****

Gestión y mantenimiento de Custom Policies en IBM API Management 4.0.3

Índice

[1. Introducción 4](#_Toc441675415)

[1.1. Propósito del documento 4](#_Toc441675416)

[1.2. Vocabulario común y acrónimos 4](#_Toc441675417)

[2. Custom Policies 5](#_Toc441675418)

[2.1. Tipos de custom policies y propiedades 5](#_Toc441675419)

[2.2. Estructura de interna de las políticas 8](#_Toc441675420)

[2.3. Caso de uso práctico: inserción de las políticas paso a paso 12](#_Toc441675421)

[2.4. Requisitos de funcionamiento: Plan CheckAPP 14](#_Toc441675422)

[3. Anexos 17](#_Toc441675423)

[3.1. Anexo (I): Fichero YAML de configuración 17](#_Toc441675424)

[4. Revisión 18](#_Toc441675425)

[5. Firma 19](#_Toc441675426)

Control de Ediciones

| **Versión** | **Fecha** | **Modificación** | **Autor** |
| --- | --- | --- | --- |
| V0.1 | 23/10/2015 | Creación de documento | IBM |
| V0.2 | 11/01/2016 | Actualización - OAuth Auth Code | IBM |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introducción

## Propósito del documento

Para proporcionar la posibilidad de añadir funcionalidades extra en las reglas de procesamiento asociadas a los recursos que forman un API, se ha decido definir un conjunto de “custom policies”. Este tipo de políticas suelen ser utilizadas para llevar a cabo tareas como el ruteo de peticiones en un proxy o añadir seguridad a los recursos publicados en una API.

En el siguiente documento, se explica detalladamente cada una de las custom policies desarrolladas y la forma de ser utilizadas en un recurso de un API.

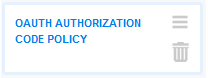
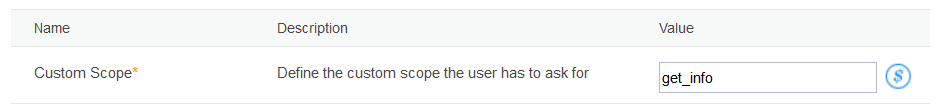
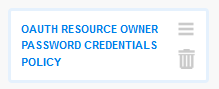
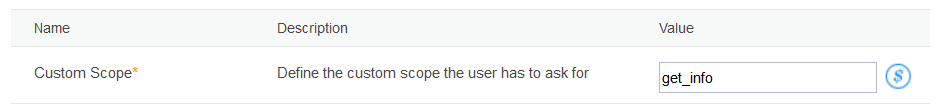
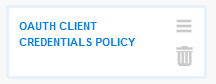
## Vocabulario común y acrónimos

| **Término Negocio** | **Definición** |
| --- | --- |
| API | *Application Programming Interface*. Imagen pública de las empresas que exponen ciertos activos de datos y/o funciones definidos expresamente para su consumo por desarrolladores a través de una interfaz documentada y sencilla de utilizar. |
| OAuth | OAuth es un protocolo de autenticación que permite a los usuarios aprobar una solicitud para que una aplicación actúe en su nombre sin la necesidad de compartir sus credenciales |

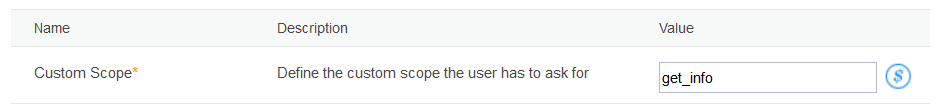
# Custom Policies

## Tipos de custom policies y propiedades

Las tres “custom policies” inicialmente desarrolladas son las siguientes, y cumplen los requisitos establecidos en el inicio del proyecto:

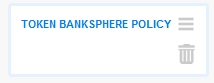
1. **OAuth Authorization Code Policy**  
   Esta custom policy es utilizada para validar el token OAuth contenido en la cabecera HTTP Authorization que llega en la petición. En situaciones normales, esta acción debe ser definida como la primera de cualquier recurso que quiera estar securizado por OAuth – Authorization Code.  
   Cuando se despliega esta custom policy durante el desarrollo del recurso deberá definirse la siguiente propiedad:
   * Scope: Se define el “scope” personalizado que el usuario debe pedir al realizar la petición del token. Por ejemplo, get\_informacion\_cuentas.  
     
