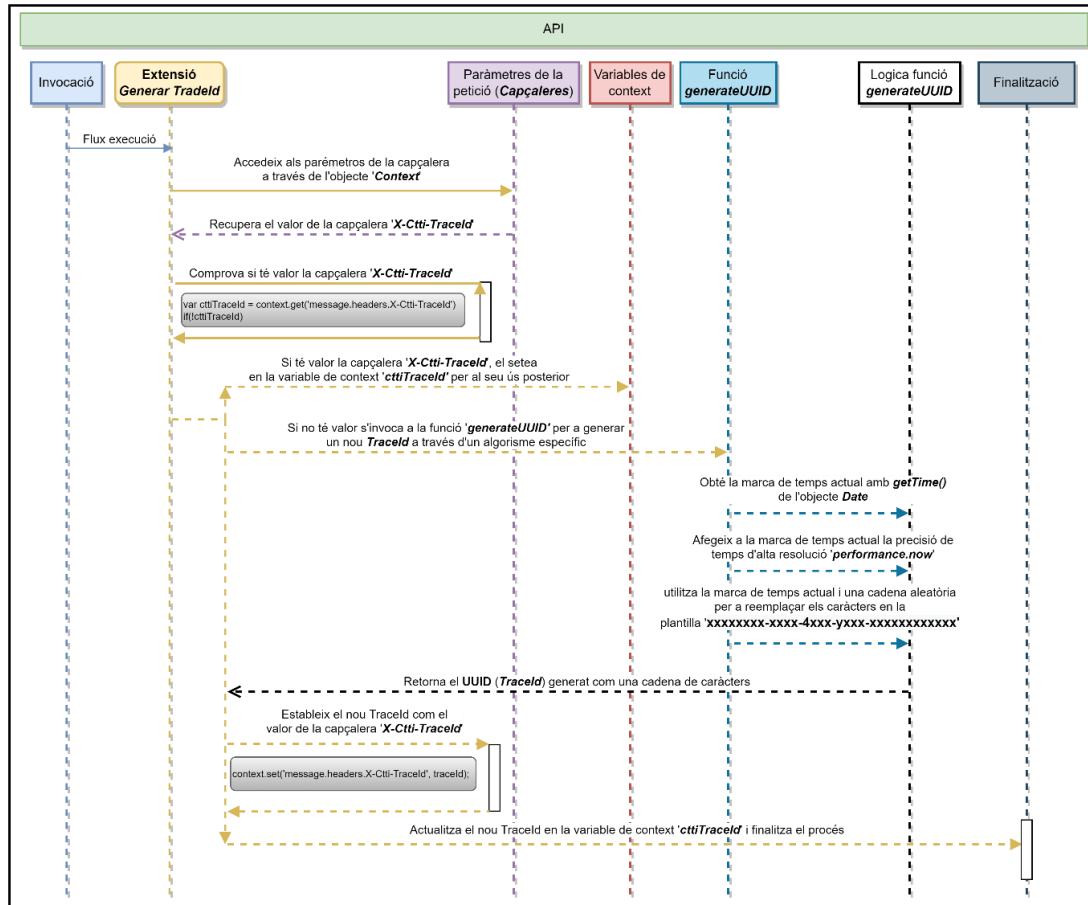
2. Es comprova si existeix un valor per a '**X-Ctti-Traceld**'. Si existeix, el valor es manté, es guarda el seu contingut en la variable de context '**cttiTraceld**' per al seu posterior ús en l'API, i es finalitza l'execució de l'extensió.

Si no n'hi ha, s'executa el bloc de codi següent:

3. S'anomena a la funció '**generateUUID()**' per generar un nou Traceld. Aquesta funció genera l'UUID combinant la marca de temps actual amb una cadena aleatòria de caràcters alfanumèrics i alguns caràcters separadors fixos. Els detalls de l'algoritme són els següents:
 - a. S'obté la marca de temps actual amb la funció **getTime()** de l'objecte **Date**.
 - b. Si està disponible, s'afegeix la precisió de temps d'alta resolució de **performance.now()** a la marca de temps actual per millorar l'aleatorietat dels UUID generats.
 - c. S'utilitza la marca de temps actual i la cadena aleatòria de caràcters per reemplaçar els caràcters **x** i **y** a la plantilla '**xxxxx-xxxx-4xxx-yxxx-xxxxxx**'. Els caràcters "**x**" són reemplaçats per valors aleatoris de 0 a f (hexadecimal), mentre que els caràcters "**i**" són reemplaçats per valors aleatoris de 8, 9, A o B (hexadecimal).
 - d. L'UUID generat es retorna com una cadena de caràcters.
4. S'estableix el nou Traceld com el valor de capçalera '**X-Ctti-Traceld**', i s'actualitza en la variable de context '**cttiTraceld**'.

