io-base
value-added data

TERĒGA
SOLUTIONS

Application on-top de IO-base

Guide Développeur

Sommaire

1. Introduction

L'accès aux APIs IO-base est sécurisé à l'aide du protocole Oauth2. Ceci permet la mise en place d'une délégation d'autorisation pour accorder à une application "**on-top**" un accès aux APIs IO-base.

Lors du développement de votre application spécifique "on-top" de IO-base, c'est-à-dire utilisant les APIs fournies par IO-base pour vos propres besoins, vous devez passer par une étape d'authentification avant de pouvoir appeler les APIs.

En effet, le fonctionnement général d'une API sécurisée par OAuth2 est le suivant :
1 - Authentification :

- L'application cliente demande au serveur d'autorisation un jeton (=Access Token) en échange d'informations d'authentification.
- Le serveur d'autorisation vérifie ces informations et délivre à l'application cliente un Access Token qui servira de preuve d'authentification.

2 - Consommation de la ressource (une fois que l'application cliente a obtenu son Access Token) :

- Dans une autre requête, l'application transmet l'Access Token au serveur de ressources.
- Le serveur de ressources vérifie que l'Access Token est valide et que ses privilèges sont suffisants pour accéder à la ressource.
- Le serveur de ressources envoie les données de la ressource à l'application cliente.

Oauth2 prévoit plusieurs modes d'authentification. **Nous préconisons l'utilisation du Device Access Proxy (DAP)**.

2. Device Access Proxy

2.1 Introduction

2.1.1 Objectif du DAP

L'objectif du Device Access Proxy (DAP) est de sécuriser les communications entre les équipements ou services tiers et IO-base.
Le système s'appuie sur la distribution de tokens d'authentification permettant de s'assurer que seuls les équipements préalablement approuvés, et donc considérés comme fiables, peuvent communiquer avec **Io-base**.

2.1.2 Les comptes de service

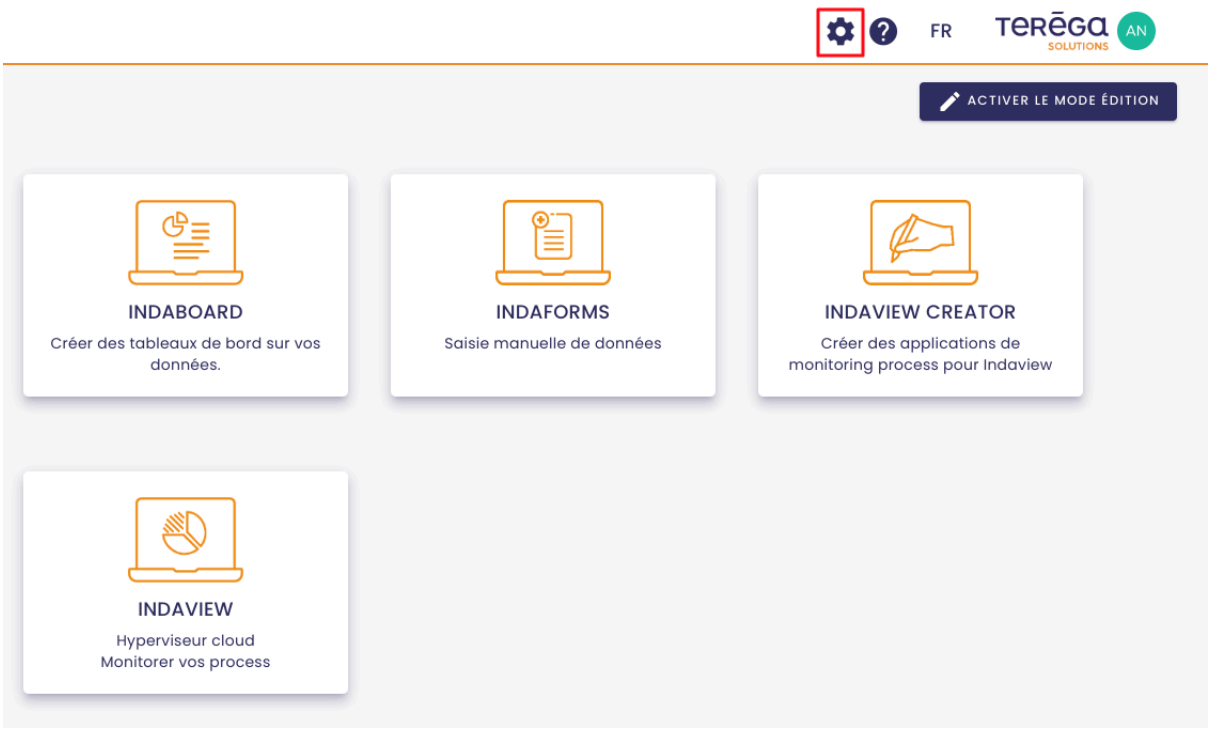
Lors de la demande de tokens, l'équipement (ou service) va utiliser un **compte de service** (voir section **3 : Validation de l'équipement dans le portail de Io-base**).
Il existe un compte de service générique "default", qui devrait de préférence être utilisé pour les tests.
Avant d'entamer la procédure, il est recommandé de demander un compte de service dédié à l'utilisation de l'équipement (ou service) pour avoir une meilleure maîtrise des droits d'accès sur Io-base.
Pour effectuer une demande de compte de service DAP, contactez le support en précisant :

- le nom du compte de service
- les droits nécessaires : read, write ou read/write

Ainsi, si besoin, il sera possible de restreindre les droits aux seules métriques nécessaires.

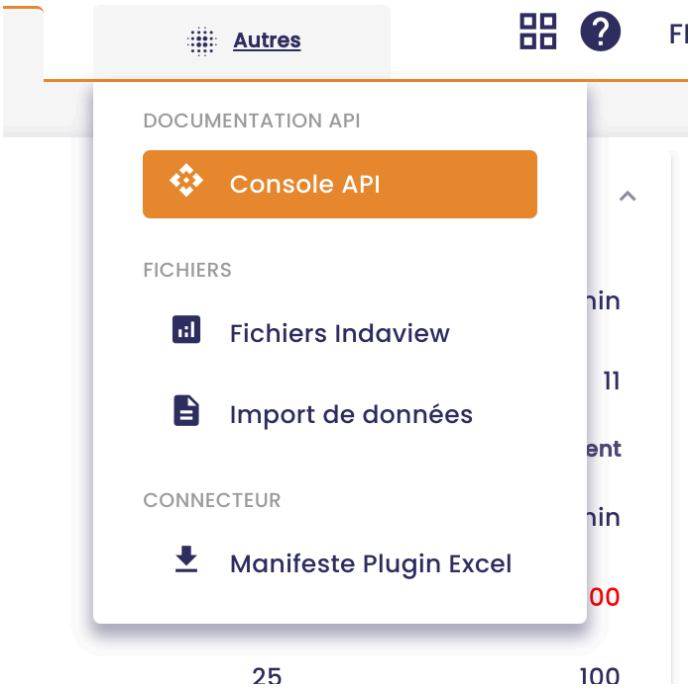
2.2 : Accès à l'interface de requêtes de l'API

Pour accéder à l'interface de requêtes de l'API, se connecter à Io-base et cliquer sur le bouton en forme de roue crantée en haut à droite de l'écran :

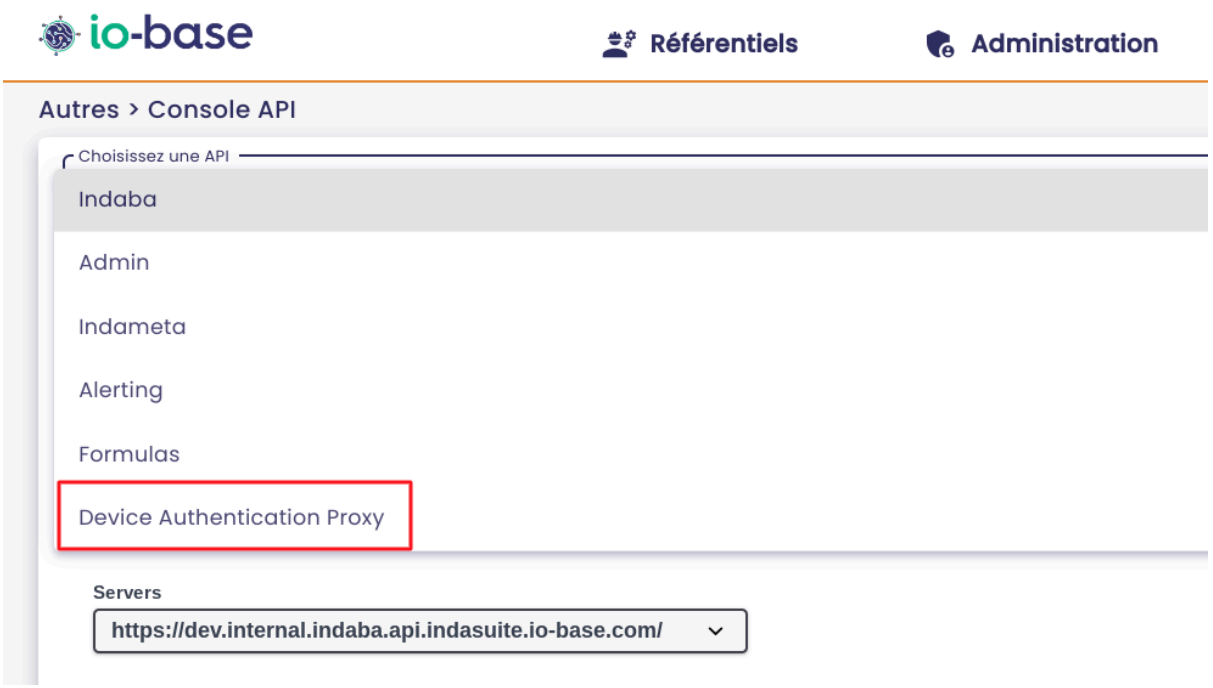


La page d'administration de Io-base s'ouvre.

