Energiapro API-Client

L'API-Client Energiapro est responsable de fournir des données en ligne aux clients de la société Energiapro SA ayant souscrits à ce service.

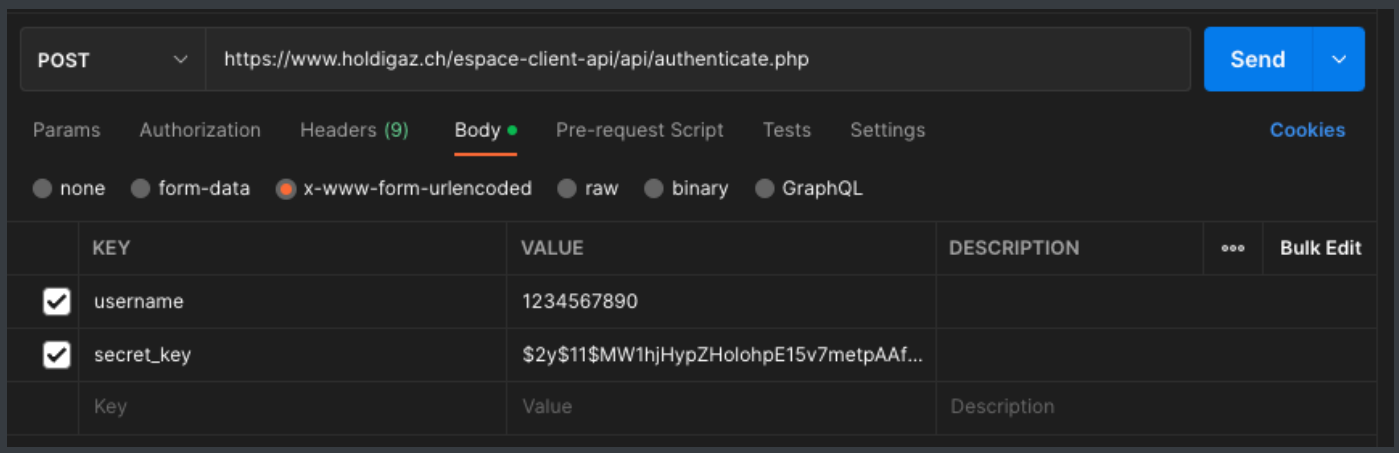
Comment utiliser l'API

Authentification

Le service d'authentification vous permet de générer un token JWT valide pendant 60 minutes et qui sera indispensable pour accéder à l'API-Client. Durant toute la durée de vie du Token, il ne sera plus nécessaire de repasser par l'authentification à chaque requête. Cependant, une fois le token expiré, vous serez obligé de vous authentifier à nouveau pour obtenir un nouveau token.

Procédure d'authentification

Les paramètres ci-dessous permettent de générer le Token JWT nécessaire pour accéder à l'API Espace-client et obtenir les données désirées.



Servers

Veuillez utiliser cettte URL : `https://www.holdigaz.ch/espace-client-api/api/authenticate.php` . L'accès en HTTPS est impératif.

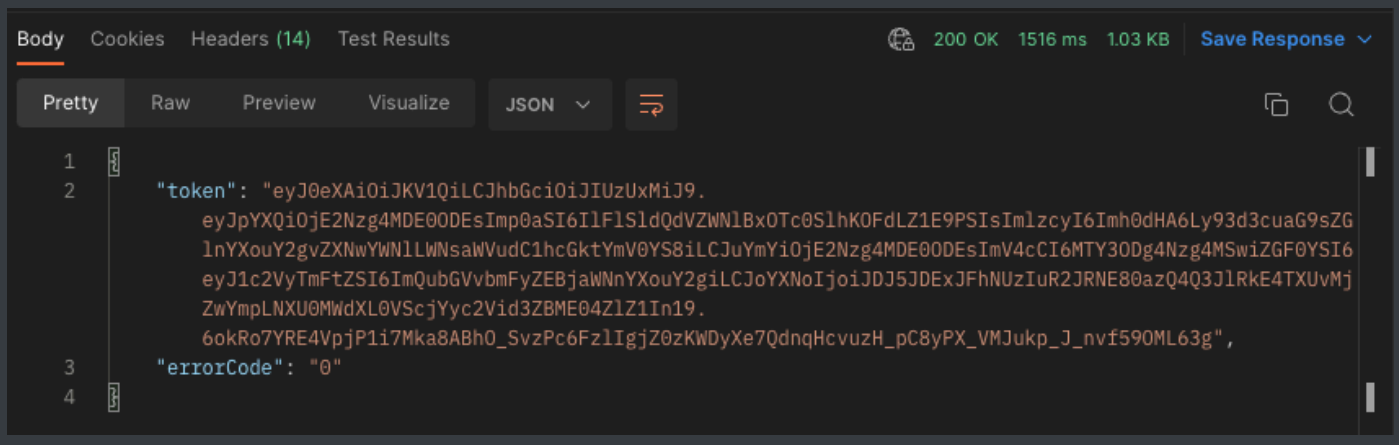
username

Cette clé correspond à votre nom d'utilisateur API qui vous a été remis avec vos informations de bienvenue.

secret_key

Correspond à un mot de passe à usage unique que vous devez calculer en partant du mot de passe qui vous a été remis avec vos informations de bienvenue. Les méthodes ci-dessous décrivent comment calculer cette valeur. Ce paramètre est obligatoire et n'est valable qu'une seule fois. Une nouvelle *secret_key* doit être générée à chaque tentative d'authentification.

