Open data tpg

API horaires temps réel

Documentation pour les utilisateurs

Version 1.1

Version	Date	Commentaire
1.0	27.08.2013	Création
1.1	09.09.2013	Mise à jour par GFI

Table des matières

1. Objet de ce document

Ce document détaille les fonctions mises à disposition par l'API Horaires temps réel des tpg. Il explique comment consommer et comment structurer les appels aux web services et détaille le format des réponses.

2. Remarques générales

2.1 Conditions générales d'utilisation

Pour accéder aux données de l'API Horaires temps réel, il est nécessaire d'accepter les conditions générales d'utilisation de ces données.
Ces conditions générales d'utilisation sont consultables sur le site open data tpg : http://data.tpg.ch.

2.2 Accès à l'API

Pour consommer les données de l'API Horaires temps réel, il est nécessaire d'utiliser une clé d'API qui sera envoyée lors de chaque appel au web service. Pour obtenir une clé d'API, consulter le site open data tpg : http://data.tpg.ch.

Il est également nécessaire d'indiquer la version de l'API demandée dans chaque appel au web service.
Les données Horaires temps réel sont mises à disposition par des appels sur le serveur: http://rtpi.data.tpg.ch.

Exemples de requêtes :
Liste des arrêts commerciaux :
http://rtpi.data.tpg.ch/v1/GetStops?key=xxxx

Info trafic : http://rtpi.data.tpg.ch/v1/GetDisruptions?key=xxxx

Prochains départs :
http://rtpi.data.tpg.ch/v1/GetNextDepartures?key=xxxx&stopCode=xxxx
où xxxx sont des paramètres d'entrées.

2.3 Format des sorties

Les réponses sont disponibles aux formats JSON et XML. Exemples :
http://rtpi.data.tpg.ch/v1/GetDisruptions.json?key=xxxxx
http://rtpi.data.tpg.ch/v1/GetDisruptions.xml?key=xxxxx

Remarque : si aucun format de sortie n'est spécifié, la réponse est au format XML par défaut.
Chaque réponse est horodatée ; elle possède un « timestamp » au format ISO étendu date, heure, minutes et fuseau horaire.
Exemple : <timestamp>2013-08-16T11:58:24+0200</timestamp> : réponse en date du 16 août 2013 à 11h58min24sec, fuseau horaire GMT+2.
L'encodage utilisé est UTF-8.

2.4 Source des données

Les données proviennent du système SAEIV des tpg (Système d'Aide à l'Exploitation et à l'Information Voyageurs) des tpg.

2.5 Vocabulaire utile

Voici quelques éléments de vocabulaire que vous retrouverez dans les descriptions détaillées des fonctions :

2.5.1 Différence entre Arrêt commercial et Arrêt physique

Un arrêt commercial regroupe plusieurs arrêts physiques de même nom.
Par exemple, l'arrêt commercial « Gare Cornavin » regroupe plusieurs arrêts physiques (correspondant à des arrêts localisés autour de la gare, chacun pouvant être desservi par des lignes distinctes).

2.5.2 Journée d'exploitation

Cela correspond à une journée au sens de l'exploitation tpg. Une journée d'exploitation débute à 4h00 le matin et se termine à 3h30 le lendemain.

2.6 Heures de mise à disposition des données

Entre 3h30 et 4h15 le matin, pour des raisons d'initialisation des données dans le système temps réel des tpg, les données de l'API ne sont pas accessibles.

3. Récupération des arrêts

3.1 Récupération des arrêts commerciaux : GetStops

3.1.1 Description

Cette fonction renvoie les caractéristiques (code, nom, lignes, destinations) d'un ou plusieurs arrêts commerciaux sélectionnés suivant différents paramètres.
L'information est valable pour la journée d'exploitation courante.

A noter : les arrêts renvoyés sont l'ensemble des arrêts du réseau tpg pour lesquels il existe au moins un horaire de passage dans la journée d'exploitation courante. Aussi, si un arrêt n'est pas desservi le dimanche et que l'API GetStops est appelée un dimanche, cet arrêt ne fera pas partie de la liste.