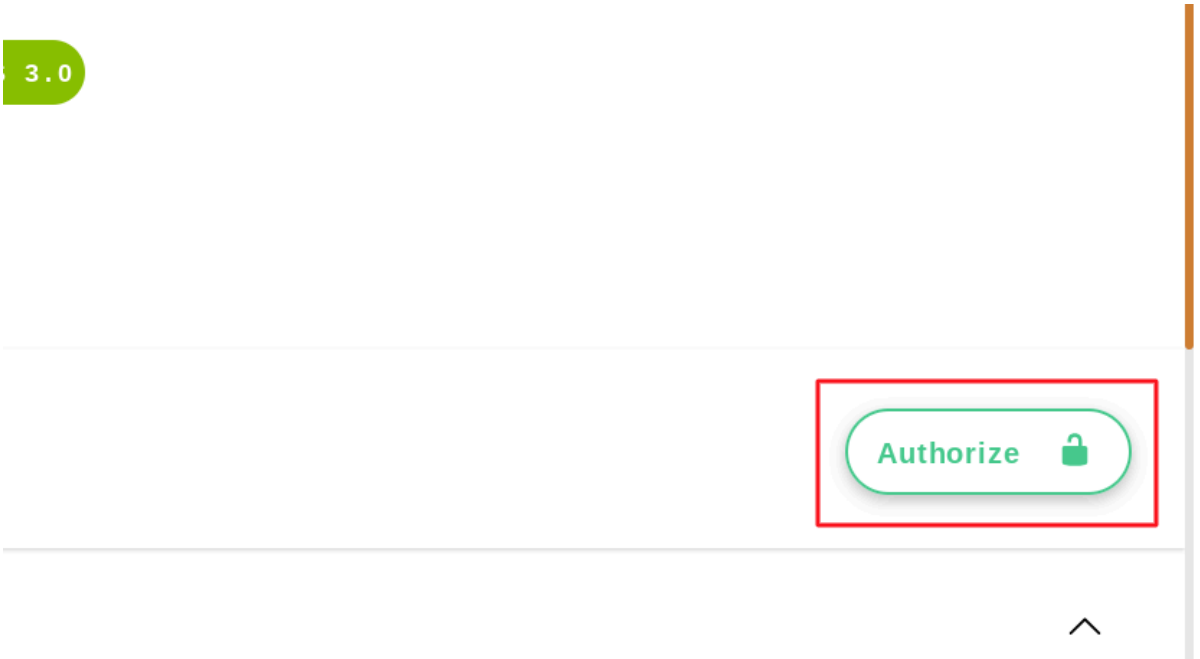
Aller au menu **Autres/Console API** :



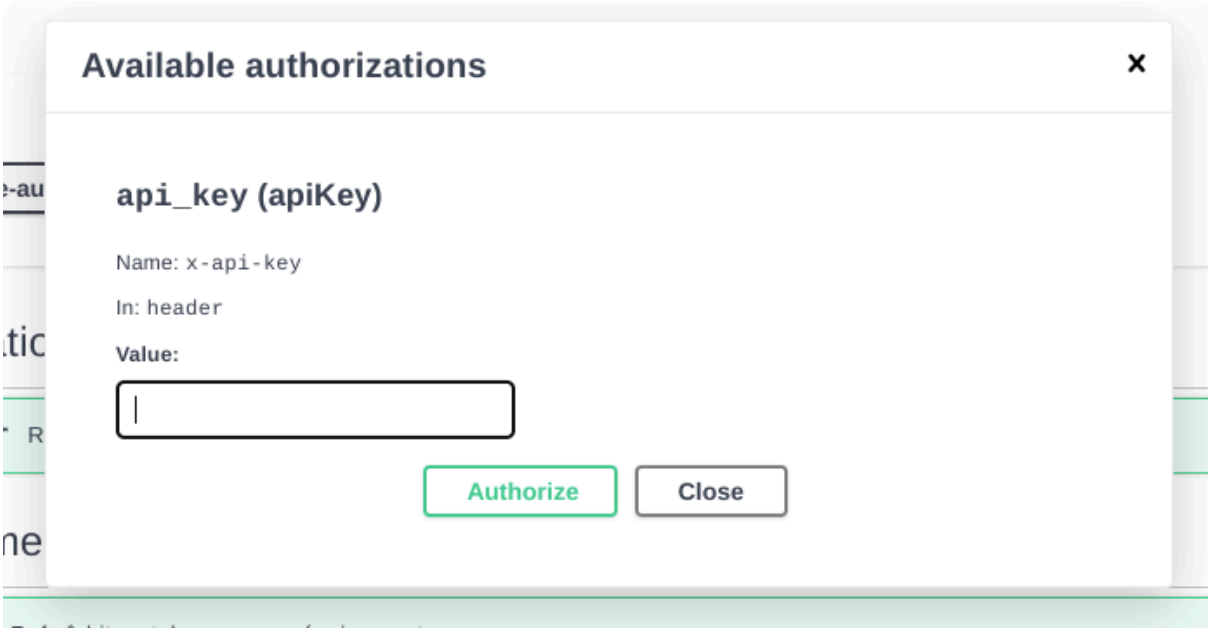
Ensuite, sélectionnez l'API **Device Authentication Proxy**, à l'aide de la liste déroulante en haut à gauche de l'écran.



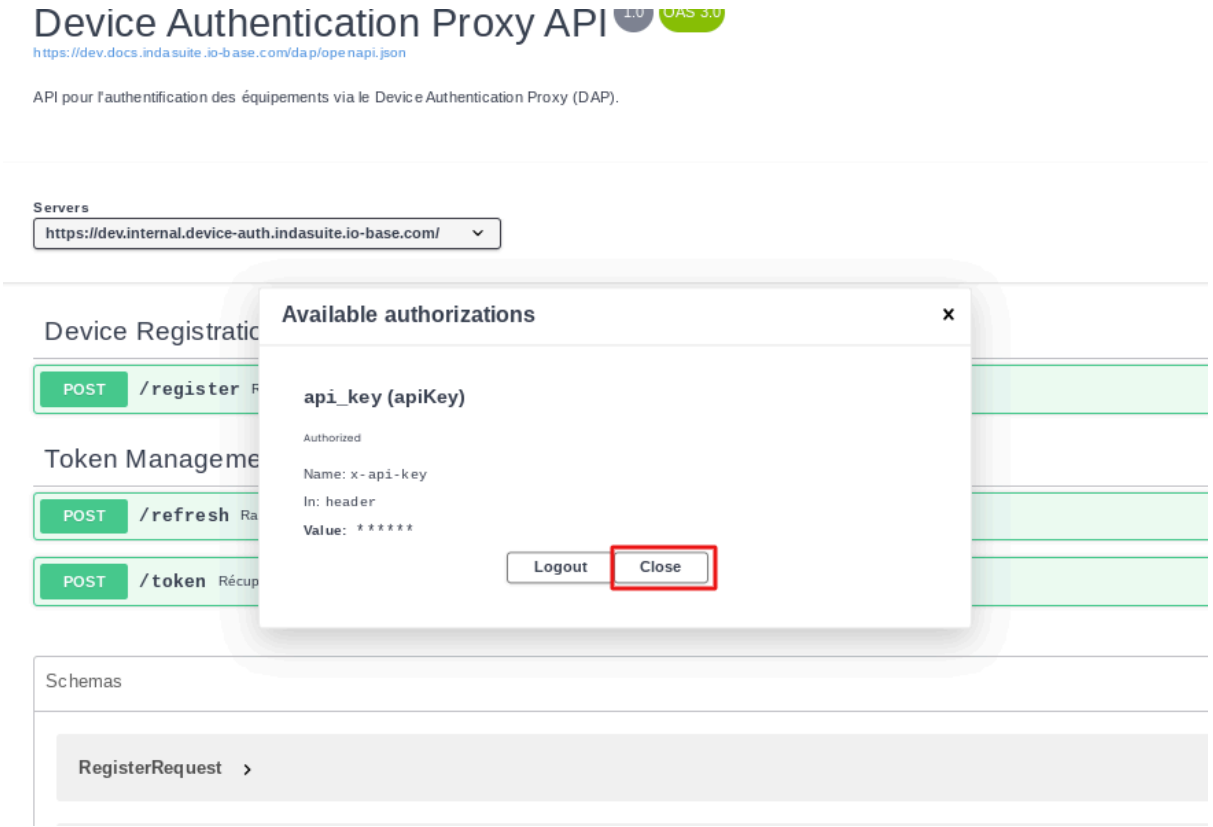
Cliquez ensuite sur le bouton **Authorize**, à droite de l'écran.



Saisissez la clé API demandée (contactez votre administrateur si vous ne l'avez pas), puis cliquez sur **Authorize**.



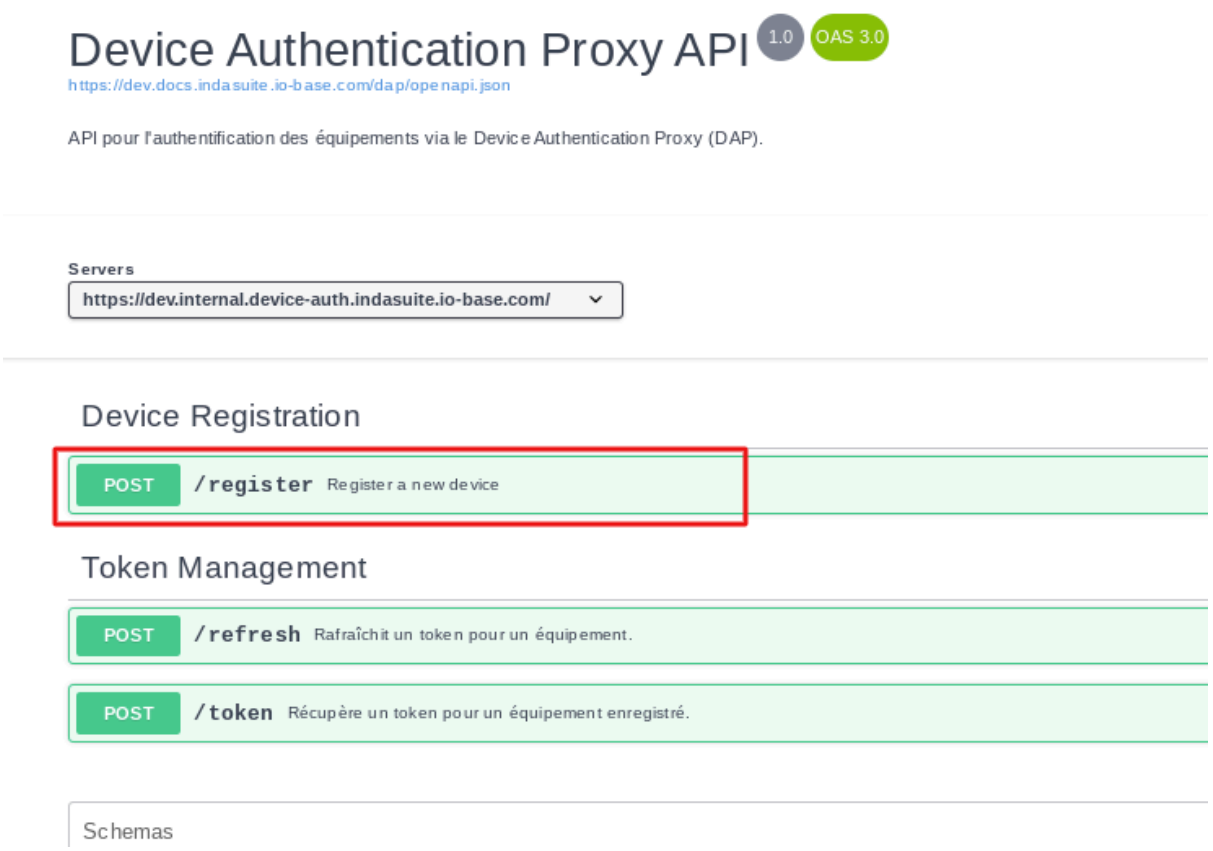
Vous pouvez maintenant fermer la fenêtre en cliquant sur **Close**.