2. **OAuth Resource Owner Password Credenciales Policy**  
   Esta custom policy es utilizada para validar el token OAuth contenido en la cabecera HTTP Authorization que llega en la petición. En situaciones normales, esta acción debe ser definida como la primera de cualquier recurso que quiera estar securizado por OAuth – Resource Owner Password Credentials.  
   Cuando se despliega esta custom policy durante el desarrollo del recurso deberá definirse la siguiente propiedad:
   * Scope: Se define el “scope” personalizado que el usuario debe pedir al realizar la petición del token. Por ejemplo, get\_informacion\_cuentas.  
     
3. **OAuth Client Credenciales Policy**Esta custom policy es utilizada para validar el token OAuth contenido en la cabecera HTTP Authorization que llega en la petición. En situaciones normales, esta acción debe ser definida como la primera de cualquier recurso que quiera estar securizado por OAuth – Client Credentials.

Cuando se despliega esta custom policy durante el desarrollo del recurso deberá definirse la siguiente propiedad:

* + Scope: Se define el “scope” personalizado que el usuario debe pedir al realizar la petición del token. Por ejemplo, get\_informacion\_cajeros.  
    

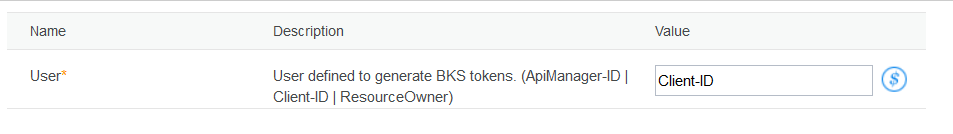
1. **Token BankSphere Policy**  
   Siguiendo las especificaciones recibidas, se requiere la generación de un token propietario (Token BankSphere) para poder ser propagado y enviado en la cabecera de seguridad de las invocaciones a servicios web.

Esta política puede ir encadenada de alguna de las dos políticas de OAuth indicadas anteriormente o sola, generando el token en función a una serie de parámetros definidos en el fichero de configuración.



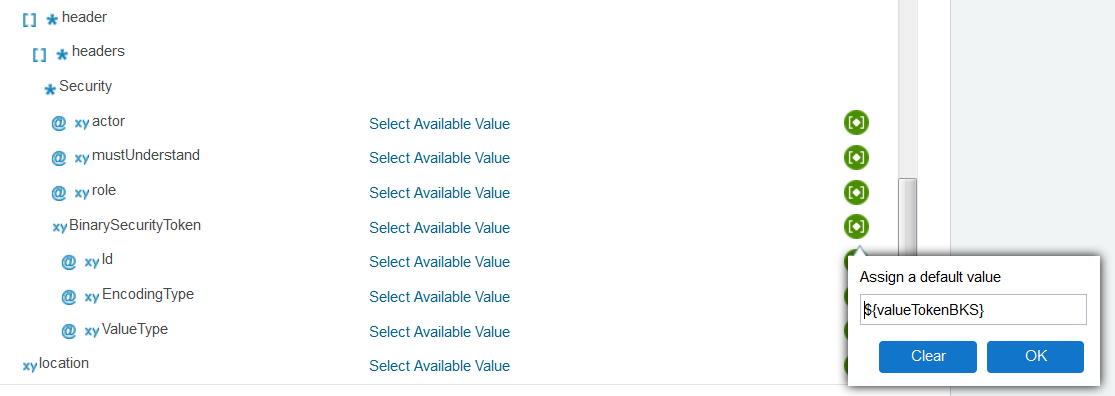
Cuando se despliega esta custom policy durante el desarrollo del recurso se deberá definir la siguiente propiedad:

* + UserType: define el tipo de usuario que va a ser utilizado en la generación del token BKS. Los tres valores diferentes que pueden ser tomados son:
    - ApiManager-ID. Utilizará el valor de un usuario genérico de API Management definido en el fichero de configuración.
    - ResourceOwner. Solamente podrá ser utilizado cuando se utilice la custom policy definida para OAuth Resource Owner Password Credenciales. Está acción será utilizada a continuación y utilizará la información del Resource Owner extraída del token.
    - Client-ID. Solamente podrá ser utilizado cuando se utilice las custom policy definidas para OAuth Resource Owner Password Credenciales o OAuth Client Credenciales. Está acción será utilizada a continuación y utilizará la información del Client-ID extraída del token.