Si votre authentification est validée, vous recevrez votre token JWT signé dans une réponse JSON. Les tokens transmis par notre serveur représentent un moyen d'accès sans mot de passe ; nous vous conseillons donc de les manipuler avec soin. En règle générale, un token ne doit pas être conservé plus longtemps que nécessaire.



Requêtes sur l'API

Pour pouvoir communiquer avec notre API, il est nécessaire d'envoyer le token signé JWT avec chacune de vos requêtes.

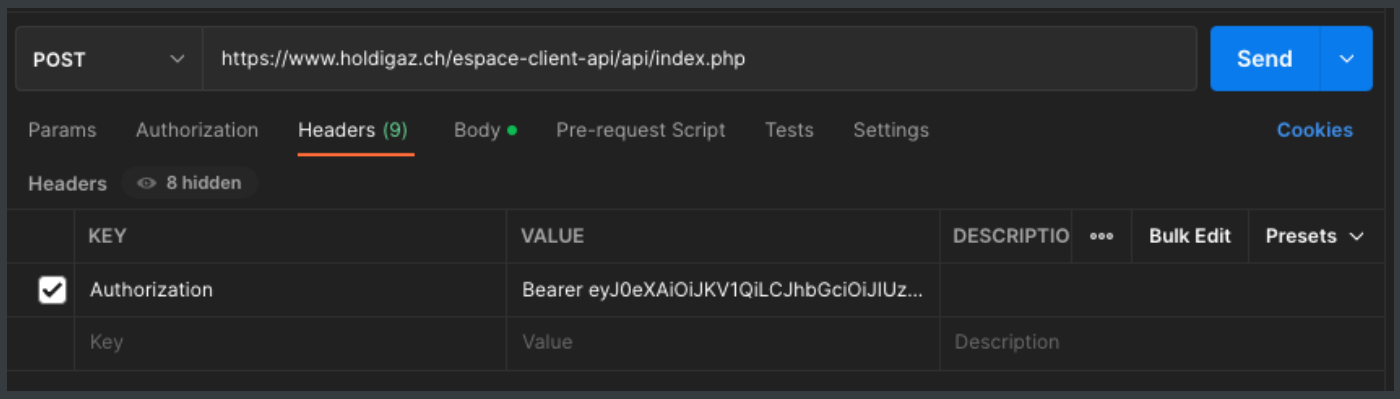
Passage de votre token pour identification

Les fonctionnalités de l'API sont accessibles à tout porteur d'un token JWT valide. Le user agent devra envoyer à chaque demande un token JWT dans le header *Authorization* en utilisant le schéma *Bearer*.

Le contenu du header sera donc du type :

```
Authorization: Bearer <token>
```

Dans Postman, il est possible d'ajouter votre token JWT dans l'onglet *Headers* de cette manière :



Création de votre requête

Servers

Veuillez utiliser cettte URL : `https://www.holdigaz.ch/espace-client-api/api/index.php` . L'accès en HTTPS est impératif.

Méthode

Tous vos appels devront être envoyés **en format POST**, sinon l'API refusera votre demande.

scope

Le `scope` correspond à la fonction API souhaitéee. Les fonctions disponibles varient en fonction des souscriptions réalisées. Elles sont en général communiquées aux clients séparément. Ce paramètre est obligatoire.

Voici une liste des `scope` standards à tous les comptes.

- `scope-test` vous permet de tester votre connexion.
- `lpn-json` vous permet d'obtenir vos relevés automatiques au format JSON.
- `installation-lpn-list` vous permet de lister les installations équipées d'un relevé automatique et disponibles dans votre souscription.

client_id

Correspond à votre numéro de client, ce numéro de client peut être trouvé sur la facture. Ce paramètre est obligatoire.