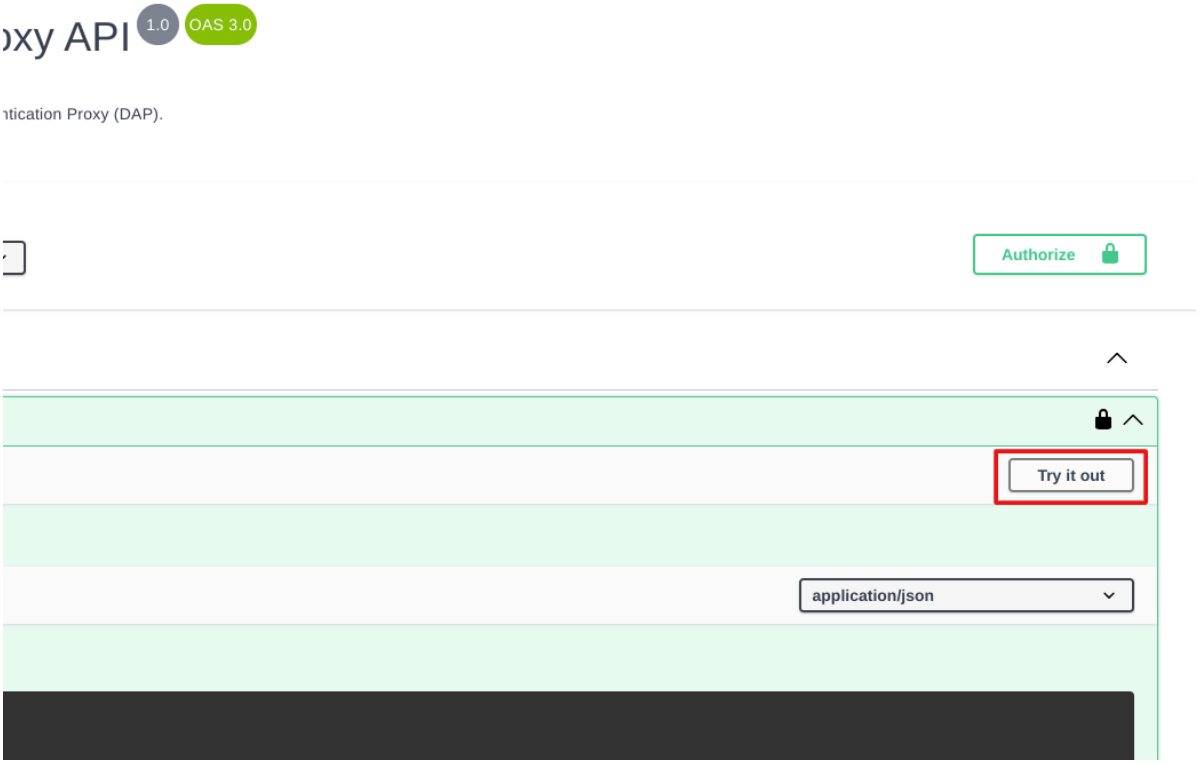
2.3 : Enregistrement de l'équipement

Vous allez maintenant pouvoir effectuer l'enregistrement du nouvel équipement.

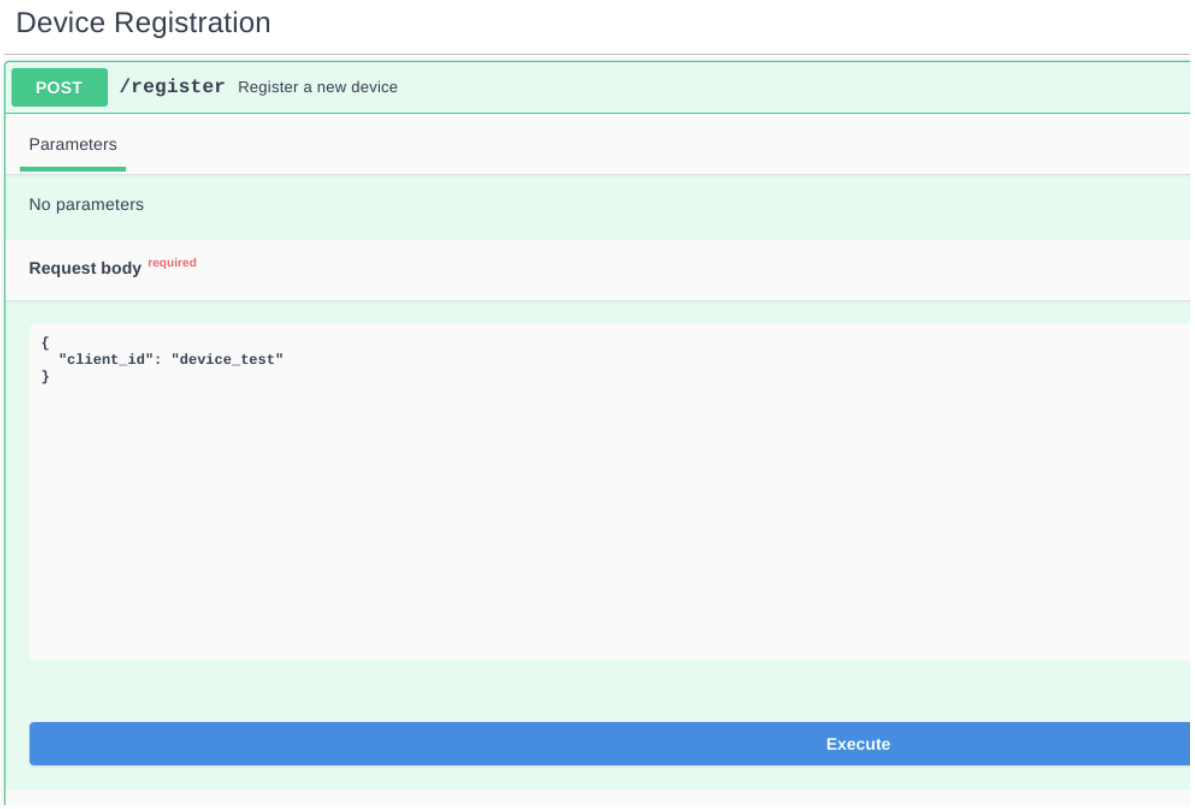
Pour ce faire, allez à la section **Device Registration**, puis cliquez sur la rubrique **POST / register**.



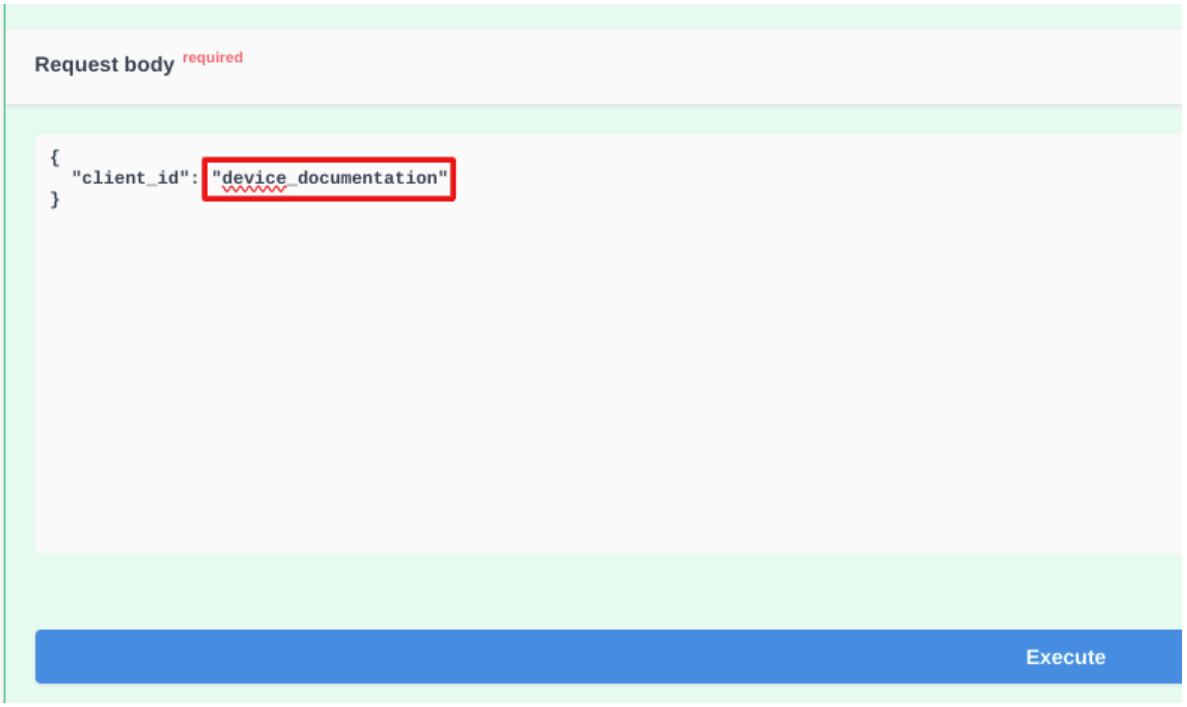
Cliquez ensuite sur le bouton "**Try it out**".



L'écran ci-dessous apparaît :



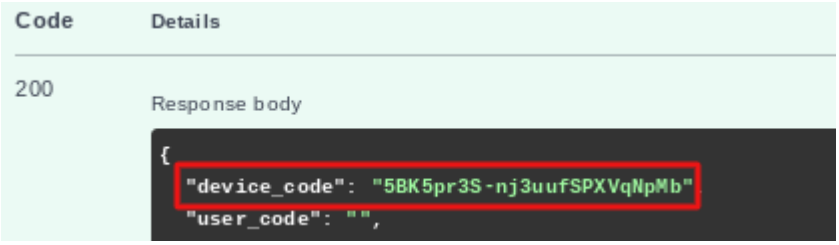
Complétez la requête en renseignant un nom pour identifier votre appareil, dans notre exemple, on l'appellera “device_documentation”.
Puis cliquez sur **Execute**.



Une réponse avec un Code 200 apparaît : l’équipement a été enregistré avec succès.

Dans le corps de cette réponse, conservez le paramètre “**device_code**”, il vous sera utile lors de la récupération du token d’accès (voir section **4 : Récupération**

du token d'accès).