****

Una vez haya sido configurada la acción en uno de los servicios, el token BankSphere generado podrá ser utilizado como entrada de alguno de los campos posteriores. Para hacer referencia al token será necesario acceder a la sección de valores e indicar en el campo correspondiente: **${valueTokenBKS}.**

En situaciones normales, este token viaje en el campo de la cabecera de seguridad de la invocación al servicio web:

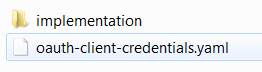


## Estructura de interna de las políticas

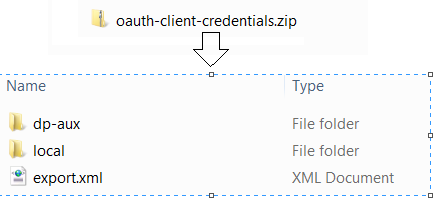
En la siguiente sección se explica la estructura interna que tiene el fichero .ZIP que contiene cada una de las "custom policies" definidas. A continuación, se presenta la estructura de la política tomando como ejemplo OAuth Client Credenciales Policy.

1.- Fichero ZIP contendor de la política: 

2.- En su interior está incluida la implementación de la política y un fichero de configuración. El fichero de configuración es del tipo YAML y permite la representación de los datos mediante el uso de listas y asignación de valores. Los datos requeridos son: la información acerca de la política, que contiene una descripción con los datos de contacto, versión y demás información explicativa, su tipo de uso: SOAP y/o REST y un esquema de propiedades que podrán ser utilizadas como parámetros de entrada en la política. Para este caso, la única propiedad será el scope.



En la sección de Anexos se puede encontrar un ejemplo de Fichero YAML que puede servir como base para la creación de nuevos.

3.- Dentro de la carpeta "implementation" (que debe tener siempre el nombre indicado) quedará contenida la exportación que se habrá realizado en DataPower con las acciones que se desean ejecutar al tener una política. Será importante normalizar los nombres de los ficheros, en este caso oauth-client-credetetials, para que funcione de la manera esperada.

Para realizar la **exportación del contenido que tendrá la carpeta implementation** se deberán seguir los **siguientes pasos**:

1. Se accede al IBM DataPower Gateway donde hayan sido creadas y preparada la implementación de las custom policies. En este caso, se ha realizado en el DataPower del entorno de Desarrollo cuya URL e IP corresponde con los siguientes datos:

* URL: islgatwd01.scisb.isban.corp
* IP: 180.101.186.26

Accediendo a la interfaz de gestión para poder realizar la exportación.

1. Se accede a la sección de Export Configuration.
2. Se exporta la implementación de la política desde DataPower. Desde la interfaz de usuario de DataPower, se exportan los objetos y archivos que se hace referencia en la regla de procesamiento principal como un archivo comprimido. Ningún otro objeto o archivo debe ser exportado. La siguiente lista de archivos es un ejemplo de configuración de una exportación de DataPower:

*Archivo: oauth-authorization-code-main.zip*

*11819 11-19-2014 22:53 dp-aux/basetypes.xml*

*3028207 11-19-2014 22:53 dp-aux/drMgmt.xml*

*6456 11-19-2014 22:53 dp-aux/map-dmz.xsl*

*7299 11-19-2014 22:53 dp-aux/management.xsl*

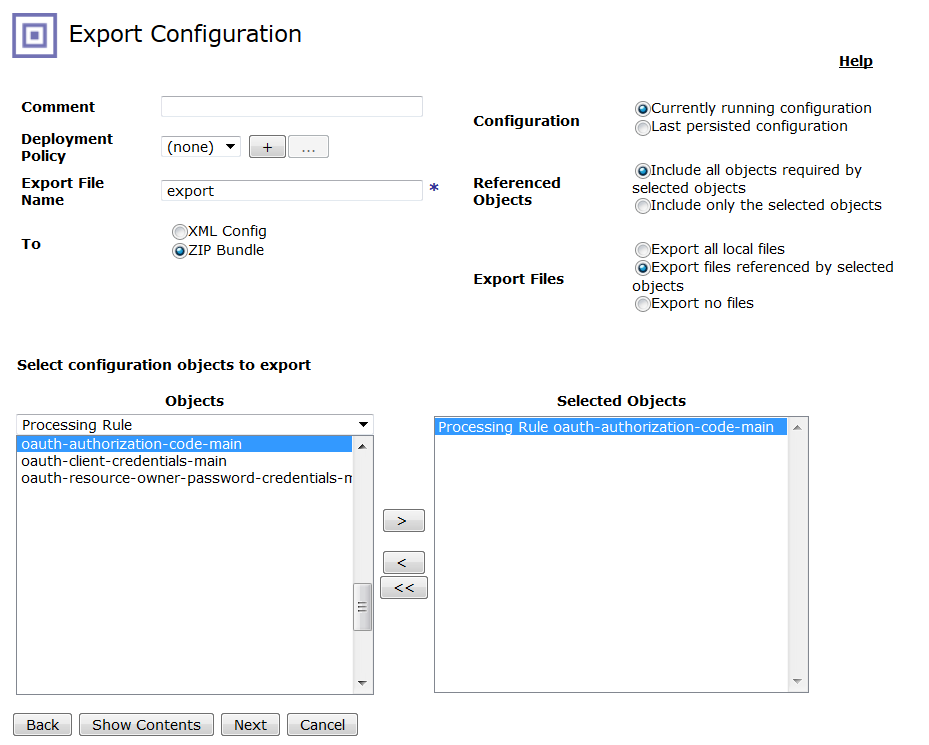
*23130 11-19-2014 22:53 dp-aux/SchemaUtil.xsl*

