PROTOCOLE API
GFENCE 2400

FR **PROTOCOLE DE COMMUNICATION API GFENCE 2400**
Notice d'utilisation - Pages 1-14

EN **GFENCE 2400 API PROTOCOL**
Installation instructions - Pages 15-28

1, rue du Dauphiné – CS 90323 – 69517 – VAULX-EN-VELIN CEDEX – FRANCE
Tél. : +33(0)4.78.03.06.10 - Fax : +33(0)4.78.68.24.61 **Site web :** www.sorhea.com **Email :** commercial@sorhea.fr

NT432/ V1.0 / 06.21

Table des matières

1 INTRODUCTION

Cette notice s'adresse principalement à des développeurs logiciels familiers avec les notions d'API, requêtes Web, JSON etc…

L'unité de gestion GFENCE 2400 intègre une API HTTP pour faciliter l'intégration du produit dans des systèmes de supervision.
Cette API est basée sur une architecture REST et les données sont échangées au format JSON.

Cette notice présentera donc les différentes requêtes http (10 au total) qui peuvent être utilisées pour communiquer avec l'unité de gestion.

L'unité de gestion est consultable en lecture via des requêtes GET :

- Lecture de l'état des alarmes
- Lecture de la configuration des câbles détecteurs
- Etc…

La seule requête d'écriture existante consiste à acquitter les alarmes intrusions.

Les requêtes http décrites dans la suite du document sont compatibles dès la version V1.0 du GFENCE 2400.

Remarque : il est conseillé de lire la notice technique NT429 pour se familiariser avec les termes unité de gestion, zones, départs etc…

2 REQUETES HTTP DISPONIBLES

Cette partie présentera les différentes requêtes http que l'utilisateur pourra implémenter pour échanger des informations avec l'unité de gestion GFENCE 2400.

2.1 AUTHENTIFICATION

La méthode d'authentification est de type Digest.
L'utilisateur s'authentifie auprès de l'unité de gestion grâce au nom d'utilisateur et au mot de passe paramétrés.

➔ Se référer à la partie 6.2 de la NT429.

2.2 RECUPERATION DE LA CONFIGURATION DE L'UNITE DE GESTION

2.2.1 Lecture de la configuration générale

Cette requête permet de récupérer l'état de configuration générale de l'UG :
- Etat de configuration des départs
- Nombre de capteurs détectés si le départ est configuré
- Configuration des entrées auxiliaires et du buzzer
- Temps de réponse du défaut technique

Requête : **GET** **http://<adresse_ip>/API/config**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
configCable1	[0 ; 1] ⇔ [non-configuré ; configuré]	
configCable2	[0 ; 1] ⇔ [non-configuré ; configuré]	
nbSensorCable1	[0 → 120]	Maximum 120 capteurs
nbSensorCable2	[0 → 120]	Maximum 120 capteurs
enableAux1	[0 ; 1] ⇔ [désactivée ; activée]	
enableAux2	[0 ; 1] ⇔ [désactivée ; activée]	
enableBuzzer	[0 ; 1] ⇔ [désactivé ; activé]	
technicalDefaultDelay	[30 ; 120 ; 300 ; 900 ; 1800 ; 3600 ; 7200 ; 14400 ; 86400]	Valeur en seconde du temps de réponse du défaut technique

Exemple de requête : **GET** **http://192.168.105.202/API/config**
Exemple de réponse : **200**

{
"configCable1": 1, → 1 le câble est configuré, 0 le câble n'est pas configuré
"configCable2": 1,
"nbSensorCable1": 120, → nombre de capteurs détectés
"nbSensorCable2": 120,
"enableAux1": 0, → 1 l'AUX1 est activée, 0 l'AUX2 est désactivée
"enableAux2": 0,
"enableBuzzer": 1, → 1 le buzzer est activé, 0 le buzzer est désactivé
"technicalDefaultDelay": 30 → la valeur du temps de réponse est donnée en secondes
}

Remarque : si configCableX est à 0, nbSensorCableX n'apparait pas dans la liste des informations.

2.2.2 Lecture de la configuration des zones

Cette requête permet de récupérer la configuration des zones crées.

Requête : **GET** **http://<adresse_ip>/API/zoning**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
nbZone	[0 → 10]	Nombre de zones crées
zoneSettings	Tableau contenant la liste des zones crées	
zone	[0 → 10]	Numéro de la zone
startCable	[1 ; 2] ⇔ [Départ 1 ; Départ 2]	
startSensor	[1 → 120]	Maximum 120 capteurs
endCable	[1 ; 2] ⇔ [Départ 1 ; Départ 2]	
endSensor	[1 → 120]	

Exemple de requête : GET **http://192.168.105.202/API/zoning**
Exemple de réponse : **200**