3.1.2 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
stopCode	Codes commerciaux des arrêts recherchés	non	String : Liste de codes arrêts séparés par des virgules (,)
stopName	Nom partiel des arrêts recherchés	non	String : Nom partiel
line	Nom de la ligne dont on souhaite connaître les arrêts	non	String : Nom de la ligne
latitude	Coordonnées de l'utilisateur	non	Float : Latitude dans le système WGS84
longitude	Coordonnées de l'utilisateur	non	Float : Longitude dans le système WGS84

3.1.3 Utilisations possibles

- **Aucun paramètre** : on renvoie la liste de tous les arrêts commerciaux pour la journée d'exploitation en cours, triés par code arrêt (stopCode) croissant.
ex : GetStops

- **Uniquement le paramètre stopCode** : on renvoie la liste des arrêts commerciaux dont le code est dans la liste saisie en entrée, triés par code arrêt croissant.
ex : GetStops?stopCode=CVIN,BHET

- **Uniquement le paramètre stopName** : on renvoie la liste des arrêts commerciaux dont le nom contient la sous-chaîne de caractère saisie, triés par code arrêt croissant.
ex : GetStops?stopName=gare

- **Uniquement le paramètre line** : on renvoie la liste des arrêts commerciaux qui sont desservis par la ligne saisie, triés par code arrêt

croissant.
ex : GetStops?lineCode=12

- **Les paramètres latitude et longitude** : on renvoie la liste des arrêts commerciaux se situant à moins de 500m des coordonnées saisies, triés par distance croissante au point saisi.
ex : GetStops?latitude=46.218176&longitude=6.146445

Ces filtres sont exclusifs. Si trop de paramètres sont renseignés en entrée, on obtient l'erreur 12 : Too many parameters (voir détails dans le chapitre *Codes d'erreurs*).

3.1.4 Format de sortie

XML :

```
<stops>
 <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

 <stops>
        <stop>
                <stopCode>CVIN</stopCode>
                <stopName>Gare Cornavin</name>

                <connections>
                        <connection>
                                <lineCode>1</lineCode>
                                <destinationName>Petit-Bel-Air</destinationName>
                                <destinationCode>PETIT-BEL-AIR</destinationCode>
                        </connection>
                        <connection>
                                <lineCode>1</lineCode>
                                <destinationName>Jar.-Botanique</destinationName>
                                <destinationCode>JAR. BOTANIQUE</destinationCode>
                        </connection>
                        …
        </connections>

        ...
        <distance>103</distance>
        </stop>
        <stop>
                …
        </stop>
```

```
        …
  </stops>

  ...
</stops>
```

JSON :

```
{
    stops:{
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        stops:[
            {
                stopCode: ”CVIN”,
                stopName:”Gare Cornavin”,
                distance:103
                connections:[
                    {
                        lineCode:1,
                        destinationName:”Petit-Bel-Air”,
                        destinationCode:”PETIT-BEL-AIR”
                    },
                {
                        lineCode:1,
                        destinationName:”Jar.-Botanique”,
                        destinationCode:”JAR. BOTANIQUE”
                    },
                    …
                ]
            },
              …
        ]
    }
}
```

Remarque : La propriété *distance* n’est disponible que lorsqu’on renseigne des coordonnées (latitude, lagitude) en entrée.

Si le codeArret ou le codeLigne demandé est inconnu du système, la réponse est vide :

XML :

```
<stops>
```

```
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
</stops>
```

JSON:

```
{ stops:{
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
            }
    }
```

3.2 Récupération des arrêts physiques : GetPhysicalStops

3.2.1 Description

Cette fonction renvoie les caractéristiques (code, coordonnées, nom, lignes, destinations) d'un ou plusieurs arrêts physiques sélectionnés suivant différents paramètres.

3.2.2 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
stopCode	Codes commerciaux des arrêts recherchés	non	String : Liste de codes arrêts séparés par des virgules (,)
stopName	Nom partiel des arrêts recherchés	non	String : Nom partiel

3.2.3 Utilisations possibles

- **Aucun paramètre** : on renvoie la liste de tous les arrêts physiques de la journée d'exploitation en cours, triés par code arrêt croissant.
  ex : GetPhysicalStops

- **Uniquement le paramètre stopCode** : on renvoie la liste des arrêts physiques dont le code figure dans la liste saisie, triés par code arrêt croissant.
  ex : GetPhysicalStops?stopCode=CVIN,BHET

- **Uniquement le paramètre name** : on renvoie la liste des arrêts physiques dont le nom contient la sous-chaîne de caractère saisie, triés par code arrêt croissant.
  ex : GetPhysicalStops?name=gare

Ces filtres sont exclusifs : si stopCode et stopName sont renseignés en entrée, on obtient l'erreur 12 : Too many parameters.