*216003 11-19-2014 22:53 dp-aux/clixform.xsl*

*5284 11-19-2014 22:53 local/policy/oauth-authorization-code/ValidateOAuthAPImCustomPolicy\_Authorization\_Code.xsl*

*5061 11-19-2014 22:53 export.xml*

El ejemplo muestra el archivo de configuración exportado (export.xml) que contiene todas las acciones de procesamiento y objetos de DataPower, los archivos de referencia, a continuación, los esquemas de configuración de DataPower que se crean mediante la exportación de configuración de la interfaz de usuario DataPower.



1. Una vez generado el archivo ZIP, éste será colocado dentro de la carpeta "implementation" citada con anterioridad.

En el caso de **necesitar generar una nueva custom policy** se deberán seguir los **siguientes pasos**:

1. Se accede al IBM DataPower Gateway donde hayan sido creadas y preparada la implementación de las custom policies. En este caso, se ha realizado en el DataPower del entorno de Desarrollo cuya URL e IP corresponde con los siguientes datos:

* URL: islgatwd01.scisb.isban.corp
* IP: 180.101.186.26

Accediendo a la interfaz de gestión para poder realizar las tareas.

1. Crear una regla de procesamiento (processing rule) principal que se llama "nombrePolitica-main", donde nombrePolitica es el nombre de la política definida por el usuario (como se define en el archivo YAML). La regla de procesamiento deberá tener establecida las siguientes propiedades:
   1. Rule direction: Both Directions
   2. Non-XML Processing: On
2. Crear una o más acciones de procesamiento. Cada acción de procesamiento debe tener un nombre único y debe comenzar con el nombre de la política definida por el usuario (como se define en el archivo YAML). Las siguientes acciones de procesamiento están disponibles:
   1. Configurar el acceso a las propiedades de entrada en XSLT (policyProperties). Cuando la política está incluida en un API, desarrollador API debe introducir los ajustes de entrada o variables para una propiedad en particular. La función policyProperties() proporciona el mecanismo para recuperar estos valores de entrada o variables en tiempo de ejecución.
   2. Configurar el acceso al contexto de ejecución en XSLT (getContext). Cuando se invoca una API, la información acerca de la solicitud de tiempo de ejecución se puede acceder mediante el uso de una política definida por el usuario, y esta información se conoce como el contexto de tiempo de ejecución. La función getContext() proporciona el mecanismo para acceder al contexto en tiempo de ejecución.
   3. Configurar el acceso al payload de entrada en una XSLT (payloadRead). Algunas políticas requieren acceso al mensaje de entrada proporcionado por la aplicación. El acceso al mensaje de entrada puede ser difícil, ya que esta información podría cambiar a medida que se ejecutan otras acciones. La función payloadRead() proporciona un mecanismo para obtener la información del mensaje de entrada en el momento de la ejecución de políticas.

La función payloadRead () devuelve un conjunto de nodos XML que contiene la carga útil de la solicitud. Si la carga está en formato JSON, un conjunto de nodos JSONx se devuelve que puede ser manipulado dentro de una hoja de estilo XSLT. Si la carga no está en formato JSON o XML, el conjunto de nodos que se devuelven están vacios.