{
"nbZone": 10, → nombre de zones crées
"zoneSettings": [
{
"zone": 1, → numéro de zone (⇔ ZONE1)
"startCable": 1, → numéro du départ de début de zone
"startSensor": 120, → numéro du capteur de début de zone
"endCable": 1, → numéro du départ de fin de zone
"endSensor": 105 → numéro du capteur de fin de zone
},
{
"zone": 2,
"startCable": 1,
"startSensor": 104,

```
        "endCable": 1,
        "endSensor": 97
      },
        .
        .
        .
      {
        "zone": 10,              → numéro de zone (⇔ ZONE10)
        "startCable": 2,
        "startSensor": 102,
        "endCable": 2,
        "endSensor": 120
      }
   ]
}
```

Remarque : la liste correspond aux nombres de zones crées. Par exemple, s'il y a 3 zones crées (ZONE1, ZONE3 et ZONE5), la liste contiendra bien 3 zones identifiées zone 1, zone 3 et zone 5.

2.2.3 Lecture de la configuration du nombre d'impacts

Cette requête permet de récupérer la configuration du nombre d'impacts affecté aux différentes zones.

Requête : **GET** **http://<adresse_ip>/API/impact-settings**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
nbZone	[0 → 10]	Nombre de zones crées
impactSettings	Tableau contenant les valeurs d'impacts réglées par zones	
impactValue	[1 → 10]	
zones	Tableau contenant la liste des zones paramétrées pour cette valeur d'impact	

Exemple de requête : **GET** **http://192.168.105.202/API/impact-settings**
Exemple de réponse : **200**

```
{
   "nbZones": 10,                → contient le nombre de zones crées
   "impactSettings": [
      {
         "impactValue": 1,       → nombre d'impacts configuré
         "zones": [1, 2…10]      → contient les numéros de zones configurées avec impact = 1
      },
      {
         "impactValue": 2,       → contient les numéros de zones configurées avec impact = 2
         "zones": [4]
      },
      {
         "impactValue": 7,       → contient les numéros de zones configurées avec impact = 7
```

```
        "zones": [8]
      }
   ]
}
```


2.2.4 Lecture de la sensibilité des capteurs

Cette requête permet de récupérer les valeurs de sensibilités des capteurs par départ.

Requête : **GET** **http://<adresse_ip>/API/sensitivity-settings**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
Cable1 / Cable2	Détails pour la direction 1 ou la direction 2	
configState	[0 ; 1] ⇔ [non-configuré ; configuré]	
nbSensor	[1 → 120]	
sensitivitySettings	Tableau contenant la liste des capteurs paramétrés avec différentes sensibilités	
sensitivityValue	[1 → 16]	1 est le moins sensible, 16 le plus sensible
sensors	[1 → 120]	Maximum 120 capteurs

Exemple de requête : **GET** **http://192.168.105.202/API/sensitivity-settings**
Exemple de réponse : **200**


```
{
   "Cable1": {                          → détail de la configuration pour le départ 1
      "configState": 1,                 → 1 le départ est configuré, 0 le départ n'est pas configuré
      "nbSensor": 120,                  → nombre de capteurs détectés
      "sensitivitySettings": [
         {
            "sensitivityValue": 10,
            "sensors": [1,2,3...70]     → contient la liste des capteurs ayant une sensibilité = 10
         },
         {
            "sensitivityValue": 12,
            "sensors": [71,72...120]    → contient la liste des capteurs ayant une sensibilité = 12
         }
      ]
   },
   "Cable2": {
      "configState": 1,
      "nbSensor": 120,
      "sensitivitySettings": [
         {
            "sensitivityValue": 12,
            "sensors": [1...120]        → contient la liste des capteurs ayant une sensibilité = 12
         }
      ]
```

```
   }
}
```

2.2.5 Lecture de la configuration réseau

Cette requête permet de récupérer la configuration réseau de l'unité de gestion.

Requête : **GET** **http://<adresse_ip>/API/network**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
ipaddr	Type attendu : string – Format X.X.X.X	
mask	Type attendu : string – Format X.X.X.X	
gwaddr	Type attendu : string – Format X.X.X.X	
gateway	[true ; false]	
speed	[10/100 Mbps ; 10 Mbps]	
HTTPport		80 par défaut

Exemple de requête : **GET** **http://192.168.105.202/API/network**
Exemple de réponse : **200**

```
{
   "ipaddr": "10.15.112.175",          → adresse IP de l'unité de gestion
   "mask": "255.255.240.0",            → valeur du masque de sous-réseau
   "gwaddr": "10.15.127.254",          → adresse IP passerelle
   "gateway": true,                    → true la passerelle est activée, false elle n'est pas activée
   "speed": "10/100 Mbps",             → débit
   "HTTPport": 80                      → valeur du port http paramétré
}
```

Remarque : si gateway est à false, gwaddr n'apparait pas dans la liste des informations.

2.2.6 Date et heure

Cette requête permet de récupérer la date et l'heure de l'unité de gestion.

