Documentation d'utilisation de l'API GTFS-RT

Société de transport de l'Outaouais

Version 1.1

Table des matières

Prérequis

Lors de votre inscription aux données ouvertes en temps réel de la STO, vous recevrez deux clés :

1. **Une clé publique de 32 caractères :**
   Cette clé est votre identifiant d'API, elle doit être transmise lors de chaque téléchargement.
2. **Une clé privée de 64 caractères :**
   Cette clé est un secret que vous nous ferez parvenir après avoir procédé à son salage (salt).

Instructions

Lors du téléchargement des fichiers GTFS-RT vous devrez fournir les chaînes de requêtes (querystring) suivantes :

key (requis) : Représente la **clé publique** en majuscule composée de 32 caractères hexadécimaux fournis par la STO.

file (requis) : Représente le fichier GTFS-RT que vous voulez télécharger.
Valeurs possibles : `alert`, `vehicule`, `trip`

hash (requis) : Représente un hash unique de l'algorithme SHA256 **en majuscule** provenant de la concaténation de la **clé privée** de 64 caractères fournie par la STO suivie du temps courant au format ISO-8601 UTC avec les particularités suivantes :

1. Le temps n'inclut pas les secondes.
2. La date est séparée du temps par un 'T' majuscule.
3. Hormis le 'T' il n'y a aucun autre caractère d'espacement ou de séparation tels que les espaces, les traits d'union '-' et les doubles points ' :'.
4. La notation *d'offset* utilise le Z.

Par exemple, `20191230T1430Z` représente le 30 décembre 2019 à 14h30 UTC.

La concaténation de ces deux valeurs devrait ressembler à l'exemple suivant :
`D4A6EF9E8F1E4F12598BF0A23805303FC24661C71D9DC6E6691540A5EA13E3C520191204T1918Z`

Une fois l'algorithme SHA256 appliqué à la valeur concaténée, le résultat devrait ressembler à ceci :
`D98AC562B7BCE903D3F644B8EC9EA91824FA60F7F58166351DB93E6E35342BF8`

Pour plus d'informations sur la norme ISO-8601, référez-vous à la page suivante : `https://fr.wikipedia.org/wiki/ISO_8601`.

stream (optionnel) : Représente le flux voulu entre *standard* et *custom*.Le flux *custom* s'applique uniquement au fichier *trip*. Ce flux contient des informations supplémentaires sur les prochains trajets. Certaines informations sont hors norme du standard GTFS-RT.

Valeurs possibles :
— `standard` (par défaut)
— `custom`

Exemples d'utilisations par langage

Les exemples suivants prennent en considération que l'on veut récupérer le fichier `alert.pb`.

PHP

```php
// constants
$host = "https://gtfs.sto.ca/download.php";
$key = "<cle_publique_32_caracteres>";
$file = "alert";
$secret = "<cle_prive_64_caracteres>";

// hash
$date_utc = new DateTime("now", new DateTimeZone("UTC"));
$date_iso8601 = $date_utc->format('Ymd') . 'T' . $date_utc->format('Hi')
    . 'Z';
$saltedSecret = $secret . $date_iso8601;
$hash = strtoupper(hash('sha256', $saltedSecret));

// url
$url = $host . "?hash=" . $hash . "&file=" . $file . "&key=" . $key;
$gtfsrt = file_get_contents($url);

$gtfsrt = file_get_contents($url);
file_put_contents("./alert.pb", $gtfsrt);
```

Python

```python
import datetime
import hashlib
import urllib.request

#constants
host = "https://gtfs.sto.ca/download.php"
key = "<cle_publique_32_caracteres>"
file = "alert"
secret = "<cle_prive_64_caracteres>"

#hash
date_utc = datetime.datetime.now(datetime.timezone.utc)
date_iso8601 = date_utc.strftime('%Y%m%dT%H%MZ')
salted_secret = secret + date_iso8601
hash_value = hashlib.sha256(salted_secret.encode('utf-8')).hexdigest()

#url
url = f"{host}?hash={hash_value.upper()}&file={file}&key={key}"

urllib.request.urlretrieve(url, './alert.pb')
```

C#

Fonctionnel avec .NET Framework et .NET Core/.NET 5+

```
using System;
using System.Net;
using System.Text;
using System.Security.Cryptography;

class Program
{
    static void Main()
    {
        //contants
        string host = "https://gtfs.sto.ca/download.php";
        string key = "<cle_publique_32_caracteres>";
        string file = "alert";
        string secret = "<cle_prive_64_caracteres>";

        //hash
        string date_iso8601 = DateTime.UtcNow.ToString("yyyyMMddTHHmmZ");
        string saltedSecret = secret + date_iso8601;

        var sha256 = new System.Security.Cryptography.SHA256Managed();
        byte[] hashBytes =
            sha256.ComputeHash(Encoding.UTF8.GetBytes(saltedSecret));
        string hash =
            System.BitConverter.ToString(hashBytes).Replace("-",
            "").ToUpperInvariant();

        //url
        string url = host + "?hash=" + hash + "&file=" + file + "&key=" +
            key;
        var client = new WebClient();

        client.DownloadFile(url, "./alert.pb");
    }
}
```

Java

Fonctionnel avec Java 8+. Le code suivant ne gère pas les exceptions pour des raisons de simplicité.

```java
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.URL;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

public class Program {
    public static void main(String[] args) {
        // Constants
        String host = "https://gtfs.sto.ca/download.php";
        String key = "<cle_publique_32_caracteres>";
        String file = "alert";
        String secret = "<cle_prive_64_caracteres>";

        try {
            //Hash
            SimpleDateFormat sdf = new
                SimpleDateFormat("yyyyMMdd'T'HHmm'Z'");
            sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
            String date_iso8601 = sdf.format(new Date());
            String saltedSecret = secret + date_iso8601;

            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hashBytes =
                digest.digest(saltedSecret.getBytes("UTF-8"));
            StringBuilder sb = new StringBuilder();
            for (byte b : hashBytes) {
                sb.append(String.format("%02X", b));
            }
            String hash = sb.toString().toUpperCase();

            // URL
            String urlStr = host + "?hash=" + hash + "&file=" + file +
                "&key=" + key;
            try (BufferedInputStream in = new BufferedInputStream(new
                URL(urlStr).openStream());
              FileOutputStream fileOutputStream = new
                 FileOutputStream("./alert.pb")) {
                byte dataBuffer[] = new byte[1024];
                int bytesRead;
                while ((bytesRead = in.read(dataBuffer, 0, 1024)) != -1) {
                    fileOutputStream.write(dataBuffer, 0, bytesRead);
                }
            }
        }
    }
}
```

JavaScript

```
const https = require('https');
const fs = require('fs');
const crypto = require('crypto');

// constants
const host = "https://gtfs.sto.ca/download.php";
const key = "<cle_publique_32_caracteres>";
const file = "alert";
const secret = "<cle_prive_64_caracteres>";

// hash
const now = new Date();
const date_iso8601 = now.toISOString().replace(/[-:]/g,
    '').split('.')[0].slice(0, -2) + 'Z';
const salted_secret = secret + date_iso8601;
const hash_value = crypto.createHash('sha256').update(salted_secret,
    'utf8').digest('hex').toUpperCase();

// url
const url = `{host}?hash=${hash_value}&file=${file}&key=${key}`;

const fileStream = fs.createWriteStream('alert.pb');
https.get(url, (response) => {
    if (response.statusCode === 200) {
        response.pipe(fileStream);
        fileStream.on('finish', () => {
            fileStream.close();
        });
    }
});
```