3.1.2 Escenaris i algorismes

A continuació, es mostra a través d'un diagrama els passos que seguirà l'extensió per generar un nou Traceld (**UUID**) si no ve informat:



Aquest document s'ha basat en la plantilla publicada al MQS
Disseny Detallat v1.2

3.1.3 Altres vistes i informació adicional

En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:

- S'ha afegit en el codi de l'extensió de **capçaleres de seguretat**, el **message.headers** permès de sortida per a la capçalera **"X-Ctti-Traceld"**, el valor del qual es recupera de la variable de context **"cttiTraceld"** que es crea dins d'aquesta extensió.

3.2 Component (Pre-Request) - Validació CORS (ctti-request-cors)


3.2.1 Disseny del component

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

Aquesta extensió custom permet principalment dos punts sobre la d'IBM.

- Ens dona la possibilitat que l'error per CORS sigui transformat a l'error per defecte establert en la política d'errors.
- Ofereix 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 permet 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

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

HttpCode	HttpMessage	Descripció
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.

Totes les validacions que realitza l'extensió seran configurables mitjançant propietats del propi API.

L'extensió validarà les capçaleres CORS que arribaran per part del sistema origen de la petició, amb els CORS que l'API té configurats, generant en funció d'això unes metadades que assigna a les respectives capçaleres. Per a això, serà necessari incloure una sèrie de propietats en la pròpia API per part del desenvolupador.

Les propietats que s'han d'estar incloses a l'API seran, per al funcionament de l'extensió serien:

- **cors-enabled**. Indicador per saber si cal fer la validació de CORS. Pren els valors **true** (fa validació) o **false**. 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 procedirà a realitzar la validació de CORS.
- **cors-origin**. Propietat on s'indiquen els valors vàlids de la capçalera **Origin** permesos per a les peticions a l'API, separats per comes.
Exemple: <https://origen1>,<https://origen2>,<https://origen3>
- **cors-headers**. Propietat on s'indiquen els valors vàlids de la capçalera **Allow-Headers** permesos per a les peticions a l'API, separats per comes. **Exemple:** *Origin, X-Requested-With, Content-Type, Accept, authorization*.
- **cors-methods**. Propietat on s'indiquen els valors vàlids de la capçalera **Allow-Methods** permesos per a les peticions a l'API, separats per comes. **Exemple:** *POST,GET*.
- **cors-credentials**. Propietat on s'indica el valor vàlid de la capçalera **Allow-Credentials**. Tindrà el valor **true** o **false**.