3.2.4 Format de sortie

XML :

```
<stops>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

  <stops>
        <stop>
              <stopCode>CVIN</stopCode>
              <stopName>Gare Cornavin</name>

              <physicalStops>
                    <physicalStop>
                          <phyicalStopCode>CVIN01</phyicalStopCode>
                          <stopName>Gare Cornavin</name>

                          <connections>
                                <connection>
                                      <lineCode>1</lineCode>
                                      <destinationName>Petit-Bel-
Air</destinationName>
                                      <destinationCode>PETIT-BEL-
AIR</destinationCode>
                                </connection>
                                …
                          </connections>

                          <coordinates>
                                <referential>WGS84</referential>
                                <latitude>46.21260795937985</latitude>
                                <longitude>6.142041223117598</longitude>
                          </coordinates>

                    </physicalStop>
                    <physicalStop>
                          <phyicalStopCode>CVIN02</phyicalStopCode>
                          <stopName>Gare Cornavin</name>

                          <coordinates>
                                <referential>WGS84</referential>
                                <latitude>46.21260795937985</latitude>
                                <longitude>6.142041223117598</longitude>
                          </coordinates>
```

```
                        <connections>
                            <connection>
                                <lineCode>1</lineCode>
                                <destinationName>Jar.-
Botanique</destinationName>
                                <destinationCode>JAR.
BOTANIQUE</destinationCode>
                            </connection>
                            …
                        </connections>
                    </physicalStop>
                    …
                </physicalStops>
            </stop>
            <stop>
                ...
            </stop>
            ...
    </stops>
</stops>
```

JSON :

```
{
    stops:{
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        stops:[
            {
                stopCode:“CVIN”,
                stopName:”Gare Cornavin”,
                physicalStops :[
                    {
                        physicalStopCode:"CVIN01",
                        stopName:”Gare Cornavin”,

                        coordinates:{
                            referential:”WGS84”,
                            latitude: 46.21260795937985,
                            longitude: 6.142041223117598
                        },
                        connections:[
                        {
                            lineCode:1,
                            destinationName:”Petit-Bel-Air”,
                            destinationCode:”PETIT-BEL-AIR”
                        },
                    {
                            lineCode:1,
                            destinationName:”Jar.-Botanique”,
                            destinationCode:”JAR. BOTANIQUE”
                        },
                        …
                        ]

                    },
                    …
                ]
            },
              …
        ]
    }
}
```

Si le codeArret demandé est inconnu du système, ou que la chaîne de caractère saisie dans stopName ne correspond à aucun arrêt, la réponse est vide :

XML :

```
<stops>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
</stops>
```

JSON:

```
{ stops:{
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
          }
    }
```

4. Récupération des prochains départs

4.1 Récupération des départs dans l'heure : GetNextDepartures

4.1.1 Description

Cette fonction renvoie les caractéristiques des prochains départs suivant différents paramètres. Les départs sont triés par heure de départ croissante.

4.1.2 Précisions

4.1.2.1 Temps d'attente

On renvoie les départs prévus dans l'heure courante, c'est-à-dire pour lesquels le temps d'attente est inférieur ou égal à 59min59 secondes.
Un horaire peut être fiable (<reliability>F</reliability>) ou théorique/approximatif (<reliability>T</reliability>).
Si pour une ligne et destination données, le départ le plus proche est dans plus d'une heure, on renvoie « >1h » : <waitingTime>>1h</waitingTime>.
Si pour une ligne et destination données, il y a eu des départs dans la journée mais qu'il n'y en a plus, on renvoie « plus de passage » : <waitingTime>no more</waitingTime>.
Un véhicule pour lequel le temps d'attente waitingTime est nul est dit « véhicule à l'approche ». Un waitingTime nul peut être associé à un temps d'attente nul, strictement positif ou strictement négatif.