num_inst

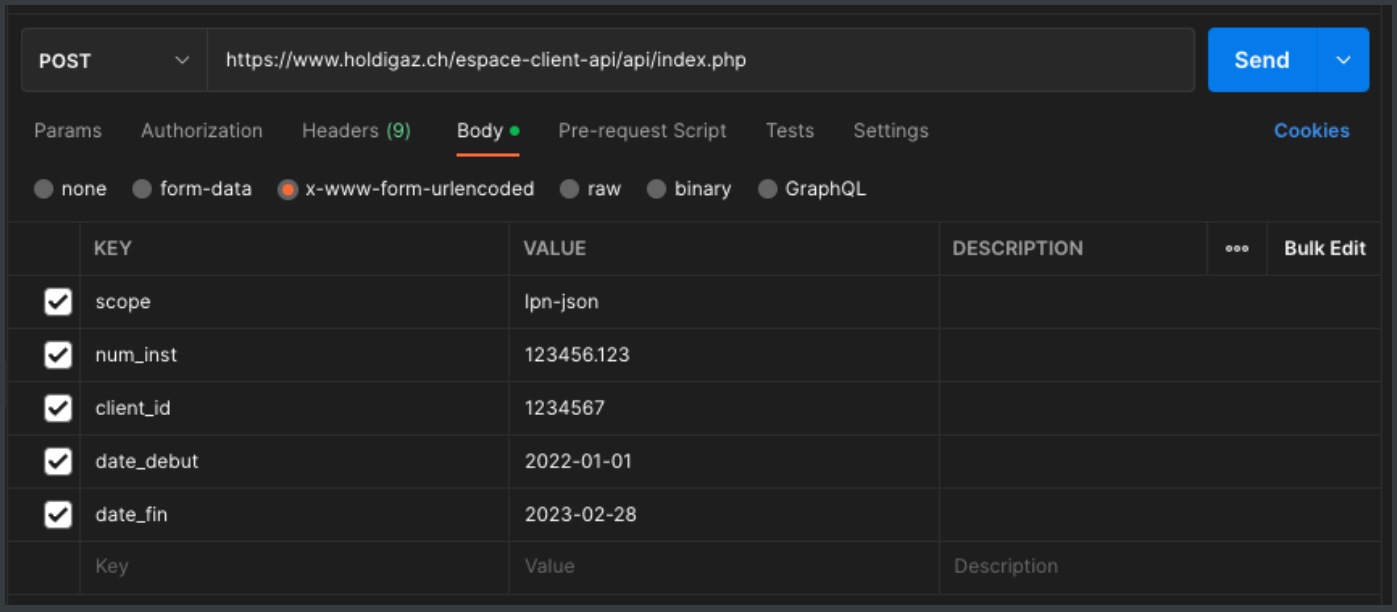
Correspond au numéro d'installation souhaité. Ce paramètre est obligatoire.

date_debut

Permet de fixer une date de début de sélection des données. Le format attendu est YYYY-MM-DD. Ce paramètre est facultatif.

date_fin

Permet de fixer une date de fin de sélection des données. Le format attendu est YYYY-MM-DD. Ce paramètre est facultatif.



Méthodes de génération de la Secret_key

La *Secret_key* est un hash temporaire **qui ne peut être utilisé qu'une fois par demande d'authentification** et qui est généré à partir du mot de passe *secret_key* que vous avez reçu dans vos informations de bienvenue. Ce hash est généré en utlisant une méthode d'encryption **Blowfish** avec *un coût de 11*. Assurez vous que cette constante soit correctement spécifiée dans votre code.

Vous trouverez ci-dessous quelques exemples de génération de la **Secret_key**.

JavaScript

```
//appelle le module bcrypt qui permet d'utiliser la méthode d'encryption blowfish
var bcrypt = require("bcrypt");

//votre mot de passe
var plaintextPassword = "password";
```

```
//définit le cost
const saltRounds = 11;

(async () => {
  //génère un salt à partir du cost
  const salt = await bcrypt.genSalt(saltRounds);
  //crée un hash compatible avec le hash de votre mot de passe en BD
  bcrypt.hash(plaintextPassword, salt, function (err, hash) {
    //imprime le hash dans la console
    //modifier cette partie afin d'avoir une solution de récupération
qui vous ciez mieux
    console.log(hash);
  });
})();
```

PHP

```
//options qui définit le cost du hash
$options = [
  'cost' => 11
];
//votre mot de passe
$password= 'password';

//affiche votre mot de secret_key
echo password_hash($password, PASSWORD_BCRYPT, $options)
```

C# / ASP.NET

```
using System.Diagnostics;

namespace app;

class Program
{
```

```
    public static void Main(string[] args)
    {
      //votre mot de passe
      var plaintextPassword = "password";

      //nombre de round utilisé (définit le cost)
      var saltRounds = 11;

      //génération du salt
      var salt = BcryptNet.GenerateSalt(saltRounds);

      //mot de passe compatible
      var hashed = BcryptNet.HashPassword(plaintextPassword, salt);

      //retourne le hash
      Console.WriteLine(hashed);
    }
  }
```