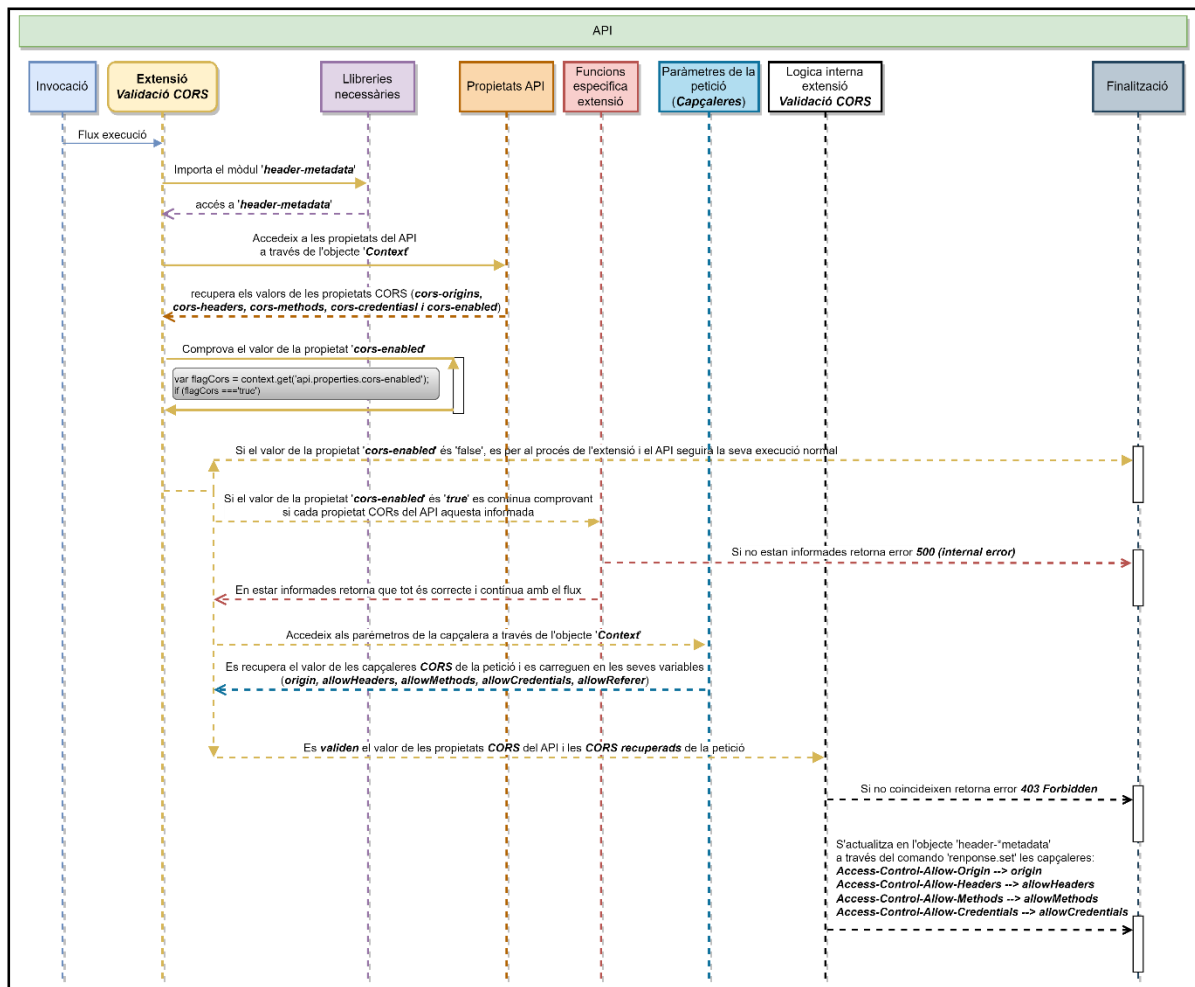
Aquests són els passos que es faran en el codi de l'extensió:

1. Es carregarà la biblioteca **'header-metadata'** necessària per a l'execució de l'extensió.
2. S'accedeix a totes les propietats de l'API a través de l'objecte **"Context"**, i d'aquesta manera obtenir el valor de les propietats associades a l'extensió CORS (**cors-origins, cors-headers, cors-methods, cors-credentials, cors-enabled**).
3. Es comprova el valor de la propietat **"cors-enabled"**; si el valor és **"true"**, es continua amb la validació de capçaleres CORS. Altrament es para el procés i l'API seguiria la seva execució normal.
4. A través de funcions específiques es verifica que cadascuna de les propietats recuperades en el pas anterior estiguin definides amb un valor. En cas contrari, es mostrarà el missatge d'error **500 internal error**, indicant en el key **"moreInformation"**, la propietat que no està ben definida.
5. S'obtenen les capçaleres CORS de la petició i es guarden en variables internes com per exemple les següents:
 - a. **origin**
 - b. **allowHeaders**
 - c. **allowMethods**
 - d. **allowCredentials**
 - e. **allowReferer**

6. Es valida cadascuna de les capçaleres CORS que s' han recuperat en el **pas 5** amb les propietats que s' han recollit en el **pas 2**, comprovant que els valors de les propietats es corresponen amb els que arriben a cada capçalera. En cas que no coincideixi la informació que ve del sistema origen amb l' API, es mostrarà el missatge d' error **403 Forbidden**. L' error es retornarà mitjançant la gestió d' errors implementada.
7. Un cop validades les capçaleres CORS, s' afegiran a l' objecte corresponent perquè vagin al Backend a la capçalera. Es realitzarà mitjançant el comandament **response.set** i informant les següents capçaleres amb el valor corresponent de les variables internes recollides en el **punt 5**:
 - a. **Access-Control-Allow-Origin** → origin
 - b. **Access-Control-Allow-Headers** → allowHeaders
 - c. **Access-Control-Allow-Methods** → allowMethods
 - d. **Access-Control-Allow-Credentials** → allowCredentials