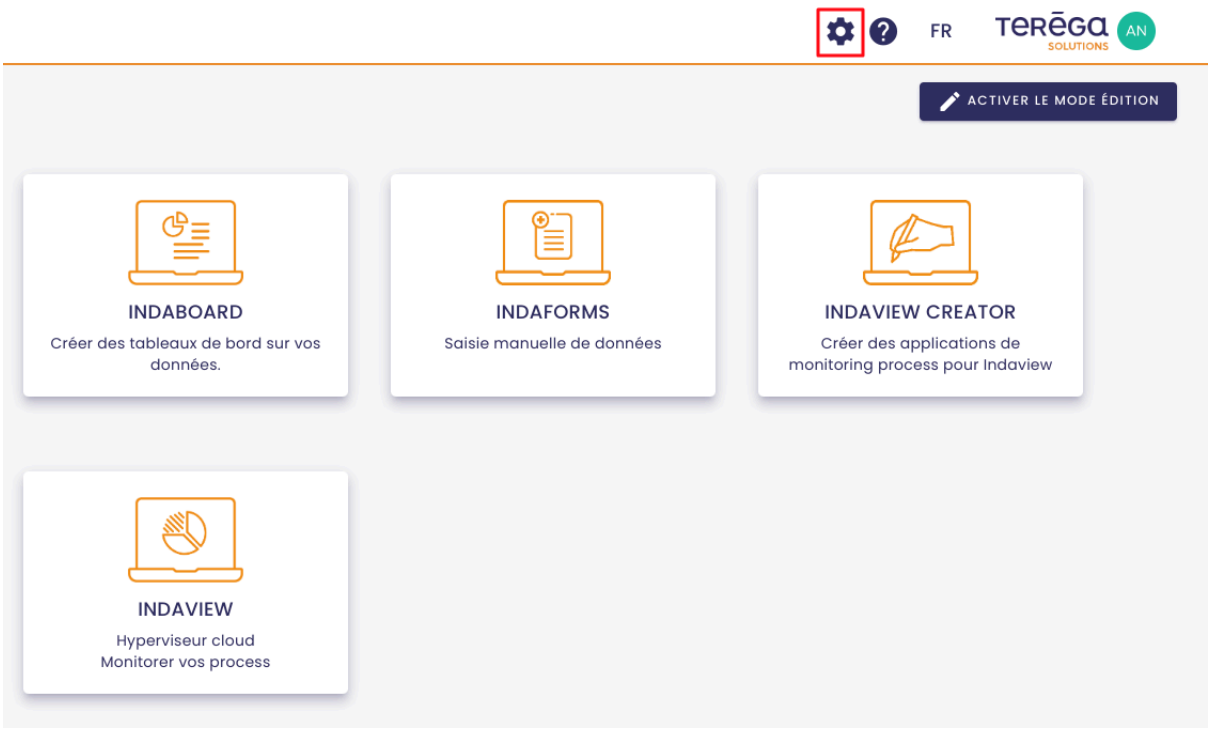


Remarque : Si un équipement portant ce nom a déjà été enregistré, l'enregistrement n'est pas possible et vous aurez une réponse avec un code 403 :

2.4 : Validation de l'équipement dans le portail de Io-base

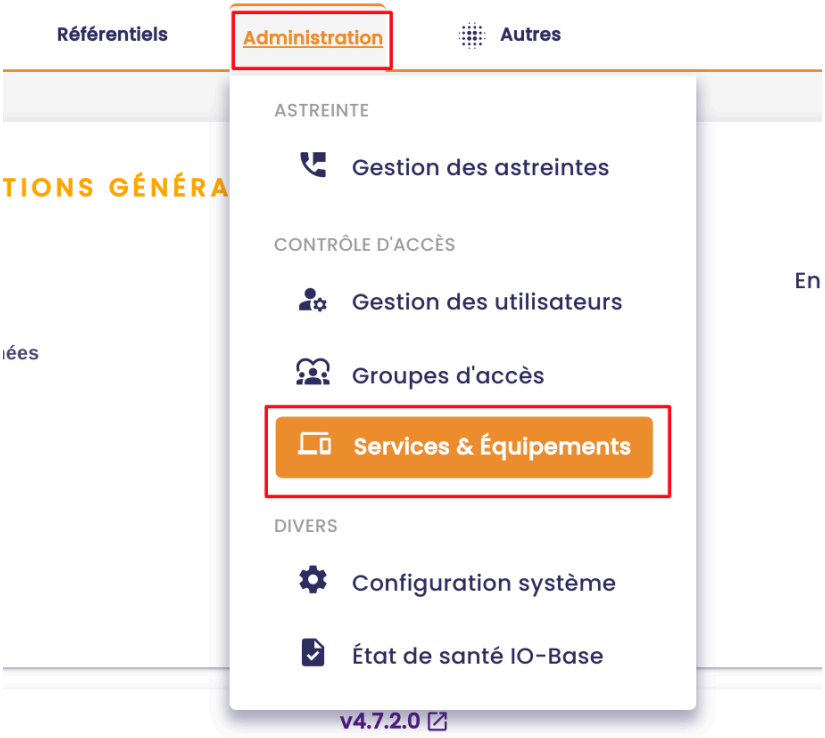
Cette action doit être effectuée par un administrateur.

Votre équipement est enregistré. Il va maintenant devoir être validé par un utilisateur ayant un rôle d'administrateur fonctionnel.
Voici la démarche à suivre pour l'administrateur :
Pour y accéder, se connecter à Io-base et cliquer sur le bouton en forme de roue crantée en haut à droite de l'écran :



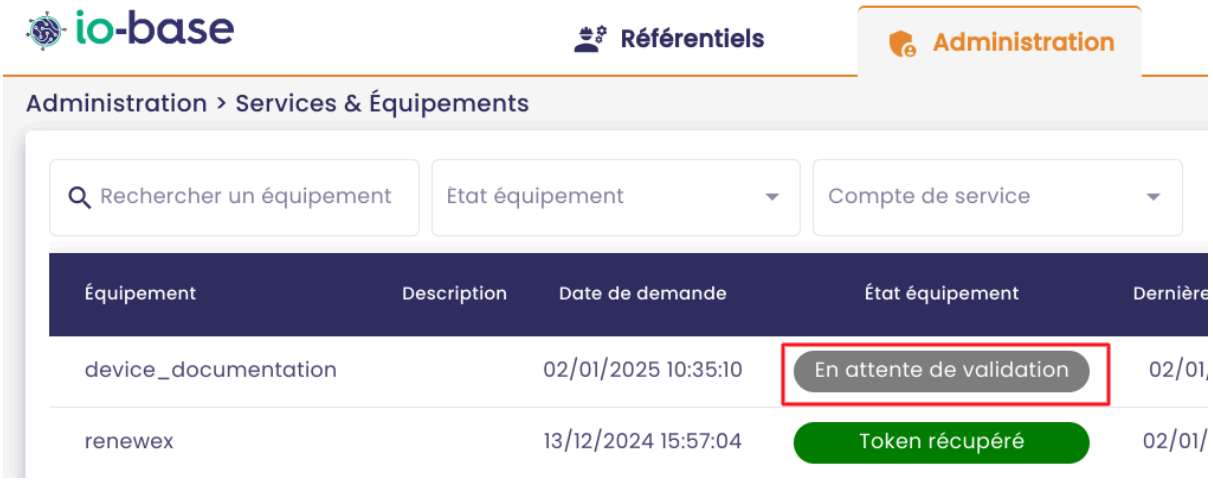
La page d'administration de Io-base s'ouvre.

Aller au menu "**Services et équipements**" :

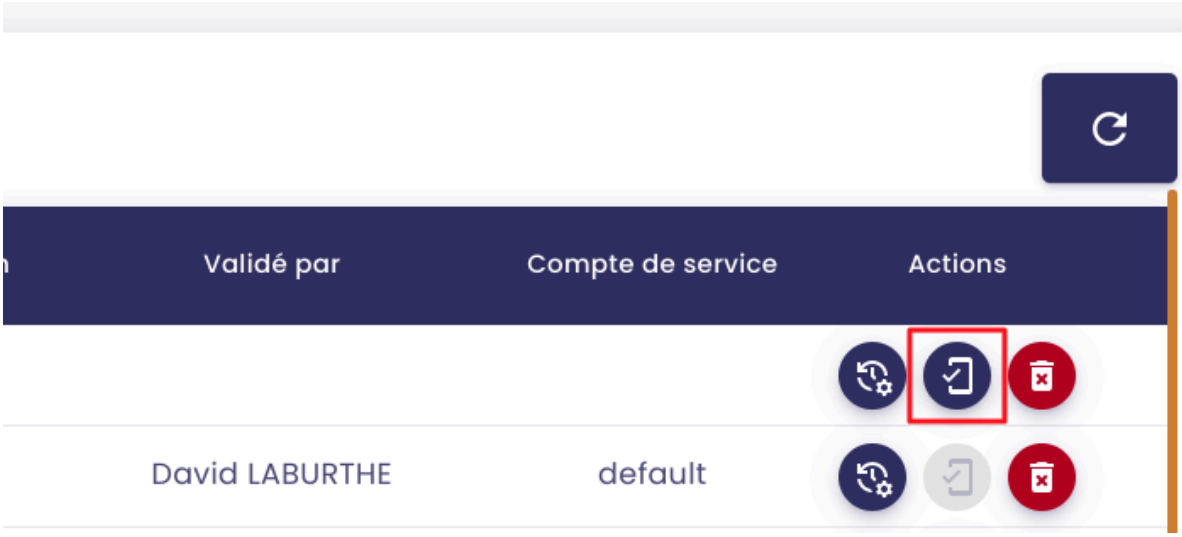


Vous êtes redirigés vers l'écran de **gestion des équipements**.