4.1.2.2 Equipement PMR

L'attribut <characteristics> indique si le véhicule est équipé pour les Personnes à Mobilité Réduite (PMR).

4.1.2.3 Perturbations et déviations

Si un arrêt habituellement desservi par le véhicule fait l'objet d'une déviation, on indique cette déviation (voir détails plus loin).
Si des perturbations affectent une course, un objet Perturbation est renvoyé (voir détails plus loin).

4.1.3 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
stopCode	Code commercial de l'arrêt souhaité	oui	String : Code arrêt
departureCode	Code horaire permettant	non	Integer : Code horaire d'un

	d'identifier les correspondances possibles (voir détails plus loin)		départ d'un véhicule
linesCode	Filtre par lignes	non	String : Liste de codes de lignes séparés par des virgules (,)
destinationsCode	Filtre par destinations	non	String : Liste de codes de destinations séparées par des virgules (,)

4.1.4 Utilisations possibles

- **Uniquement le paramètre stopCode** : on renvoie la liste des prochains départs dans l'heure à l'arrêt demandé.
  ex : GetNextDepartures?stopCode=CVIN

- **Les paramètres stopCode et departureCode** : on renvoie la liste des prochains départs dans l'heure à l'arrêt demandé et on précise pour chaque départ le temps d'attente par rapport à la correspondance indiquée en entrée.
  ex : GetNextDepartures?stopCode=WTC0&departureCode=15468 :

Un passager souhaite se rendre de l'arrêt De Joinville à l'arrêt Blandonnet. Pour cela il souhaite prendre la ligne 10 de De Joinville à WTC (code arrêt WTC0, departureCode=15468), puis changer à WTC pour prendre la ligne Y en direction de Val-Thoiry et descendre à Blandonnet. L'arrivée de son bus à.WTC est estimée dans 28 minutes. Tous les départs à l'arrêt WTC prévus avant cette date auront donc des connectionWaitingTime négatifs, ce qui signifie que ces correspondances ne seront pas possibles.

- **Utilisation des paramètres linesCode et destinationsCode** : on ne renvoie que les départs répondant aux filtres choisis.

4.1.5 Format de sortie

XML :

```
<nextDepartures>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

  <stop> (rappel des caractéristiques de l'arrêt ; même objet arrêt que pour les fonctions GetStops…)
        ...
  </stop>

  <departures>
        <departure>
              <departureCode>27439</departureCode>
              <waitingTime>0</waitingTime>
              <waitingTimeMillis>-20000</waitingMillis>

              <connectionWaitingTime>12</connectionWaitingTime>

              <connection>
                    <lineCode>12</lineCode>
                    <destinationName>Palettes</destinationName>
                    <destinationCode>PALETTES</destinationCode>
              </connection>

              <reliability>F</reliability>
              <characteristics>PMR</characteristics>

              <disruptions>
                    <disruption>
                          <disruptionCode>13306</disruptionCode>
                    <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
                    <place>77 Route de Chêne</lieu>
                    <nature>Suite accident</nature>
                    <consequence>Retard de 15 minutes</consequence>
                          <stopName>Grange Canal</stopName>
                    </disruption>
                    …
              </disruptions>

              <deviation>
                          <deviationCode>O</deviationCode>
              </deviation>
        <departure>
```

```
        …
  </departures>
</nextDepartures>
```

JSON :

```
{
    nextDepartures:{
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        stop:{ (même objet arrêt que pour les fonctions GetStops…) },
        departures:[
            {
                departureCode:27439,
                waitingTime:0,
                waitingTimeMillis:-20000,
                connectionWaitingTime:12,
                connection:{
                    lineCode:12,
                    destinationName:”Palettes”,
                    destinationCode:”PALETTES”
                },
                reliability:”F”,
                characteristics:”PMR”,
                deviation:{
            deviationCode:"O"
            }
                disruptions:{
                    disruptionCode:”137907”,
             timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
                    place:”77 Route de Chêne”,
                    nature:”Suite accident”,
                    consequence:”Retard de 15 minutes”
            stopName:”Grange Canal”,
                }
            },
            ...
        ]
    }
}
```

Remarques :

- La propriété *connectionWaitingTime* est disponible uniquement si le paramètre d’entrée **departureCode** est renseigné.