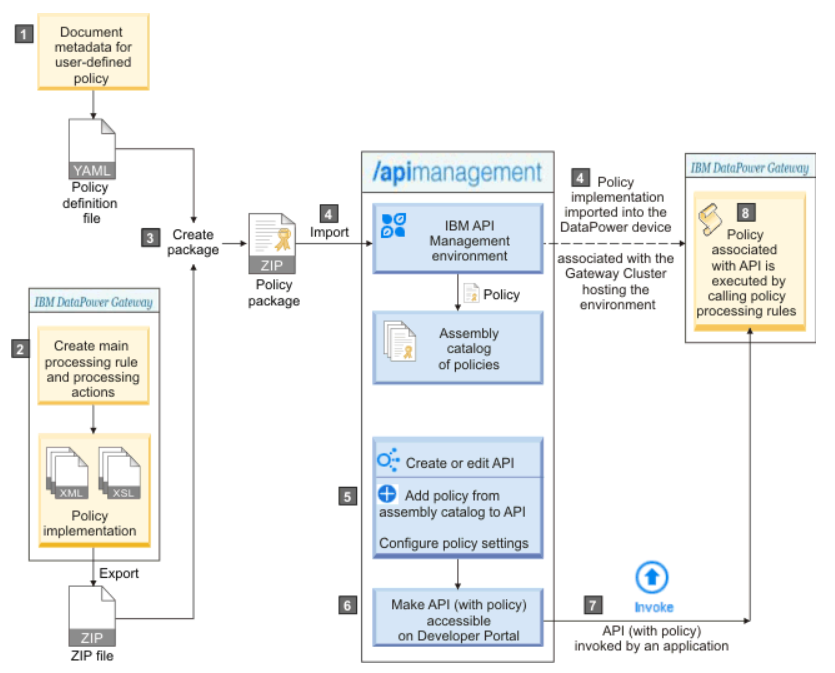
* 1. Configurar el acceso a las cabeceras HTTP en una XSLT (getContext). La información de la cabecera HTTP de la petición puede ser recuperada desde el contexto de la petición. Es importante resaltar que los nombres de las cabeceras HTTP se normalizarán a minúsculas.

La función getContext() será la encargada de proporcionar el mecanismo para acceder a la información que contiene la cabecera HTTP.

* 1. Configurar de la custom policy para producir información de error. Si se requiere la implementación de políticas para producir información de error cuando se produce un fallo, se debe configurar la aplicación llamando a la plantilla de APIM-error. Un ejemplo de código sería el siguiente:
  2. Uso de variables. Se utiliza la plantilla setVariable para establecer una variable en tiempo de ejecución con un valor de cadena especificado. Este valor puede entonces ser recuperada utilizando la función getVariable(), o mediante la asignación del valor uso de la notación ${nombreVariable}.

1. Una vez esté completa la custom policy y definidas todas sus acciones, se realizará la exportación del mismo que ha sido explicado al inicio de este punto.

Por último, en el siguiente diagrama se muestra un resumen acerca de cómo se crean y se ejecutan las políticas que son definidas en API Management.

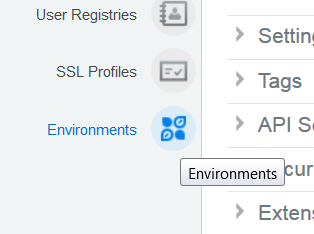


## Caso de uso práctico: inserción de las políticas paso a paso

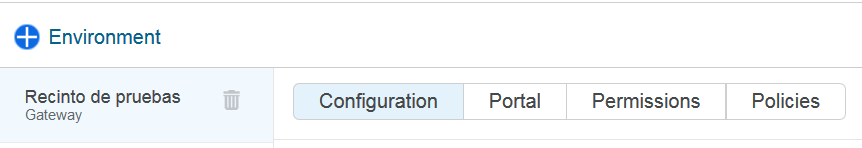
A continuación, se presenta un ejemplo práctico con los pasos que habrá que seguir para proceder a la inserción de las custom policies desarrolladas. Los pasos son los siguientes:

1.- Se accede al portal web del API Manager: https://islmnagd01.scisb.isban.corp/apimanager

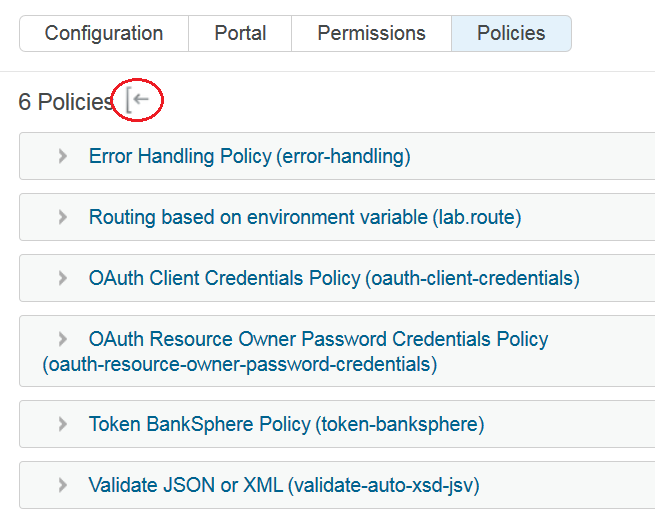
2.- Se accede a la sección de "Environments":