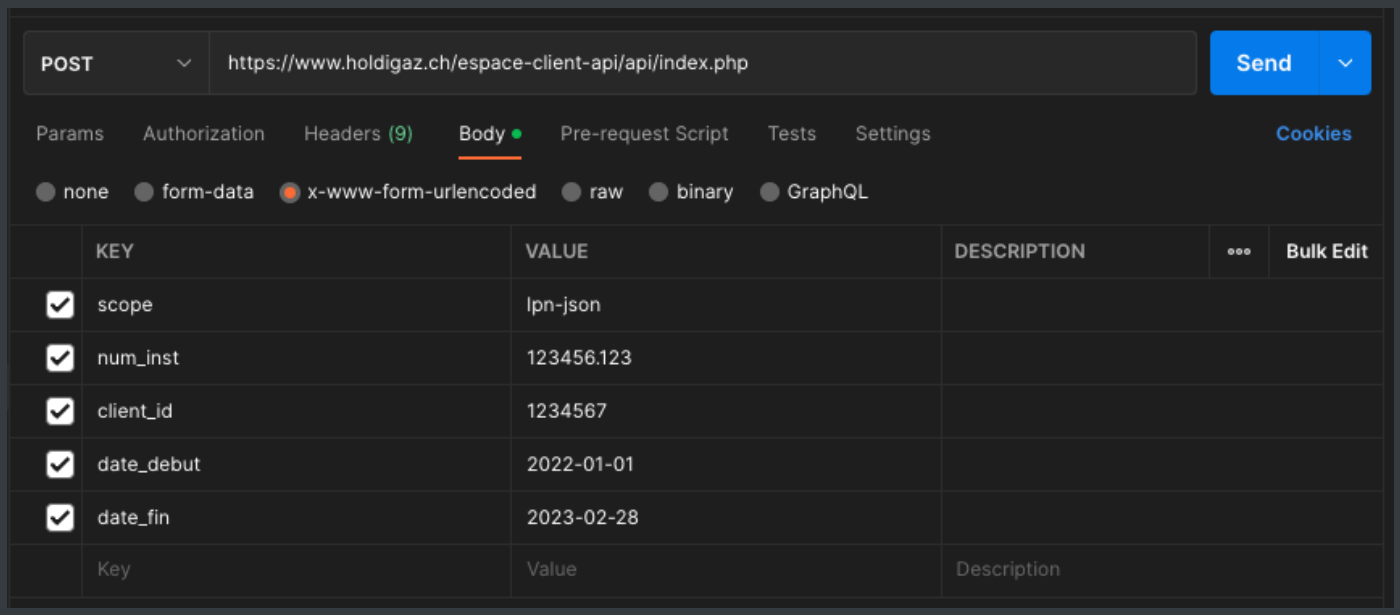
PowerShell

```
#ligne de code dans votre fichier .sh, 11 correspond au cost, password
doit être remplacer par votre mot de passe.
htpasswd -bnBC 11 "" password | tr -d ':\n'
#insère une ligne après le hash pour la lisibilité
printf "\n"
```

Tester vos accès avec Postman

Nous vous conseillons de tester vos accès avec Postman, vous trouverez une collection Postman prête à l'emploi en téléchargeant cette collection : espace-client.postman_collection.json

Il vous suffira tout simplement de saisir vos données personnelles. **Veillez à générer une secret_key à chaque demande d'authenfication requête Postman**.



Error Handling

Vous trouverez ci-dessous un tableau des différentes réponses possibles.

Description de l'erreur	JSON
La connexion est établie correctement.	`{"error":"Success.","errorCode":"0"}`
La méthode de connexion n'est pas un POST.	`{"error":"Not allowed.","errorCode":"1"}`
Le hash déja été utilisé.	`{"error":"Not allowed.","errorCode":"2"}`
La fonction donnée n'existe pas.	`{"error":"Not allowed.","errorCode":"3"}`
Le maximum de sessions simultanées est atteint.	`{"error":"Not allowed.","errorCode":"4"}`
Pas de paramètres associés à la requête.	`{"error":"Not allowed.","errorCode":"5"}`
Pas de connexion SSL	`{"error":"Not allowed.","errorCode":"6"}`

Nom d'utilisateur erroné.	`{"error":"Not allowed.","errorCode":"10"}`
Mot de passe nom spécifié.	`{"error":"Not allowed.","errorCode":"11"}`
Compte portail désactivé.	`{"error":"Not allowed.","errorCode":"12"}`
Compte API désactivé.	`{"error":"Not allowed.","errorCode":"15"}`
Pas de données LPN.	`{"error":"Not allowed.","errorCode":"100"}`
Aucune installation disponible.	`{"error":"Not allowed.","errorCode":"110"}`
Token corrompu.	`{"error":"Not allowed.","errorCode":"210"}`
Token invalide.	`{"error":"Not allowed.","errorCode":"220"}`

v1.0.0