3.2.2 Escenaris i algorismes

A continuació, es mostra a través d' un diagrama els passos que seguirà l' extensió per realitzar les validacions de les capçaleres CORS:



3.2.3 Altres vistes i informació adicional

En fer ús d' aquesta política s' ha de tenir en compte les consideracions següents:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

- S'han d'informar totes les propietats descrites en l'apartat anterior, ja que, en cas de no ser així, l'extensió retornarà un error **500 Internal error**, indicant que propietat no està ben informada.
- El valor de la propietat "**cors-enabled**" determinarà si s'aplica o no la lògica de validació CORS, sent "**false**" en el cas que no es requereixi.

3.3 Component (*Post-Response*) - Gestió de capçaleres de seguretat (*ctti-response-headers-secure*)

3.3.1 Disseny del component


Aquesta extensió s'executarà en la fase **Post-Response** del processament de l'API. Seguint l'estàndard **OWASP Secure Headers Project**, també anomenat **OSHP**, aquesta funció (Extensió) 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.

L'extensió s'inclourà tant com a post-response global policy (PostResponse), com post-error global policy (PostError). D'aquesta manera es garanteix 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, tant si va bé, com si hi ha un error.

Les capçaleres que es permetran per defecte, a més de les indicades pel desenvolupador, 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**
Pot ser usat per indicar si se li hagués de permetre 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 bloquegessin el rastreig de contingut, podent transformar tipus MIME no executables en tipus MIME executables.
- **Content-Security-Policy**
Permet als administradors d'un lloc web controlar els recursos que l'User-Agent pot carregar a una pàgina.
- **Location**
Aquest encapçalat s'utilitza 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-Traceld**

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

Aquest encapçalat inclourà el Traceld únic de la petició de l' API, permetent un seguiment eficient i una millor traçabilitat de les peticions. Aquest Traceld 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**".

L'extensió no té dades d'entrada com a tal; utilitzarà dues propietats, que hauran d' estar definides al YAML de l' API, i les capçaleres de la petició. Les propietats de l' API seran les següents:

- **cabecerasSeguridad-enabled**: Indicador per saber si cal fer la validació 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.
- **cabeceras-seguridad**: Tindrà un format JSON per permetre la conservació de les capçaleres de resposta indicades pel desenvolupador i la customització del valor de cadascuna de les capçaleres de seguretat descrites anteriorment.

La propietat contindrà els següents valors: **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 Capçalera-Custom-Example**. Si no se li assigna cap valor, es mantindran els valors per defecte que vinguin i se setejaran els corresponents en l' extensió.


Un exemple del contingut que hauria de tenir seria el següent:

```
{ \t\"Strict-Transport-Security\": \"max-age=86400; includeSubDomains\", \"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\": \"\"}
```

En el codi intern de l'extensió es crearà la variable "**cabecerasSeguridad**", la qual contindrà la llista de capçaleres de seguretat separades per comes, provinents de la propietat "**cabeceras-seguridad**" comentada anteriorment.