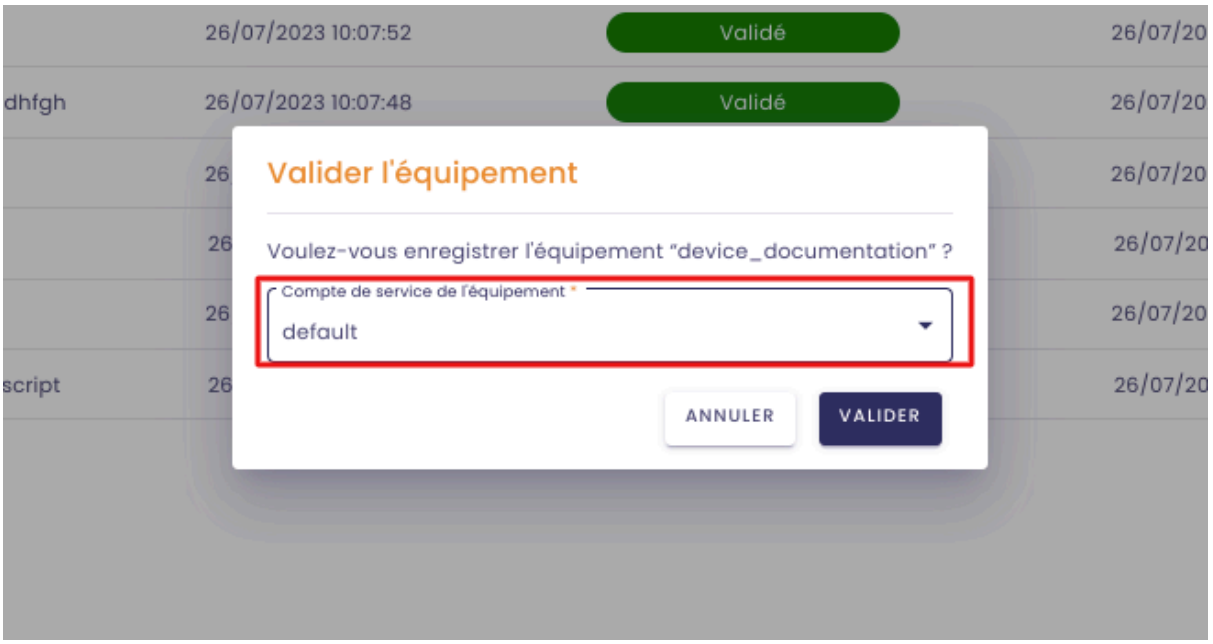
On voit que notre équipement “device_documentation” est en **Attente de validation**.



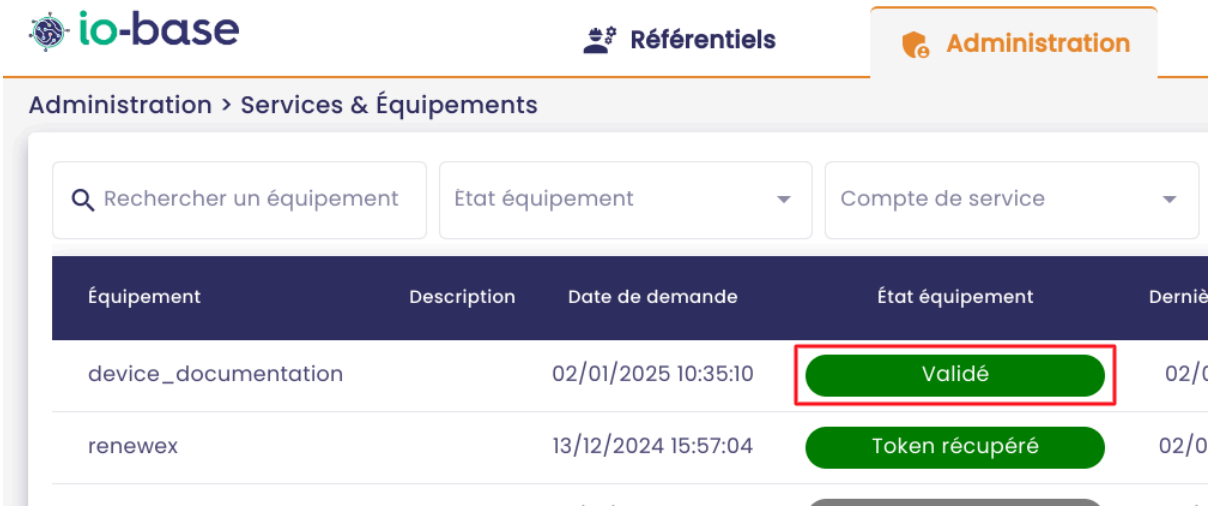
Dans la colonne **Actions**, cliquez sur le bouton **Valider l'équipement**.



Une fenêtre s'ouvre. Dans la liste déroulante, choisissez le compte de service que vous souhaitez associer à l'équipement.
Ensuite, cliquez sur **Valider**.



Remarque : Il est préférable ici d'utiliser un compte de service dédié à l'équipement plutôt que le compte de service par défaut (voir **Introduction**).
L'équipement est maintenant validé.



Pour plus d'informations sur le portail des équipements, consultez notre documentation sur la gestion des équipements dans Io-base.

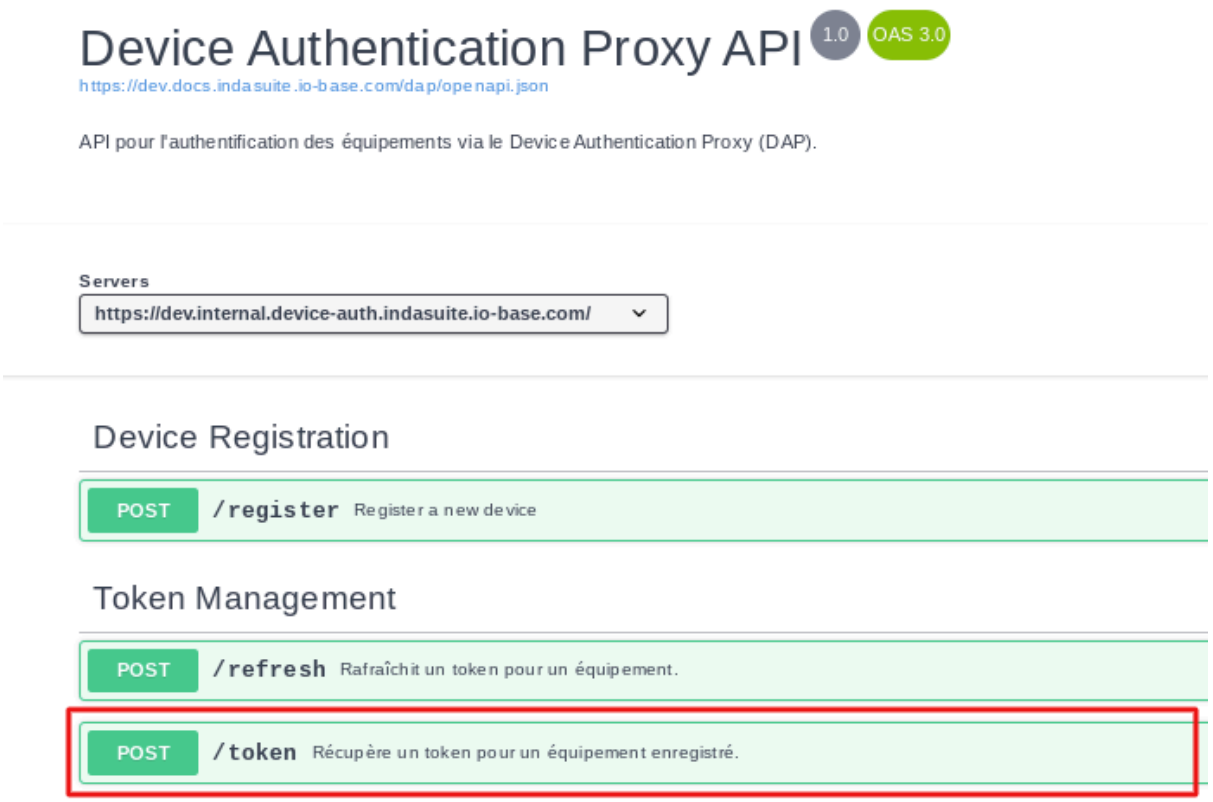
2.5 : Récupération du token d'accès

Une fois que votre équipement a été enregistré et validé par un administrateur, il va pouvoir récupérer un token d'accès, qui vous permet d'appeler les API Indaba.

Ce token a une durée de validité donc une fois expiré, il faut en demander un autre (voir section **5 : Rafraîchissement du token**).
Suivez la procédure suivante pour récupérer un token d'accès :

Accéder à nouveau à l'interface de requêtes de l'API (voir section 1: **Accès à l'interface de requêtes de l'API**).

Aller à la section **Token management,** puis cliquer sur la rubrique **Post / Token.**

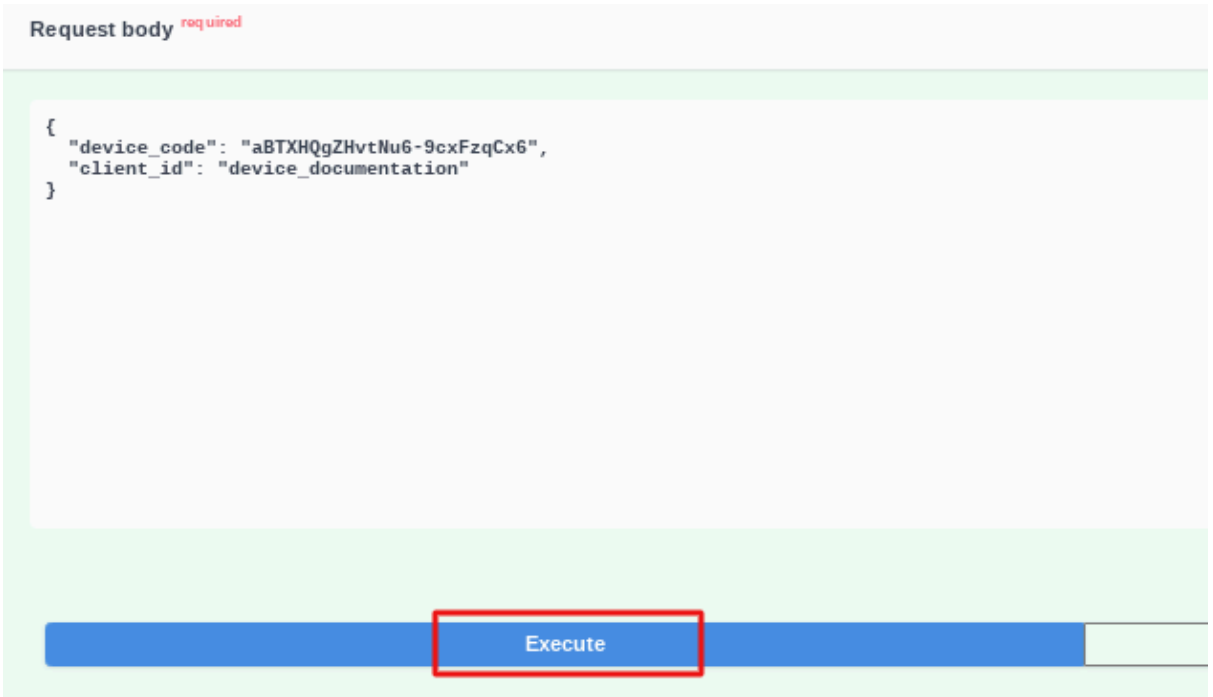


Compléter la requête en saisissant :