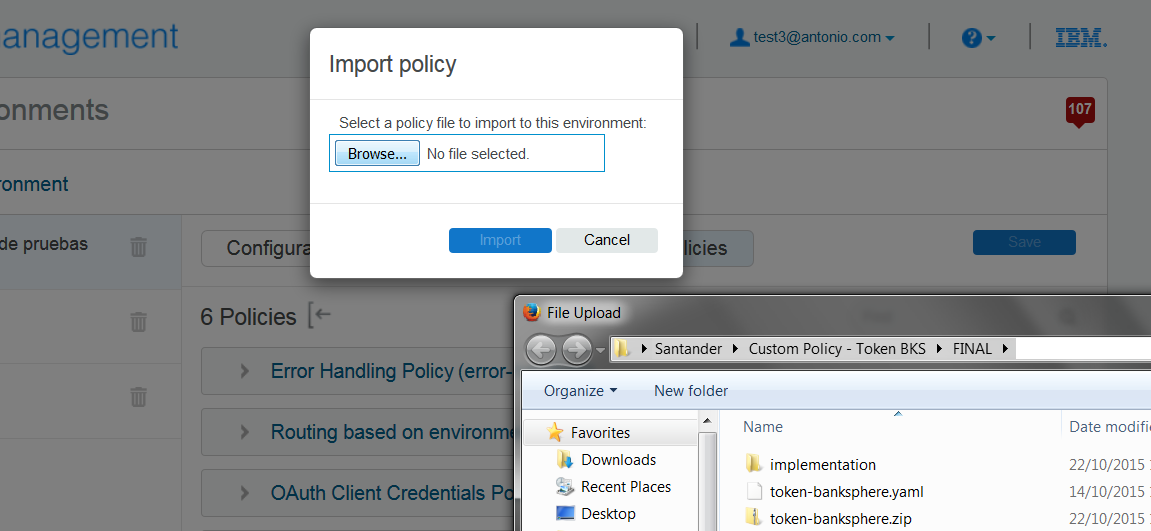
3.- Se accede el entorno donde pretenden ser subidas las políticas definidas. En el caso de tener más de un entorno y se quiera realizar un pase entre entornos, habrá que subir las políticas a todos los entornos intervinientes.



4.- Se accede a la sección referida con el nombre de "Policies" o Politicas. Y se pulsa sobre el botón resaltado mediante un círculo rojo en la siguiente imagen para subir las politicas desarrolladas y descritas en el punto "2.1.Tipos de custom policies y propiedades":

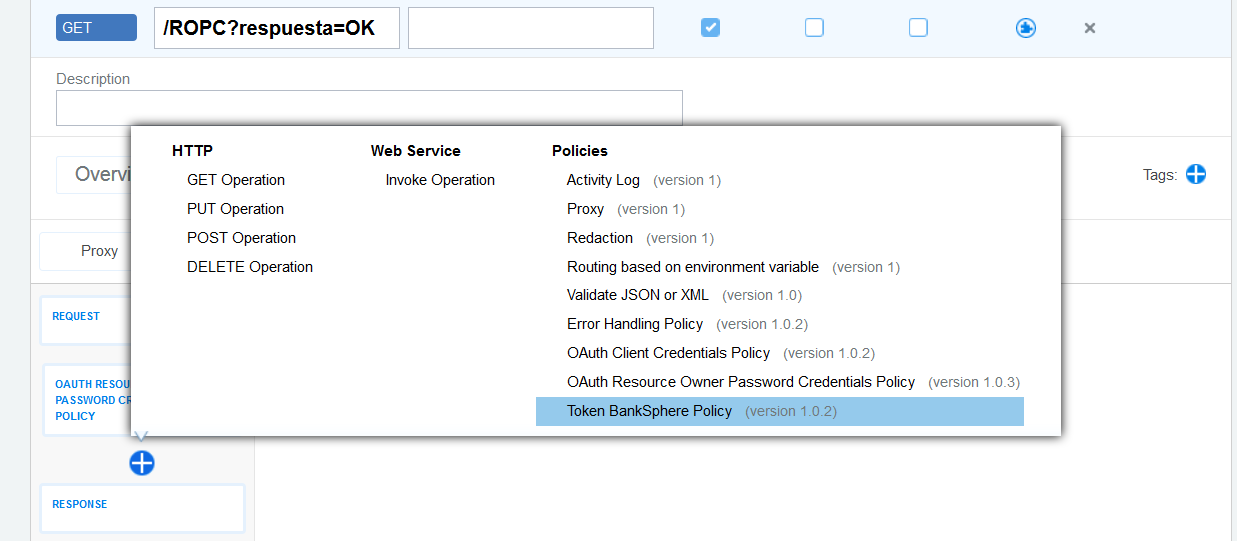


5.- Se procede a subir la política creada en formato ZIP, tal como se describe en el punto "2.2. Estructura de interna de las políticas".