Aquests són els passos que es realitzaran en el codi principal

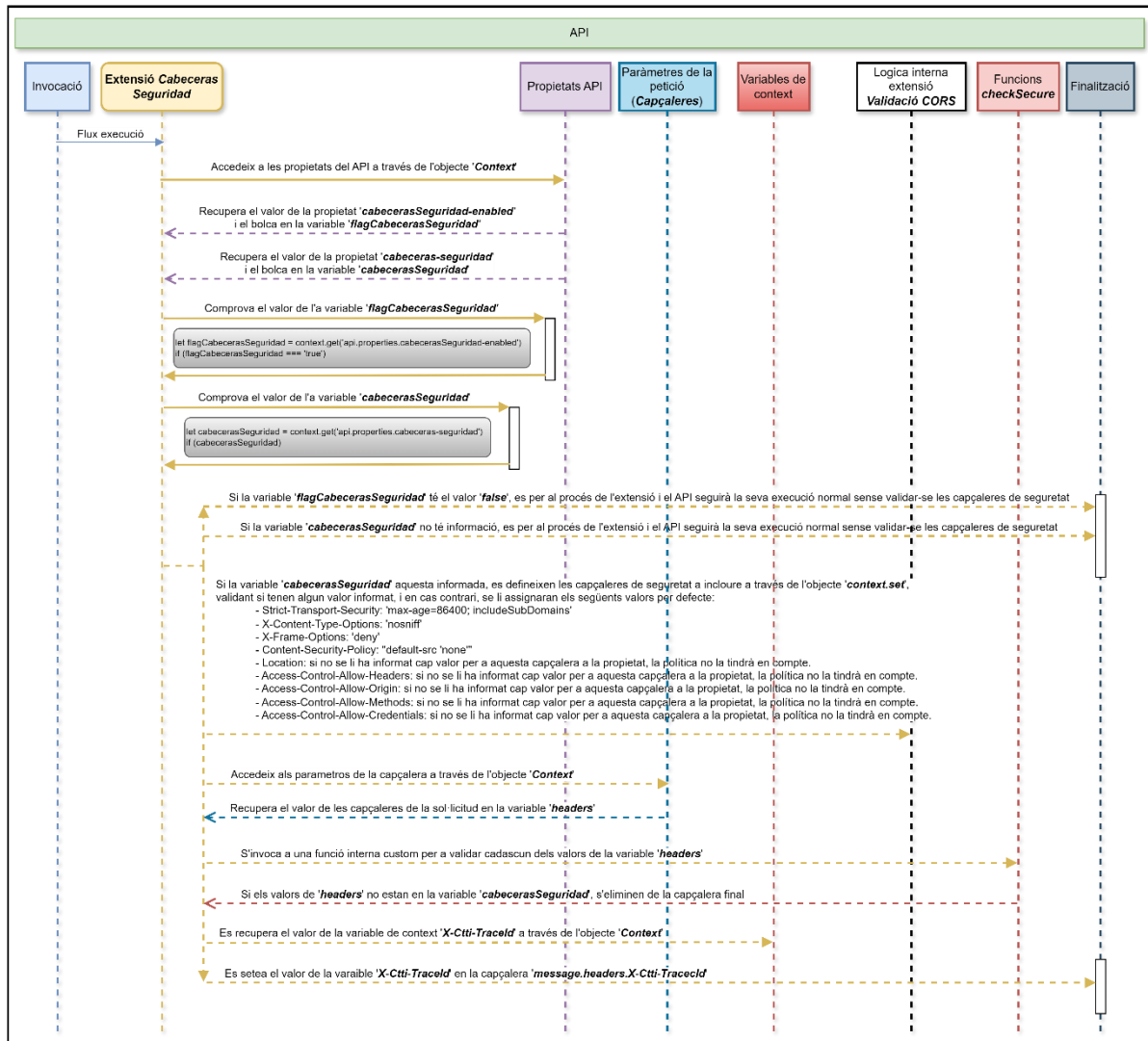
1. S'accedeix a través de l'objecte "**Context**" a les propietats de l'API i d'aquesta manera obtenim el valor de les propietats "**cabeceras-seguridad**" i "**cabecerasSeguridad-enabled**".
2. Guarda el valor de la propietat "**cabeceras-seguridad**" recuperada, en una variable interna, per exemple, "**cabecerasSeguridad**", per al seu posterior ús en la lògica de l'extensió.
3. Guarda el valor de la propietat "**cabecerasSeguridad-enabled**" recuperada, en una variable interna, per exemple, "**flagCabecerasSeguridad**", per al seu posterior ús en la lògica de l'extensió.
4. Es comprova el valor de la variable "**flagCabecerasSeguridad**"; si el valor és "**true**", es continua amb la validació de les capçaleres de seguretat. Altrament es para el procés i l'API seguiria la seva execució normal.
5. Es comprova el valor de la variable "**cabecerasSeguridad**"; si està informada, es continua amb lògica. En cas contrari, es para el procés i l' API seguirà la seva execució normal, no validant les capçaleres de seguretat.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