- La propriété *déviation* d'un départ est renvoyée uniquement lorsqu'une déviation affecte ce départ.

- La propriété *disruptions* d'un départ est renvoyée uniquement lorsqu'une perturbation affecte ce départ.

Si le code Arret demandé est inconnu du système, la réponse est vide :

XML :

```
<nextDepartures>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
</ nextDepartures >
```

JSON:

```
{ nextDepartures:{
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
          }
    }
```

Si le code Horaire demandé est inconnu du système, ce paramètre est ignoré et les ConnectionWaitingTime ne sont pas renvoyés.

Si le triplet (stopCode, lineCode, destinationCode) demandé n'existe pas, on renvoie l'information concernant l'arrêt mais aucun départ n'est renvoyé :

XML :

```
<nextDepartures>
<timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
<stop> … </stop> (Objet Stop, voir plus haut)
</ nextDepartures >
```

JSON:

```
{ nextDepartures:{
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
        stop {
          }
          }
    }
```

4.2 Récupération des prochains départs : GetAllNextDepartures

4.2.1 Description

Cette fonction renvoie les horaires des prochains départs (pour le reste de la journée d'exploitation) pour une ligne et destination données. Les départs sont triés par heure de départ croissante.
Remarque : Les perturbations les déviations et la distinction (fiable/théorique) ne sont pas indiquées.

4.2.2 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
stopCode	Code commercial de l'arrêt souhaité	oui	String : Code arrêt
lineCode	Nom de la ligne souhaitée	oui	String : Code de la ligne
destinationCode	Code de la destination souhaitée	oui	String : Code de la destination

4.2.3 Utilisations possibles

- **Les paramètres stopCode, lineCode et destinationCode** : on renvoie la liste des prochains départs pour la ligne et destination demandées.
ex :
GetAllNextDepartures?stopCode=CVIN&lineCode=12&destination=BHET

4.2.4 Format de sortie

XML :

```
<nextDepartures>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

  <stop> (même objet arrêt que pour la fonction GetStops)
        ...
  </stop>

  <departures>
        <departure>
              <departureCode>27439</departureCode>
              <timestamp>2013-08-16T13:58:49+0200</timestamp>
              <waitingTimeMillis>-20000</waitingMillis>
              <waitingTime>0</waitingTime>

              <connection>
                    <lineCode>12</lineCode>
                    <destinationName>Palettes</destinationName>
                    <destinationCode>PALETTES</destinationCode>
              </connection>

              <reliability>F</reliability>
        <departure>
        …
  </departures>
  ...
</nextDepartures>
```

JSON :

```
{
    nextDepartures:{
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        stop:{ (même objet arrêt que pour les fonctions GetStops…) },
        departures:[
              {
                    departureCode:27439,
                    timestamp":"2013-08-16T13:57:31+0200"
                    waitingTimeMillis:-20000,
                waitingTime:0,
                    connection:{
                          lineCode:12,
                          destinationName:”Palettes”,
                          destinationCode:”PALETTES”
```

```
                        },
                        reliability:”F”,
                    },
                    ...
        ]
    }
}
```

Si le triplet (stopCode, lineCode, destinationCode) demandé n’existe pas, on renvoie l’information concernant l’arrêt mais aucun départ n’est renvoyé :

XML :

```
<nextDepartures>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <stop> … </stop> (Objet Stop, voir plus haut)
</ nextDepartures >
```

JSON:

```
{ nextDepartures:{
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        stop {
          }
          }
    }
```

5. Récupération des thermomètres

5.1 Récupération du thermomètre des arrêts commerciaux : GetThermometer

5.1.1 Description

Cette fonction renvoie la séquence d'arrêts commerciaux de la course qui correspond au départ souhaité. Cette séquence ordonnée de couples (arrêt, horaire de passage) est appelé « Thermomètre des arrêts».
On indique également les perturbations et déviations potentielles.

Exemple : La ligne 8 effectue le trajet Veyrier-Douane / OMS. On souhaite connaître la séquence des arrêts jusqu'à l'arrêt OMS en partant de l'arrêt IUT.
On indique donc en entrée le code horaire correspondant au passage de la ligne 8 à l'arrêt IUT en direction d'OMS, qui nous intéresse. On récupère ainsi l'ensemble des arrêts et horaires de passage du trajet entre Veyrier-Douane et OMS.
La balise <visible>True</visible> indique les arrêts situés entre IUT et OMS.