6.- Se accede a la sección de implementación de las APIs para definir una API nueva o seleccionar una ya implementada. Dentro del API se accede a un recurso que ya creado o se crea uno nuevo de tipo "assembly", para poder incluir la política en su procesamiento.

Como se puede ver en la imagen, podrán ser localizadas en la sección de Policies:



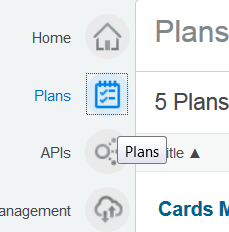
## Requisitos de funcionamiento: Plan CheckAPP

Para asegurar el correcto funcionamiento de las custom policies de OAuth, el plan y API llamado "CheckAPP" debe estar desplegado en todas las organizaciones y entornos donde vayan a ser utilizadas dichas políticas. El plan que contiene este API puede ser publicado para que no sea visible para los desarrolladores, evitando que aparezca en el portal avanzado.

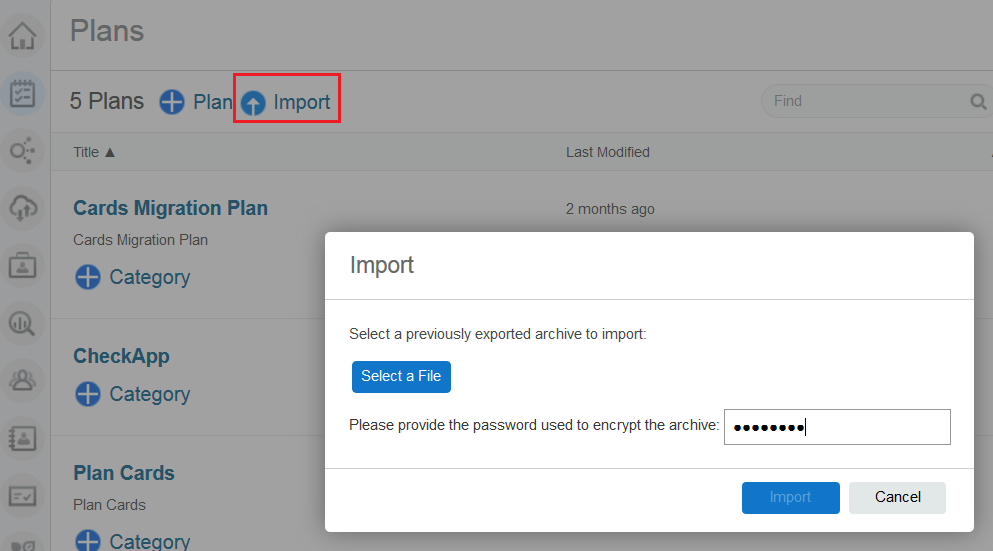
El zip que contiene el plan CheckAPP se adjunta junto a la documentación. Para importar un plan en una organización y desplegarlo en el portal siguiendo los requisitos establecidos será necesario realizar los siguientes pasos:

1.- Acceder al portal web del API Manager: https://islmnagd01.scisb.isban.corp/apimanager

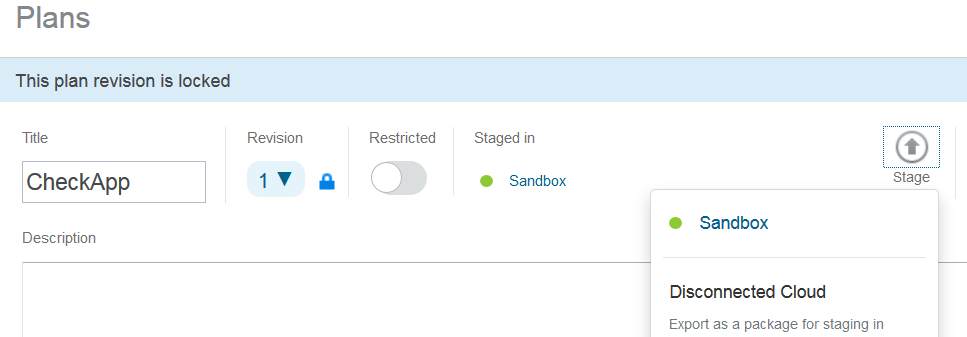
2.- Acceder a la sección de "Plans":



3.- Se selecciona la opción: "Importar". Al hacer clic en el botón, se podrá importar el plan que se adjunta indicando la contraseña: Passw0rd.