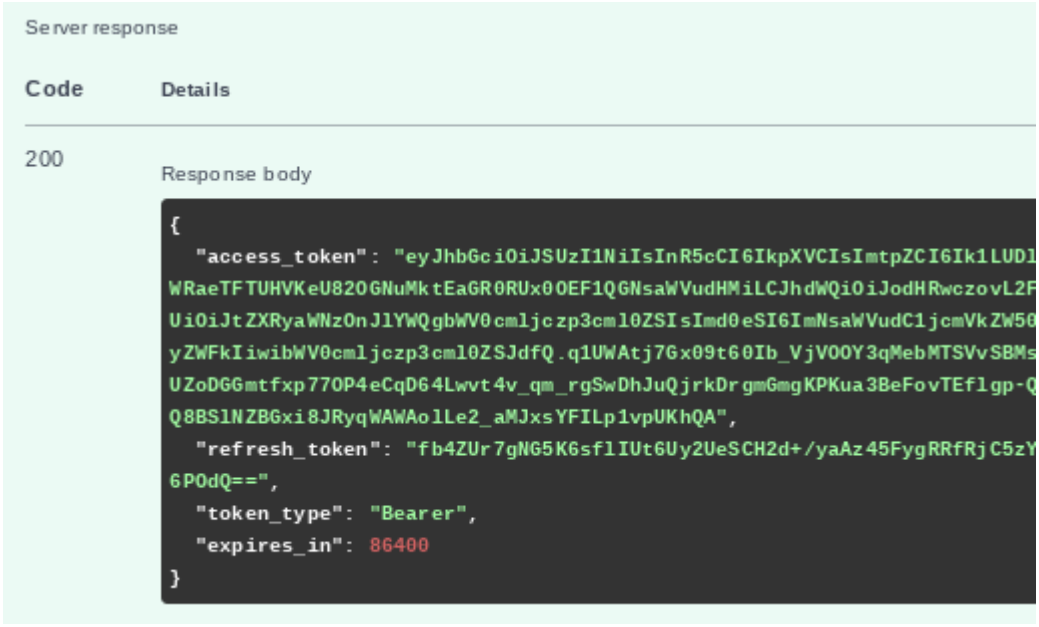
- le **device_code** : il s'agit du code envoyé dans le corps de la réponse reçue suite à votre requête d'enregistrement de l'équipement (voir section **2 : Enregistrement de l'équipement**)

- le **client_id** : il s'agit du nom que vous avez indiqué lors de l'enregistrement de votre équipement, dans notre exemple "device_documentation"

Cliquez ensuite sur **Execute** :

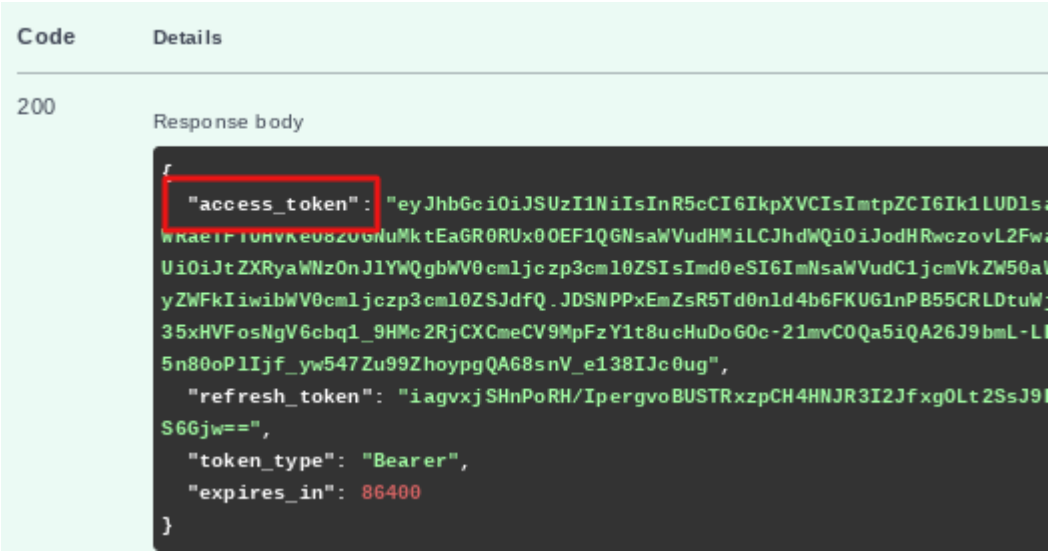


La réponse suivante apparaît :



Dans le corps de cette réponse, vous allez retrouver :

- l'**access_token**, qui permet d'appeler les API Indaba :

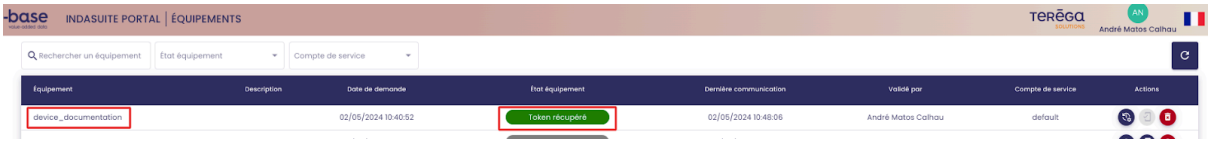


- le **refresh token**, il va vous servir à renouveler l'accès à Io-base lorsque la validité du token d'accès aura expiré.

Code	Details
200	Response body


```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI
WRaeTFTUHVKeU82OGNuMktEaGR0RUx0OEF1QGNsaWVudHMiLCJhdWQiOiJodHR
UiOiJtZXRyaWNzOnJlYWQgbWV0cmljczp3cml0ZSIsImd0eSI6ImNsaWVudC1j
yZWFkIiwibWV0cmljczp3cml0ZSJdfQ.JDSNPPxEmZsR5Td0nld4b6FKUG1nPB
35xHVFosNgV6cbq1_9HMc2RjCXCmeCV9MpFzY1t8ucHuDoGOc-21mvCOQa5iQA
5n80oPlIjf_yw547Zu99ZhoypgQA68snV_e138IJc0ug",
  "refresh_token": "iagvxjSHnPoRH/IpergvoBUSTRxzpCH4HNJR3I2Jfx
S6Gjw==",
  "token_type": "Bearer",
  "expires_in": 86400
}
```


Remarque : dans le portail des équipements de Io-base, l'état de l'équipement passe en "**Token récupéré**".

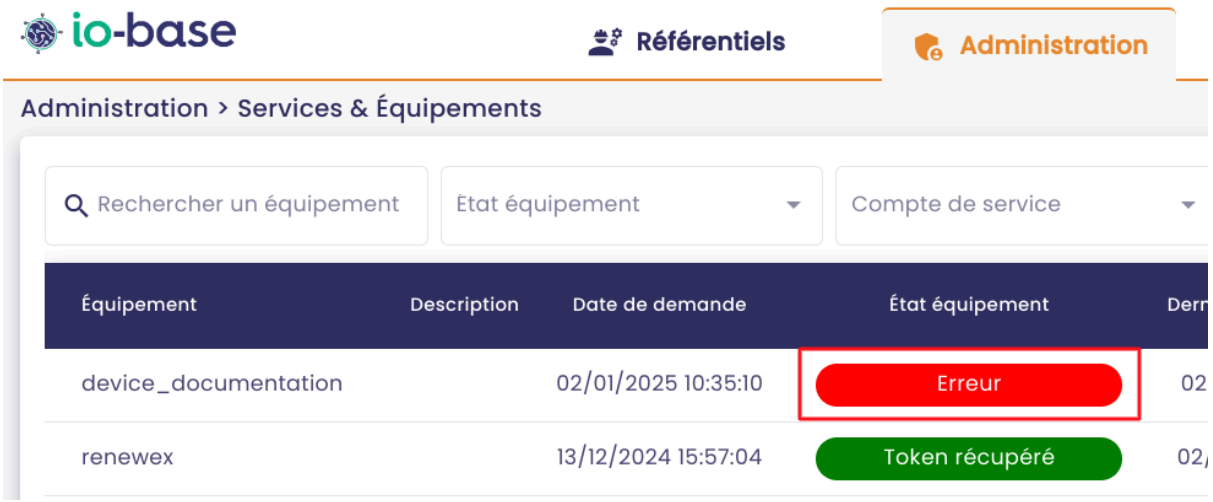


Attention, pour des raisons de sécurité, l'appel à token n'est valide qu'une seule fois !

En cas d'appel à token après récupération, l'équipement se met en erreur.

Code	Details
403	Error: response status is 403 Response body


```
{
  "error": "authorization_failed",
  "error_description": "The token was already retrieved, the device is now in error"
}
```



Dans ce cas, l'équipement doit être supprimé par un administrateur dans **Io-base**, et il faut recommencer la procédure.

Remarque : Si l'équipement n'a pas encore été validé par un administrateur, le token d'accès ne pourra pas être récupéré.
On aura une réponse avec un code 425 : "Authorization Pending".

Server response

Code	Details
425	Error: response status is 425 Response body

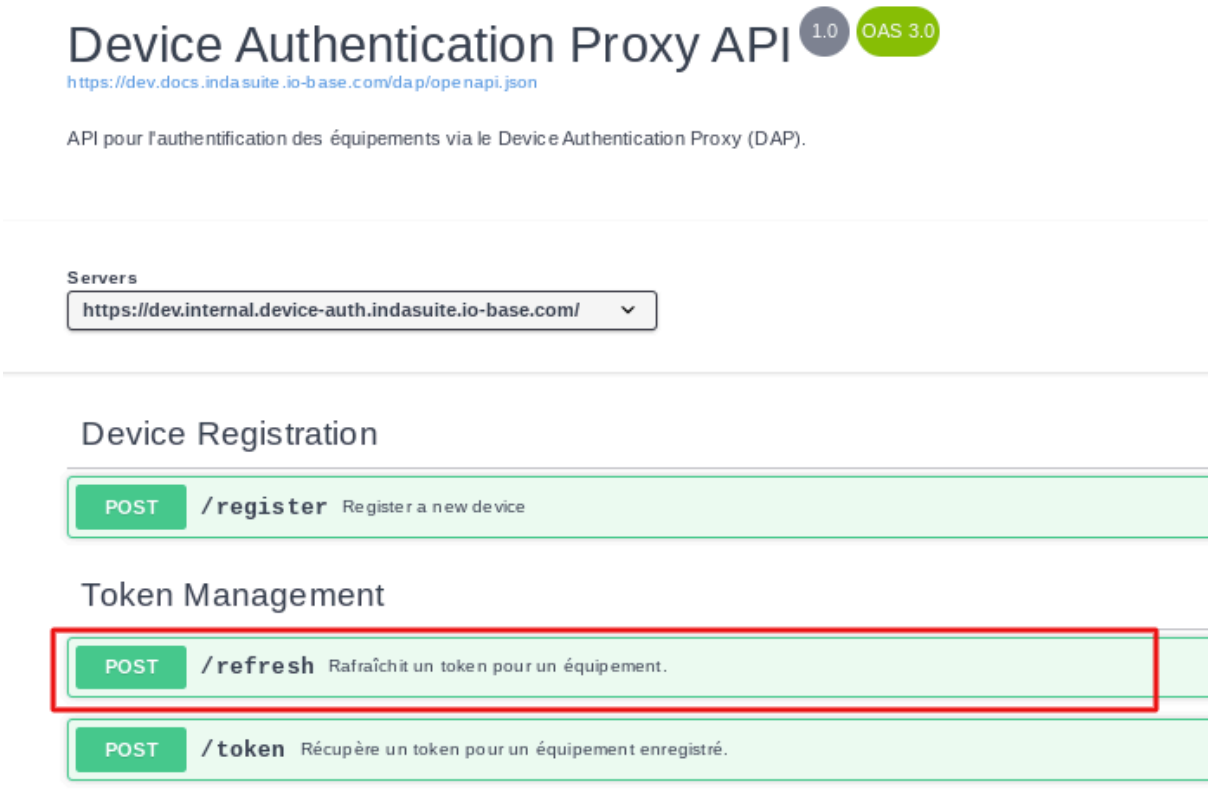
```
{
  "error": "authorization_pending",
  "error_description": "User has yet to authorize device code."
}
```

Assurez-vous que l'équipement soit validé puis effectuez l'opération à nouveau.