5.1.2 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
departureCode	Code horaire permettant d'indiquer le trajet souhaité	oui	Integer : Code horaire d'un départ d'un véhicule

A noter : le paramètre d'entrée est un code horaire, ce qui permet de se situer précisément sur le thermomètre du trajet concerné, y compris lorsqu'il s'agit de trajets circulaires ou desservant deux fois le même arrêt.

5.1.3 Utilisations possibles

- **paramètre departureCode** : on renvoie la liste des arrêts avec les horaires de passage à chaque arrêt ainsi que les perturbations et déviations potentielles.
  ex : GetThermometer?departureCode=156879

5.1.4 Format de sortie

XML :

```
<thermometer>
```

```
<timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

<stop> (même objet arrêt que pour la fonction GetStops)
        ...
</stop>

<lineCode>12</lineCode>
<destinationName>Palettes</destinationName>
<destinationCode>PALETTES</destinationCode>

<steps>
        <step>
                <departureCode>123324</departureCode>
                <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
                <stop>
                        (même objet arrêt que pour la fonction GetStops)
                </stop>
                <reliability>F</reliability>
                <arrivalTime>12</arrivalTime>
                <deviation>false</deviation>
                <visible>false</visible>
        </step>
        ...
        <step>
                <departureCode>123339</departureCode>
                <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
                <stop>
                        (même objet arrêt que pour la fonction GetStops)
                </stop>
                <reliability>F</reliability>
                <arrivalTime>15</arrivalTime>
                <deviation>true</deviation>
                <deviationCode>475</deviationCode>
                <visible>true</visible>
        </step>
</steps>

<disruptions>
        … (même objet que pour GetNextDepartures, si des perturbations affectent le trajet)
</disruptions>

<deviations>
        <deviation>
                <deviationCode>475</deviationCode>
                <startStop>
```

```
                    (même objet arrêt que pour la fonction GetStops)
                </startStop>
                <endStop>
                    (même objet arrêt que pour la fonction GetStops)
                </endStop>
            </deviation>
            …
    </deviations>


</thermometer>
```

JSON :

```
{ thermometer :
  {
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
        stop:{ (même objet arrêt que pour les fonctions GetStops…) },

        lineCode : 3,
        destinationName : "Gardiol",
        destinationCode : "GARDIOL",

        steps : [
                {
                        stop" : { (même objet arrêt que pour les fonctions GetStops…) },
                        departureCode : 79934,
                        timestamp : "2013-08-14T16:34:00+0200",
                        arrivalTime : 284,
                        reliability : "T",
                        deviation : true,
                        deviationCode : 475
                        visible : false
                },
                {
                        stop : { (même objet arrêt que pour les fonctions GetStops…) },
                        departureCode : 24568,
                        timestamp : "2013-08-14T16:34:00+0200",
                        arrivalTime : 284,
                        reliability : "T",
                        deviation : false,
                        visible : true
                },
                ...
      ]

        disruptions : { (même objet que pour les fonctions GetNextDepartures)
```

```
            deviations : [
                {
                        deviationCode:475,
                        startStop:{(même objet arrêt que pour les fonctions GetStops…) },
                        endStop:{ (même objet arrêt que pour les fonctions GetStops…) }
                }
        ],

    }
}
```

Remarques :

- l'attribut *deviationCode* d'un départ n'est renvoyé que lorsqu'une déviation affecte ce départ.
  Le détail de la déviation (arrêt de début, arrêt de fin) est alors disponible dans l'objet *deviations.*
- L'objet *disruptions* est renvoyé uniquement lorsqu'une perturbation affecte le thermomètre.

5.2 Récupération du thermomètre des arrêts physiques : GetThermometerPhysicalStops

5.2.1 Description

Cette fonction renvoie la séquence d'arrêts physiques de la course qui correspond au départ souhaité. Cette séquence ordonnée de couples (arrêt physique, horaire de passage) est appelé « Thermomètre des arrêts physiques».
On indique également les perturbations et déviations potentielles.