6. Es defineixen les capçaleres que s' inclouran com a capçaleres de seguretat mitjançant **context.set**. Es comprova si alguna de les capçaleres que s' ha definit a la propietat **cabeceras-seguridad** té algun valor informat. Si té algun valor, s' assigna aquest mateix a la capçalera corresponent. Si no és així, se' ls assignen els següents valors per defecte:
 - a. **Strict-Transport-Security**: 'max-age=86400; includeSubDomains'
 - b. **X-Content-Type-Options**: 'nosniff'
 - c. **X-Frame-Options**: 'deny'
 - d. **Content-Security-Policy**: "default-src 'none'"
 - e. **Location**: si no se li ha informat cap valor per a aquesta capçalera a la propietat, la política no la tindrà en compte.
 - f. **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.
 - g. **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.
 - h. **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.
 - i. **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.
7. Recupera totes les capçaleres de la sol·licitud actual en la variable **headers (headers = context.get('message.headers'))**.
8. Es recorren els valors de la variable **"headers"**, invocant per a cadascun d'ells a la una funció interna custom, on es compara el valor de la variable **headers** (capçaleres) amb els quals tenim en la variable **cabecerasSeguridad**. Si el valor no està en **"cabecerasSeguridad"**, s'elimina de la capçalera. D'aquesta manera, ens assegurem que romanguin a la capçalera únicament els valors definits a la propietat cabeceras-seguridad.
9. Es recupera el valor de la variable de context **"X-Ctti-Traceld"**, creada en l'**extensió "ctti-trace-id"**, i se setea a la capçalera **"message.headers.X-Ctti-Traceld"**, perquè també s'envii sempre.

3.3.2 Escenaris i algorismes

A continuació, es mostra a través d' un diagrama els passos que seguirà l' extensió per definir les capçaleres de seguretat:



Aquest document s'ha basat en la plantilla publicada al MQS Disseny Detallat V1.2

3.3.3 Altres vistes i informació adicional


En fer ús d'aquesta política s'ha de tenir en compte la consideració següent:

- En el cas que s'introdueixi la mateixa capçalera duplicada amb diferents valors, en la propietat de **cabeceras-seguridad**, l'extensió li assignarà el valor de la darrera entrada de la capçalera que hi hagi.

4. ANNEXOS

4.1 Procediment per importar polítiques a nivell de LTE

En aquest apartat s'adjunta la guia d'importació de les polítiques a nivell de **LTE (Local Testing Environment)**, de cara a poder realitzar les proves necessàries en l'entorn local. S'inclou al seu torn, dins del document, els passos per crear un scriptor que faciliti l'automatització del procés de l'arrencada de LTE i la importació de les polítiques en l'entorn local (**Toolkit**).

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0




Manual_importacio_p
olitiqes_v1.0.pdf

4.2 Procediment per desplegar polítiques i extensions sobre la plataforma

Al llarg del cicle de vida de l' API en un projecte dins l' API Manager de CTTI, l' OFT pot identificar problemes o necessitats en projectes en curs, que facin recomanable o necessària la creació de polítiques o extensions addicionals més enllà de les definides en aquest document. En aquest cas, s' estudiaria la realització d' evolutius, els quals es mobilitzarien per abordar aquestes necessitats o problemes específics trobats. Tanmateix, és important destacar que la implementació d' aquests evolutius no seria responsabilitat directa de l' OFT, sinó que, durant la generació de la proposta de cada evolutiu, s' acordarà que equip haurà de realitzar aquest desenvolupament i desplegament.

Els desenvolupadors pertanyents a aquests equips, encarregats de desenvolupar les noves polítiques o extensions, hauran de seguir els següents passos per procedir al desplegament de les mateixes sobre la plataforma de l' API Manager de CTTI:

1. S'obre [un nou tiquet JIRA d'ACOAPIM](#), per tal de notificar a l'OFT la necessitat de pujar polítiques/extensions a la plataforma de Sandbox per realitzar les proves corresponents.
2. Dins del tiquet, es proporcionen les informacions requerides per procedir amb l' operació. Un exemple de petició JIRA vàlida contindran les següents informacions:
 - a. **Organisme/Projecte Afectat:** CENTRE TELECOMUNICACIONS I TECNOLOGIES DE LA INFORMACIÓ.
 - b. **Si no el projecte en el desplegable, aquí:** "En aquest camp s'indica el nom del projecte que demana la pujada de política/extensió".
 - c. **Resum:** "En aquest camp s'indica el motiu de la petició (Pujada de política i/o extensió)".
 - d. **Descripció:** "En aquest camp s'indica amb més detall tota informació necessària, com els entorns per on es desplegarà la política/extensió (Sandbox si és la primera pujada a la plataforma), el motiu/necessitat d'usar aquesta política/extensió, descripció breu del funcionament de la política/extensió, adjuntar el document de disseny tècnic, etc."
 - e. **Adjunt:** "En aquest apartat, s'adjunten els arxius .zip que contenen les polítiques i/o extensions que es vagin a desplegar, juntament amb la documentació necessària per entendre-les*".
**Nota: En el cas que la política o extensió contingui informació sensible, es pot adjuntar en el seu cas un link al Sharepoint de CTTI en el camp de 'Descripció' en lloc d'adjuntar directament l'arxiu .zip o discutir durant el transcurs del JIRA la millor forma de passar el codi d'aquesta política/extensió.*
3. L'OFT, en rebre el tiquet **JIRA**, l'avaluarà en detall, comprovant que tot l'indicat i sol·licitat és correcte i té sentit. Un cop es comprovi tot i s'aprovi la pujada, es prendrà la mesura necessària per pujar el codi:
 - Si es tracta d'una **política**, l'OFT la pujarà a l'entorn que correspongui a través de la plataforma d'API Manager d'IBM Cloud (secció '**Manage Catalogs**'). Per a això, es faran els següents passos:
 1. Primer, s'iniciarà sessió amb l'usuari de GICAR a l'IBM Cloud accedint a la url <https://cloud.ibm.com/authorize/gicar>
 2. Un cop s'accedeix al Cloud, s'obrirà l'API Manager polsant primer a **API Management** → **Services** i, a la següent pantalla, a la següent pantalla, a la següent

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

3. Un cop s'accedeixi a l'API Manager, es polsarà a **Manage Catalogs**
 4. A continuació, es polsarà en el **catàleg** en el qual s'hagi de pujar la política
 5. Després, es polsarà a **Spaces** i, a continuació, es polsarà en l'espai del **codi de diàleg** que ha demanat pujar la política (aquest pas no es farà si es puja al catàleg de **Sandbox**, atès que aquest no compta amb espais)
 6. Un cop s'hagi accedit a l'espai, es polsarà a **Space Settings** (si es fa per al catàleg de **Sandbox**, s'ha de polsar a **Catalog Settings**) i, a continuació, a **Gateway services**
 7. Per al Gateway service corresponent, es premerà en els tres punts de la dreta i en **View policies**
 8. Es polsarà a **Upload**, s'importarà el fitxer zip amb la política i es tornarà a polsar a **Upload**. Amb això es pujarà correctament no només a l'espai on s'ha pujat, sinó al catàleg sencer, ja que es puja realment al Gateway service associat a l'espai i, en CTTI, el Gateway service és el mateix per a tots els espais dins d'un mateix catàleg.
- Si es tracta d'una **extensió**, l' OFT la pujarà a través de l' ús dels comandaments CLI, seguint els passos a continuació:

1. Primer, s'iniciarà sessió amb l'usuari de GICAR a l'IBM Cloud accedint a la url <https://cloud.ibm.com/authorize/gicar>
2. A continuació, s'obre una consola de comandaments dins la carpeta on es trobi instal·lat el CLI d'API Connect i s'inicia sessió amb el comandament

```

apic login --server {URL_gestió_instància} --sso
Context? provider
Please copy and paste the url
https://cloud.ibm.com/apis/apim/{url_ctti} to a browser to start
the authentication process.
Do you want to open the url in default browser? [i/n]: i
API Key? {API_Key rebuda al obrir el enllac}
Logged into {URL_gestió_instància} successfully
  
```

3. A continuació, es crea una global policy (extensió) mitjançant el comando


```

apic global-policies:create --catalog {nom_catàleg} --configured-
gateway-service {nom_servei_gateway} --org {nom_organització} --
server {URL_gestió_instància} --scope {scope} [--space {espai}]
{nom_fitxer_global_policy}
  
```

4. Ara, s'escriu l'URL de la global policy necessària en un arxiu i mitjançant el següent comandament:

```

apic global-policies:get --catalog {nom_catàleg} --configured-
gateway-service {nom_servei_gateway} --org {nom_organització} --
server {URL_gestió_instància} --scope {scope} [--space {espai}]
{nom_política}:{versió_política} --fields url
  