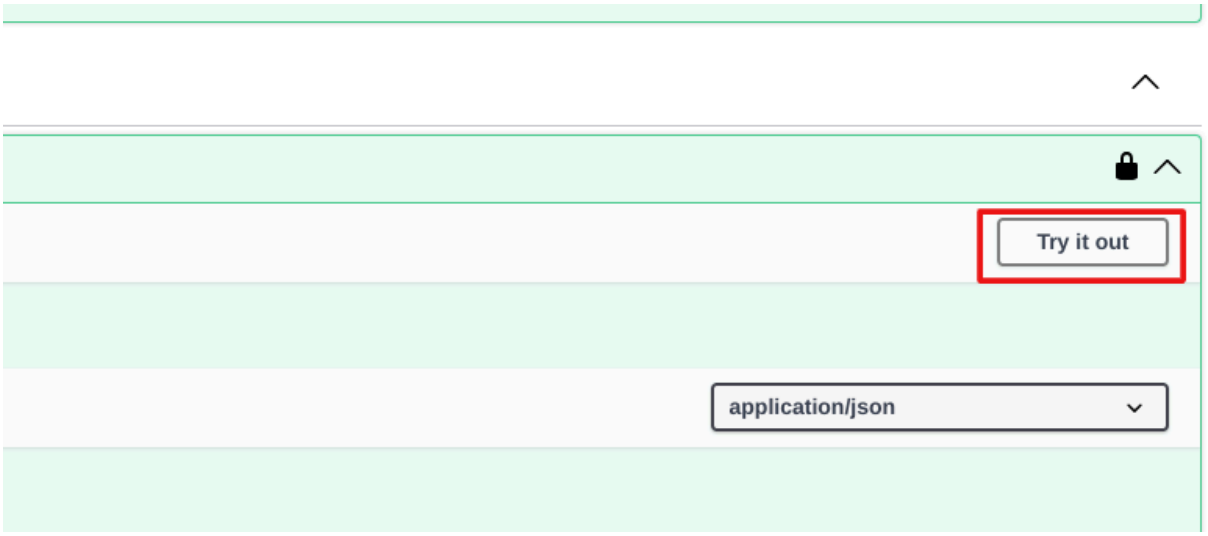
2.6 : Rafraîchissement du token

Le token d'accès attribué à l'équipement a une durée de validité limitée. Ainsi, pour continuer à communiquer avec Io-base, il va falloir renouveler ce token.
Pour ce faire, retournez à l'interface de requête API (voir section **1 : Accès à l'interface de requêtes de l'API**).

Dans la section Token Management, cliquez sur la rubrique **Post/refresh**.



Puis, **Try it out**.



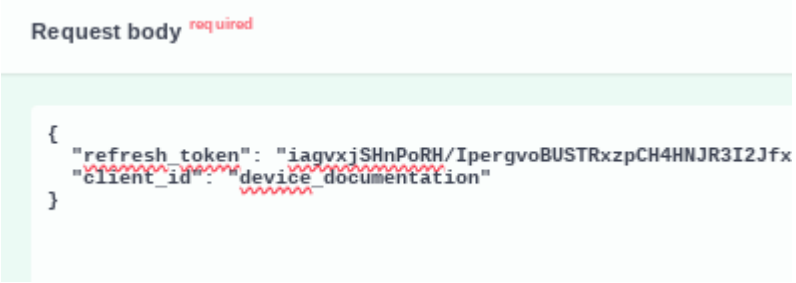
On arrive sur l'écran suivant :

Compléter la requête en saisissant :

- le "**refresh_token**" : que vous trouverez dans le corps de la réponse d'attribution du token précédent. Dans notre cas, ici :

- le "**client_id**" : le nom que vous avez saisi lors de l'enregistrement de l'équipement, dans notre cas "device_documentation" (voir section **2 : Enregistrement de l'équipement**)

On aura donc :



Cliquez ensuite sur **Execute**.

Le token a bien été rafraîchi et la communication entre l'équipement et Io-base est assurée (réponse avec un code 200) :

Server response

Code	Details
200	Response body


```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik1LU
WRaeTFTUHVKeU82OGNuMktEaGR0RUx0OEF1QGNsaWVudHMiLCJhdWQiOiJodHRwczovL
UiOiJtZXRyaWNzOnJlYWQgbWV0cmljczp3cml0ZSIsImd0eSI6ImNsaWVudC1jcmVkZW
yZWFkIiwibWV0cmljczp3cml0ZSJdfQ.YVyHluqDnWYobUlVtqVE08g7E6ZD7CFgsULp
wiqg28YHTauPegWiB8hOgGs3oha207mQYj3Wsullqg8GCnA_xjangdLVNHzhwPfKcgwu
vTUyz3Wr-_wX1n405NSzERAIXKrJSBtRLFr2an-o18iw",
  "refresh_token": "/SM07K6WITK0JP+fWDg2XLh3SpcMgU0SXvJ8irRop+4ptfgB
p4DCw==",
  "token_type": "Bearer",
  "expires_in": 86400
}
```


Dans le corps de la réponse, vous allez trouver :

- l'**access_token**, qui permet d'appeler les API Indaba :

200 Response body


```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik1LUDlsa081W
WRaeTFTUHVKeU82OGNuMktEaGR0RUx0OEF1QGNsaWVudHMiLCJhdWQiOiJodHRwczovL2FwaS5pb
UiOiJtZXRyaWNzOnJlYWQgbWV0cmljczp3cml0ZSIsImd0eSI6ImNsaWVudC1jcmVkZW50aWFscy
yZWFkIiwibWV0cmljczp3cml0ZSJdfQ.VbNg1nnvFuQAG4G5Jd53i2shWy3BUQWIhTxsWEbDDXPA
yjP9twfOs8GMJlZCK-ZaMe16GboF9nxzlf0cL6vzo4Y_pS9ho46QwWJBwvoDcoIXNNdyMmit6BcG
WcNK20Kixuegok_-Bsv_KXElujYyhqFQn8LR_Cd28s9A",
  "refresh_token": "zj0imKlC+/b0yZcGb+x+sELTilMX2fE8bbLFZN0ZvVQTLqquc1MTP3N1
GKHOQ==",
  "token_type": "Bearer",
  "expires_in": 86400
}
```


- le refresh token, qui vous servira à renouveler la communication entre l'équipement et **Io-base** lorsque ce nouveau token sera expiré à son tour.

200

Response body


```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik1LUDlsa081V
WRaeTFTUHVKeU82OGNuMktEaGR0RUx0OEF1QGNsaWVudHMiLCJhdWQiOiJodHRwczovL2FwaS5pb
UiOiJtZXRyaWNzOnJlYWQgbWV0cmljczp3cml0ZSIsImd0eSI6ImNsaWVudC1jcmVkZW50aWFscy
yZWFkIiwibWV0cmljczp3cml0ZSJdfQ.VbNg1nnvFuQAG4G5Jd53i2shWy3BUQWIhTxsWEbDDXPA
yjP9twfOs8GMJlZCK-ZaMe16GboF9nxzlf0cL6vzo4Y_pS9ho46QwWJBwvoDcoIXNNdyMmit6BcC
WcNK20Kixuegok_-Bsv_KXElujYyhqFQn8LR_Cd28s9A",
  "refresh_token" "zj0imKlC+/b0yZcGb+x+sELTilMX2fE8bbLFZN0ZvVQTLqquc1MTP3N1
GKHOQ==",
  "token_type": "Bearer",
  "expires_in": 86400
}
```


Attention, un “refresh_token” ne peut être utilisé qu’une seule fois !

Si le même token est utilisé deux fois, l’équipement se met en erreur.

Server response

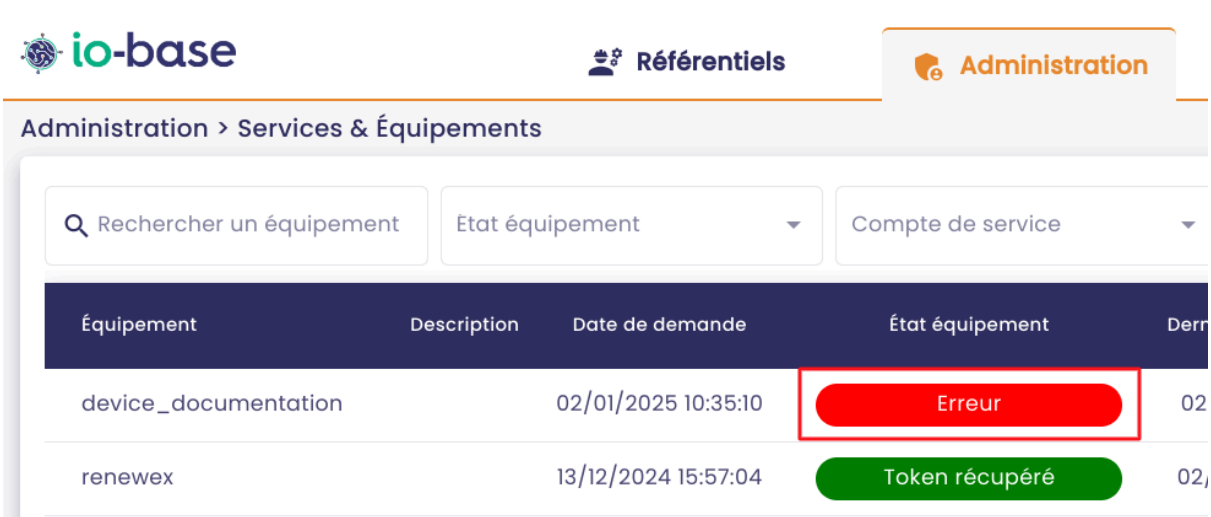
Code	Details
403	Error: response status is 403

Response body


```
{
  "error": "authorization_failed",
  "error_description": "Forbidden operation, wrong refresh token, device is now in error"
}
```


Response headers


```
content-length: 114
content-type: application/json
```

Dans ce cas, l'équipement doit être supprimé par un administrateur dans **Io-base**, et il faut recommencer la procédure.