5.2.2 Paramètres d'entrée

Nom	Description	Obligatoire	Valeur attendue
departureCode	Code horaire permettant d'indiquer le trajet souhaité	oui	Integer : Code horaire d'un départ d'un véhicule

5.2.3 Utilisations possibles

- **paramètre horaireRef** : on renvoie la liste des arrêts physiques avec les horaires de passage à chaque arrêt ainsi que les perturbations et déviations potentielles.
  ex : GetThermometerPhysicalStops?horaireRef=156879

5.2.4 Format de sortie

XML :

```
<thermometer>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
  <stop> (même objet arrêt que pour la fonction GetStops)
        ...
  </stop>
  <lineCode>12</lineCode>
  <destinationCode>PALETTES</destinationCode>
  <destinationName>Palettes</destinationName>

  <steps>
        <step>
              <departureCode>123324</departureCode>
              <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
              <stop> (même objet arrêt que pour la fonction GetStops)
```

```
                ...
            </stop>
            <physicalStop>
                (même objet arrêt que pour la fonction GetPhysicalStops)
            </physicalStop>

            <deviation>true</deviation>
            <deviationCode>150</deviationCode>

            <reliability>F</reliability>
            <arrivalTime>15</arrivalTime>
            <visible>false</visible>
        </step>
        ...
        <step>
            <departureCode>123324</departureCode>
            <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
            <stop> (même objet arrêt que pour la fonction GetStops)
                ...
            </stop>
            <physicalStop>
                (même objet arrêt que pour la fonction GetPhysicalStops)
            </physicalStop>

            <deviation>false</deviation>
            <reliability>F</reliability>
            <arrivalTime>15</arrivalTime>
            <visible>true</visible>
        </step>
</steps>


<disruptions>
        … (même objet que pour GetNextDepartures, si des perturbations affectent le trajet)
</disruptions>

<deviations>
        <deviation>
            <deviationCode>475</deviationCode>
            <startStop>
                (même objet arrêt que pour la fonction GetStops)
            </startStop>
            <endStop>
                (même objet arrêt que pour la fonction GetStops)
            </endStop>
```

```
        </deviation>
        …
  </deviations>

</thermometer>
```

JSON :

```
{ thermometer :
  {
        timestamp:"YYYY-MM-DDThh:mm:ssTZD",
        stop:{ (même objet arrêt que pour les fonctions GetStops…) },
        lineCode : 3,
        destinationName : "Gardiol",
        destinationCode : "GARDIOL",

        steps : [
              {
                    physicalStop " : { (même objet arrêt que pour la fonction GetPhysicalStops…) },
                    departureCode : 79934,
                    timestamp : "2013-08-14T16:34:00+0200",
                    arrivalTime : 284,
                    reliability : "T",
                    deviation : true,
                    deviationCode : 150
                    visible : false
              },
              {
                    physicalStop " : { (même objet arrêt que pour la fonction GetPhysicalStops…) },
                    departureCode : 24568,
                    timestamp : "2013-08-14T16:34:00+0200",
                    arrivalTime : 284,
                    reliability : "T",
                    deviation : false,
                    visible : true
              },
              ...
      ]
        disruptions : { (même objet que pour les fonctions GetNextDepartures)

            deviations : [
              {
                    deviationCode:150,
                    startStop:{(même objet arrêt que pour les fonctions GetStops…)
              },
```

```
                    endStop:{ (même objet arrêt que pour les fonctions GetStops…) }
                    }
        ],

    }
}
```

6. Récupération des couleurs des lignes

6.1 Récupération des couleurs des lignes : GetLinesColors

6.1.1 Description

Cette fonction renvoie les couleurs des lignes. L'information est valable pour la journée d'exploitation courante.

6.1.2 Paramètres d'entrée

Aucun paramètre pour cette fonction.

6.1.3 Utilisations possibles

- **Aucun paramètre** : on renvoie la liste des couleurs des lignes au format hexadécimal.
  ex : GetLinesColors

6.1.4 Format de sortie

XML :

```
<colors>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

  <colors>
        <color>
              <lineCode>12</lineCode>
              <hexa>56FF42</hexa>
        </color>
        <color>
              <lineCode>13</lineCode>
              <hexa>46EF41</hexa>
        </color>
        …
  </colors>
</colors>
```

JSON :

```
{ colors :
  {
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        colors : [
              {
                    lineCode : "12",
                    hexa :"56FF42"
              },
              {
                    lineCode : "13",
                    hexa :"46EF41"
              },
              ...
        ]
        }
}
```

7. Récupération de l'info trafic

7.1 Récupération des perturbations du réseau : GetDisruptions

7.1.1 Description

Cette fonction renvoie les perturbations temps réel affectant le réseau tpg. Remarque : les perturbations planifiées (travaux, manifestations) ne sont pas renvoyées.