```

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

4. Després, s'edita l'arxiu **GlobalPolicy.yaml** generat, substituint el camp **'url'** per **'global_policy_url'**. L'arxiu resultant tindria el format següent:

```
global_policy_url: >-
https://{nom_host_servidor}/api/catalogs/{id_catàleg}/configured-
gateway-services/{id_servei_gateway}/global-
policies/{id_política}
```

5. Finalment, es designa la global policy segons la naturalesa que té:

- Si la global policy és del tipus **pre-hook**, es designa amb el següent comandament:

```
apic global-policy-prehooks:create --catalog {nom_catàleg} --
configured-gateway-service {nom_servei_gateway} --org
{nom_organització} --server {URL_gestió_instància} --scope
{scope} [--space {espai}] GlobalPolicy.yaml
```

- Si la global policy és del tipus **post-hook**, es designa amb el següent comandament:

```
apic global-policy-posthooks:create --catalog {nom_catàleg} --
configured-gateway-service {nom_servei_gateway} --org
{nom_organització} --server {URL_gestió_instància} --scope
{scope} [--space {espai}] GlobalPolicy.yaml
```

- Si la global policy és del tipus **post-error**, es designa amb el següent comandament:

```
apic global-policy-errors:create --catalog {nom_catàleg} --
configured-gateway-service {nom_servei_gateway} --org
{nom_organització} --server {URL_gestió_instància} --scope
{scope} [--space {espai}] GlobalPolicy.yaml
```

Nota: Cal destacar que, en el cas que s'hagi de pujar una nova extensió (global policy), caldria eliminar prèviament l'extensió antiga amb el següent comandament:


```
apic global-policy-{tipus_de_extensió}:delete --catalog
{nom_catàleg} --configured-gateway-service {nom_servei_gateway} --
-org {nom_organització} --server {URL_gestió_instància} --scope
{scope} [--space {espai}] GlobalPolicy.yaml
```

A continuació, s'eliminarà la global policy també amb el comandament **delete**, indicant l'extensió específica que es vulgui eliminar.

```
apic global-policies:delete --catalog {nom_catàleg} --configured-
gateway-service {nom_servei_gateway} --org {nom_organització} --
server {URL_gestió_instància} --scope {scope} [--space {espai}]
{nom_política}:{versió_política}
```

Un cop eliminada, es tornarien a seguir els passos anteriors des del comandament per definir novament la global policy com l'extensió amb el tipus que s'especifica.

4. Un cop l'OFT hagi pujat la política/extensió, se li notificarà al proveïdor que correspon a través del tiquet **JIRA**. En cas que aquesta fos la primera pujada de la política/extensió, que ha de ser al catàleg de **Sandbox**, s'indicarà al proveïdor que podrà realitzar les proves corresponents. Un cop

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

les proves siguin correctes i es comprovi que no provoca cap fallada a la plataforma, es procedirà amb la pujada en els altres catàlegs.

- Un cop es puja la política/extensió a tots els entorns requerits, es procedirà amb el tancament del tiquet **JIRA**, considerant finalitzat el procés.

4.3 Errors custom 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 inclou per defecte, perquè puguin 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 permet incloure nous errors customs per a poder ser referenciats en aixecar una excepció, en funció de les necessitats del projecte.

Per poder dur a terme això, s'ha de realitzar a través de la propietat "**erroresParticularesCtti**", la qual està 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:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	3292 (<i>Arquitectura API Manager</i>)	N. revisió doc.: 1.1
	Disseny detallat de la solució	
	N. versió aplicació: 1.0	Build: 1.0

properties:

```

erroresParticularesCtti:
  value: >-
    EXAMPLE: {"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 **"E0"**, la política internament en recollir-lo de la propietat i bolcar-lo a la taula d'errors, ho afegirà.

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

- Error custom indicat a la propietat: **"CTE0190"**
- Error carregat per la política a la taula d' errors per a la seva utilització seria: **E0CTE0190**

Es recomana que els errors custom que es defineixin, mantinguin, dins del que es pugui, la nomenclatura **E0XXX**.

Nota: L'error **E0XXX** representa l'error que es retornaria, en el cas que en escalar-se un error s'hagi informat un codi d'error (**E0AAA**), que no es trobi a la taula d'errors o afegit a través de la propietat proporcionada per la política.