3. Authorization Code

3.1 Prérequis

Pour implémenter ce mode d'authentification vous devez disposer de :

- A. L'url du point de terminaison pour l'authentification : **{AUTH_BASE_URL}**
- B. L'url du point de terminaison pour la récupération des tokens : **{TOKEN_ENDPOINT}**
- C. le clientId de votre client dans la solution d'authentification : **{CLIENT_ID}**
- D. le clientSecret de votre client dans la solution d'authentification : **<CLIENT_SECRET>**
- E. L'audience et le scope nécessaire pour appeler votre API : **{AUDIENCE}** et **{SCOPES}**

Pour A.B.C.D. Si vous ne disposez pas de ces informations vous pouvez les obtenir en sollicitant le support sur l'adresse support.io-base@terega.fr.
Le client envoie un courriel au service de support en fournissant des informations suivantes :
- le nom de l'application,
- la description,
- les redirect_uri : {REDIRECT_URI}
- l'url de l'application
- le mode d'authentification souhaité

- une description de l'utilisation de l'api qui va être faite
Le service de support vérifie les informations fournies par le client et génère un client_id et un client_secret.
Le service de support envoie un courriel au client avec les informations client_id et client_secret.
Le client doit conserver ces informations en sécurité, car elles ne doivent pas être partagées avec des tiers.

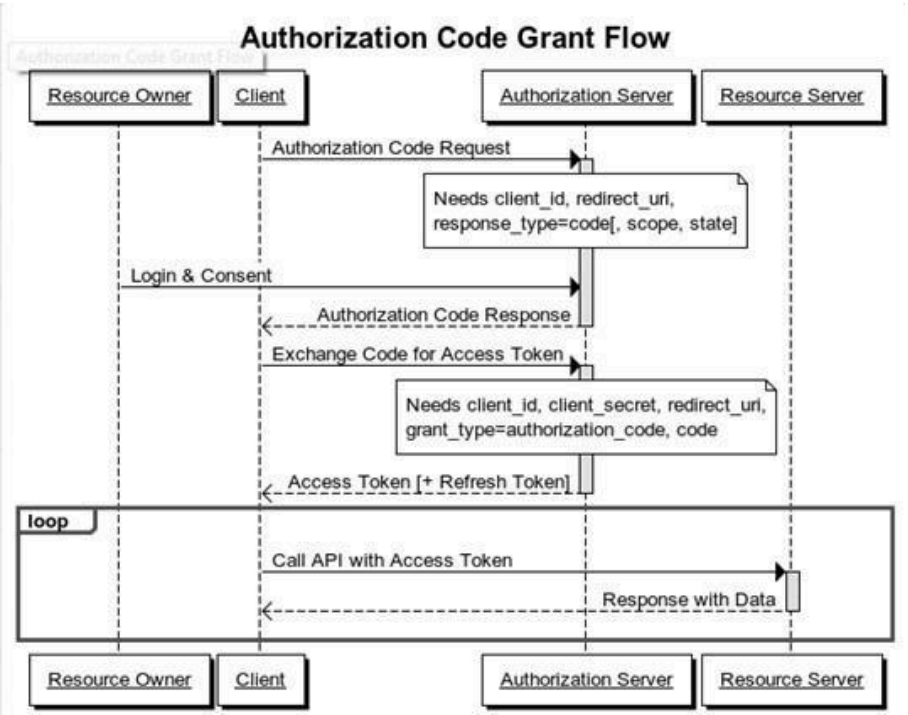
Remarque : Seules les adresses de retour fixées lors de la création du client seront acceptées par la solution d'authentification. Pour faire ajouter une nouvelle adresse vous devez faire une demande au support io-base : support.io-base@terega.fr.

Pour E. les jetons ne sont valables que pour une seule audience à la fois. Si les scopes nécessaires sont présents dans plusieurs audiences il vous faut plusieurs jetons.
La documentation des apis io-base permettent si nécessaire d'identifier les audiences et scopes nécessaires pour les besoins de votre client.

3.2 Principe

L' Authorization Code Flow permet une authentification de l'utilisateur final (ou Ressource Owner) et de l'application consommatrice (ou Client), sans communiquer les identifiants (compte utilisateur et mot de passe) de l'utilisateur final.

Ce mode d'authentification est le plus sécurisé. Il nécessite d'avoir une mire d'authentification et permet de bloquer un utilisateur en cas d'accès frauduleux.



3.3. Définitions

3.3.1 Url de redirection

Lorsqu'un utilisateur accède à une application qui demande l'accès à ses données protégées par une API, le protocole OAuth2 nécessite que l'application fournisse une URL de redirection (REDIRECT_URI) pour recevoir un code d'autorisation après que l'utilisateur ait donné son consentement.

Après avoir autorisé l'application, l'utilisateur est redirigé vers une page d'autorisation où il peut décider de donner ou non l'accès à l'application. Si l'utilisateur accorde l'accès, l'API génère un code d'autorisation qui est envoyé à l'URL de redirection spécifiée par l'application.
Cela permet de garantir que l'application qui demande l'autorisation est bien celle qui a été enregistrée auprès de l'API, et que le code d'autorisation est envoyé à l'application correcte. De plus, cela permet à l'application de récupérer de manière sécurisée le code d'autorisation et de l'utiliser pour obtenir un jeton d'accès qui pourra être utilisé pour accéder aux données protégées par l'API.

Il est important pour les développeurs d'applications de comprendre comment fonctionne l'URL de redirection dans le contexte de l'authentification OAuth2, afin de garantir que leur application est correctement enregistrée auprès de l'API et que les données de l'utilisateur sont protégées de manière adéquate.

Note : nous acceptons actuellement de répondre sur une adresse de type https://*.<client>.fr, le développeur a donc deux manières de réaliser des tests en localhost :

- le développeur peut ajouter une entrée dans le fichier "hosts" de son système d'exploitation afin de faire correspondre l'url de redirection avec une adresse IP locale. Cela permet de simuler la redirection vers une page d'autorisation hébergée localement sur sa machine.

- il est également possible de créer une page de redirection personnalisée en utilisant un langage de programmation côté serveur, tels que PHP ou Node.js,

pour traiter le code d'autorisation envoyé par l'API et rediriger l'utilisateur vers la page souhaitée après avoir récupéré le jeton d'accès.

3.3.2 Scope

Le scope est un concept utilisé dans le protocole OAuth2 pour spécifier les actions et les données auxquelles une application a accès lorsqu'elle utilise un jeton d'accès. Il s'agit d'une liste de permissions qui indique à l'API ce que l'application peut faire avec les données de l'utilisateur.

Un exemple de scope est "metrics:read" et "metrics:write", qui permettent à l'application de lire ou écrire des données dans une métrique spécifique.

Avec le scope "metrics:read", l'application pourrait récupérer les données de la métrique, mais ne pourrait pas les modifier. Avec le scope "metrics:write", l'application pourrait quant à elle écrire.

Il est important de noter que l'application ne peut pas accéder à toutes les données et toutes les actions qui sont disponibles auprès de l'API, mais uniquement celles qui ont été accordées par le scope. Cela permet de limiter l'accès aux données de l'utilisateur uniquement aux fonctionnalités nécessaires pour l'application, et de protéger les données de l'utilisateur en cas d'utilisation abusive de l'application.

Les scopes que l'on peut utiliser sur l'api indaba sont les suivants :

metrics:read	Permet l'accès en lecture sur Api Indaba
metrics:write	Permet l'accès en écriture sur Api Indaba

Voici les différentes étapes à réaliser pour obtenir un token d'autorisation et faire les appels APIs.

3.4 Implémentation technique

3.4.1 Autoriser l'utilisateur et récupérer un authorization code

Requête :

La première étape consiste en un appel GET sur l'url d'autorisation suivante afin d'autoriser l'utilisateur à utiliser l'api et récupérer son authorization_code :

```
GET {AUTH_BASE_URL}/authorize?response_type=code & client_id={CLIENT_ID} &
redirect_uri={REDIRECT_URI} & scope={SCOPES}& audience={AUDIENCE}
```

Exemple de scope : scope=metrics:read%20metrics:write
Exemple de {AUTH_BASE_URL} : https://io-base.eu.auth0.com/authorize

Réponse :

Le statut de la réponse doit être 302 et elle doit contenir l'authorization_code nécessaire pour la prochaine étape :

HTTP/1.1 302 Found Location : https://{REDIRECT_URI}?code=AUTHORIZATION_CODE

3.4.2 Demander le token

La deuxième étape consiste à demander un token en utilisant l'authorization_code préalablement obtenu, il faut pour cela faire un appel POST avec les informations suivantes :

```
curl --request POST \
--url '{TOKEN_ENDPOINT}' \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=authorization_code \
```

```
--data client_id=<CLIENT_ID> \
--data client_secret=<CLIENT_SECRET> \
--data code=<AUTHORIZATION_CODE> \
--data redirect_uri=<REDIRECT_URI>
```

exemple de {TOKEN_ENDPOINT} =
https://io-base.eu.auth0.com/authorize/oauth/token

avec

<CLIENT_ID> : voir pré-requis

<CLIENT_SECRET> : voir pré-requis

<AUTHORIZATION_CODE> : récupéré en réponse de l'étape 1

<REDIRECT_URI> : voir prérequis

Réponse :
La réponse doit renvoyer un status 200 et l'access token nécessaire
pour appeler l'API

{ "access_token": "eyJz93a...k4laUWw", "refresh_token": "GEbRxBN...edjnXbL", "id_token" "eyJ0XAi...4faeEoQ", "token_type": "Bearer" }

3.4.3 Appeler l'API

L'appel à l'api peut se faire alors en passant l'access_token dans le header de la manière suivante (exemple pour un get databases) :

```
GET https://<CLIENTNAME>.indaba.api.indasuite.io-base.com/v1/databases
```

avec comme header :

```
'authorization: Bearer <ACCESS_TOKEN>'
```

avec

<CLIENTNAME> : votre nom de client

<ACCESS_TOKEN> : l'access token préalablement obtenu