7.1.2 Paramètres d'entrée

Aucun paramètre pour cette fonction.

7.1.3 Utilisations possibles

- **Aucun paramètre** : on renvoie la liste des perturbations sur le réseau.

ex : GetDisruptions

7.1.4 Format de sortie

XML :

```
<disruptions>
  <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>

  <disruptions>
        <disruption>
              <disruptionCode>150</disruptionCode>
              <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
              <place>77 Route de Chêne</lieu>
              <nature>Suite accident</nature>
              <stopName>Grange Canal</stopName>
              <lineCode>12</lineCode>
              <consequence>Retard de 15 minutes entre Grange Canal et
Carouge</consequence>
        </disruption>
        …
  </disruptions>

</disruptions>
```

JSON :

```
{ disruptions :
  {
```

```
        timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
        disruptions : [
                        {
                                timestamp:”YYYY-MM-DDThh:mm:ssTZD”,
                                place:”77 Route de Chêne”,
                                nature:”Suite accident”,
                                stopName:”Grange Canal”,
                                lineCode :"12",
                                consequence:”Retard de 15 minutes entre Grange Canal et
Carouge”
                        },
                ...
        ]
        }
}
```

8. Codes d'erreurs

Les messages d'erreurs possibles sont les suivants :

8.1 Erreur 400 : Bad request

Cette erreur est renvoyée lorsque la requête par le client est syntaxiquement incorrecte, c'est-à-dire dans les cas suivants :

8.1.1 Paramètre obligatoire manquant

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>10</errorCode>
        <errorMessage>Parameter [param] is missing</errorMessage>
</error>
```

où [paramètre] correspond au premier paramètre manquant de la liste des paramètres d'entrée.

8.1.2 Mauvais format de paramètre

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>11</errorCode>
        <errorMessage>Parameter [param] format is incorrect </errorMessage>
</Error>
```

où [paramètre] correspond au premier paramètre de la liste des paramètres d'entrée dont le format est incorrect.

8.1.3 Trop de paramètres

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>12</errorCode>
        <errorMessage>Too many parameters </errorMessage>
</error>
```

8.2 Erreur 403 : Forbidden

Cette erreur est renvoyée lorsque la clé d'API n'est pas valide :

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>20</errorCode>
```

```
        <errorMessage>invalid API key </errorMessage>
</error>
```

8.3 Erreur 404 : Not found

Cette erreur est renvoyée lorsque l'url demandée est inconnue du serveur.

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>30</errorCode>
        <errorMessage>Page not found</errorMessage>
</error>
```

8.4 Erreur 410 : Gone

Cette erreur est renvoyée lorsque la version de l'api demandée n'est pas la version disponible sur le serveur.

8.4.1 API obsolète

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>40</errorCode>
        <errorMessage>The requested API version is no more available. Please
upgrade.</errorMessage>
</error>
```

8.4.2 Version d'API inconnue

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>41</errorCode>
        <errorMessage>The requested API version is incorrect<\errorMessage>
</error>
```

8.5 Erreur 503 : Service unavailable

Cette erreur est renvoyée quand le serveur API Horaires temps réel indique qu'il n'est pas disponible. Techniquement, cela correspond à deux cas possibles:

- l'initialisation en début de journée d'exploitation (réseau des arrêts et des lignes desservant ces arrêts, chaînages, couleurs des lignes) est en cours
- les données issues du système temps-réel ne semblent pas à jour (rupture de la chaîne du « temps réel »)

Dans ces deux cas, le message renvoyé est le suivant :

```
<error>
        <timestamp>YYYY-MM-DDThh:mm:ssTZD</timestamp>
        <errorCode>50</errorCode>
        <errorMessage>Service unavailable</errorMessage>
</error>
```