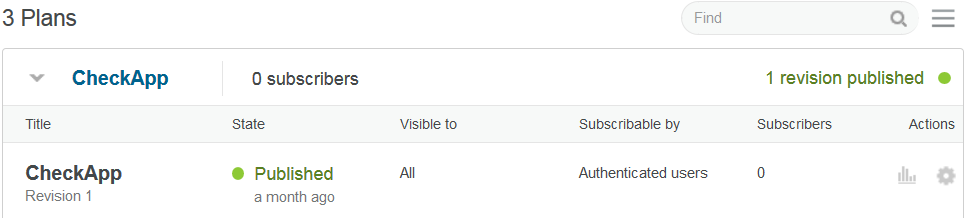


4.- Una vez el plan ha sido subido será necesario desplegarlo en todos los entornos de la organización donde vayan a utilizarse las custom policies de OAuth definidas. Para publicar el plan será necesario acceder al plan y seleccionar la opción "Stage" para todos aquellos entornos donde se quiera desplegar el plan:

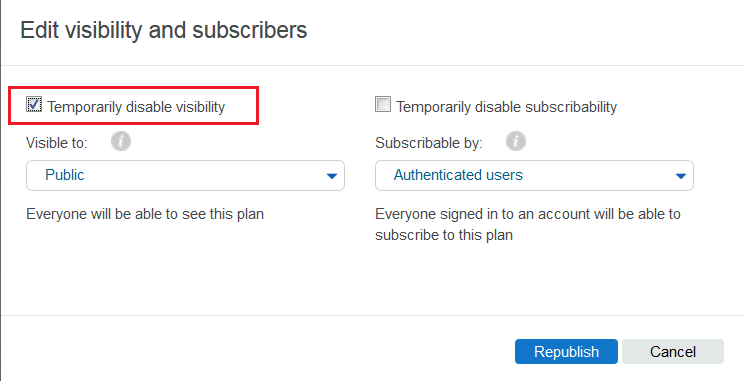


5.- Finalmente, se accede a la sección de administración para comprobar el estado del plan y editar el plan para que sea invisible pero accesible para todos los usuarios.

Inicialmente se publica estableciendo la visibilidad para todos:



Y después, se establece temporalmente como invisible y se publica:



**Es fundamental que todos estos pasos sean seguidos para asegurar el correcto funcionamiento de los servicios de generación y validación de tokens OAuth que han sido implementados.**

# Anexos

## Anexo (I): Fichero YAML de configuración

A continuación, se presenta un ejemplo del fichero YAML que ha sido integrado en la política OAuth Client Credentials definida:

policy: 1.0.0

info:

title: OAuth Client Credentials Policy

name: oauth-client-credentials

version: 1.0.2

description: Secure an API with OAuth - Client Credentials

contact:

name: Antonio Carrasco

url:

email: antonio.carrasco@es.ibm.com

attach:

- rest

- soap

properties:

$schema: "http://json-schema.org/draft-04/schema#"

type: object

properties:

scope:

label: Custom Scope

description: Define the custom scope the user has to ask for

type: string

required:

- scope

# Revisión

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisión** | | | |
| Nombre |  | | |
| Función |  | | |
| Departamento/Área |  | | |
| Resultado Revisión | *[OK/KO]* | Fecha | *[Cuándo se realiza la revisión]* |
| Motivo Rechazo | *[Detalle motivo rechazo]* | | |
| Nombre |  | | |
| Función |  | | |
| Departamento/Área |  | | |
| Resultado Revisión | *[OK/KO]* | Fecha | *[Cuándo se realiza la revisión]* |
| Motivo Rechazo | *[Detalle motivo rechazo]* | | |

# Firma

|  |  |  |  |
| --- | --- | --- | --- |
| **Firma** | | | |
| Nombre |  | | |
| Función |  | | |
| Departamento/Área |  | | |
| Firma | *[OK/KO]* | Fecha | *[Cuándo se realiza la firma]* |
| Motivo Rechazo | *[Detalle motivo rechazo]* | | |
| Nombre |  | | |
| Función |  | | |
| Departamento/Área |  | | |
| Firma | *[OK/KO]* | Fecha | *[Cuándo se realiza la firma]* |
| Motivo Rechazo | *[Detalle motivo rechazo]* | | |
| Nombre |  | | |
| Función |  | | |
| Departamento/Área |  | | |
| Resultado Revisión | *[OK/KO]* | Fecha | *[Cuándo se realiza la firma]* |
| Motivo Rechazo | *[Detalle motivo rechazo]* | | |