Requête : **GET** **http://<adresse_ip>/API/date-time**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
date	Type attendu : string – Format AAAA-MM-JJ	
time	Type attendu : string – Format HH:MM:SS	

Exemple de requête : **GET** **http://192.168.105.202/API/date-time**
Exemple de réponse : **200**

```
{
   "date": "2021-06-10",               → format attendu AAAA-MM-JJ
```

```
   "time": "12:29:53"                    → format attendu HH:MM:SS
}
```

2.2.7 Lecture de l'état des relais

Cette requête permet de connaitre l'état des 12 relais de l'unité de gestion.

Requête : **GET** **http://<adresse_ip>/API/relay-states**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
relays_status	Liste contenant l'état des 12 relais	
zone1 → zone10	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
technical	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
AP	[0 ; 1] ⇔ [hors-alarme ; en alarme]	

Exemple de requête : **GET** **http://192.168.105.202/API/relay-states**
Exemple de réponse : **200**

```
{
   "relays_status": {
      "zone1": 1,                    → 1 le relais est en alarme, 0 le relais est hors-alarme
      "zone2": 1,
      "zone3": 1,
      "zone4": 1,
      "zone5": 1,
      "zone6": 1,
      "zone7": 1,
      "zone8": 1,
      "zone9": 1,
      "zone10": 1,
      "technical": 1,
      "AP": 0
   }
}
```

Remarque : lorsque les zones ne sont pas configurées, les relais sont en alarme.

2.2.8 Lecture de la version de l'unité de gestion

Cette requête permet de récupérer la version et le nom de l'unité de gestion, ainsi que le type du produit.

Requête : **GET** **http://<adresse_ip>/API/version**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
version	Type attendu : string	
name	Type attendu : string	

type	"gfence2400"	Valeur figée

Exemple de requête : **GET** **http://192.168.105.202/API/version**
Exemple de réponse : **200**

```
{
   "version": "V 1.0  04/06/21        ",      → numéro et date de la version
   "name": "GFENCE 2400 ",                    → nom de l'unité de gestion
   "type": "gfence2400"                       → type du produit
}
```

2.3 GESTION DES ALARMES

2.3.1 Lecture de l'état des alarmes

Cette requête permet de récupérer l'état d'alarmes des différents évènements de l'unité de gestion (intrusions sur zones, défauts techniques, AP, auxiliaires, défaut batterie).

Requête : **GET** **http://<adresse_ip>/API/alarms**
Contenu de la réponse :

Données	Valeurs	Commentaire(s)
nbZone	[0 → 10]	
intrusions	Détails des alarmes intrusions sur les 10 zones	
zone	[1 → 10]	
alarm	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
triggeredSensor	Détail du câble et du capteur déclencheur + valeur du choc	Ce champ n'apparait que s'il y a une alarme en cours (non-acquittée) sur la zone
cable	[1 ; 2]	
sensor	[1 → 120]	
sensitivity	[1 → 16]	
technicalAlarms	Détails des alarmes techniques sur la direction 1 ou direction 2	
cable1 / cable2	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
triggeredSensorCable1 / triggeredSensorCable2	Détail du capteur en défaut (selon si le défaut est présent sur le câble 1 ou le câble 2)	Ce champ n'apparait que s'il y a une alarme en cours sur le départ
sensor	[1 → 120]	
VBattDefault	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
tamper	[0 ; 1] ⇔ [hors-alarme ; en alarme]	
aux1 / aux2	[0 ; 1] ⇔ [hors-alarme ; en alarme]	

Requête : **GET** **http://192.168.105.202/API/alarms**
Exemple de réponse : **200**

```
{
```

```
  "nbZone": 10,                    → nombre de zones configurées
  "intrusions": [                  → liste des alarmes d'intrusions sur les zones
    {
      "zone": 1,                   → zone numéro 1
      "alarm": 0                   → 1 en alarme, 0 hors-alarme
    },
    {
      "zone": 2,
      "alarm": 0
    },
    {
      "zone": 3,
      "alarm": 0
    },
    {
      "zone": 4,
      "alarm": 0
    },
    {
      "zone": 5,
      "alarm": 0
    },
    {
      "zone": 6,
      "alarm": 0
    },
    {
      "zone": 7,
      "alarm": 0
    },
    {
      "zone": 8,
      "alarm": 0
    },
    {
      "zone": 9,                   → zone numéro 9
      "alarm": 1,                  → zone numéro 9 en alarme
      "triggeredSensor": {         → données du capteur en alarme
        "cable": 2,                → numéro de départ
        "sensor": 1,               → numéro du capteur déclencheur
        "sensitivity": 11          → valeur d'amplitude du choc
      }
    },
    {
      "zone": 10,
      "alarm": 0
    }
  ],
  "technicalAlarms": {             → liste des alarmes techniques
    "cable1": 0,                   → 1 défaut technique actif, 0 défaut technique inactif
    "cable2": 1,
    "triggeredSensorCable2": {     → localisation du défaut technique sur le départ 2
      "sensor": 1                  → numéro de capteur en défaut
    },
```

```
      "VBattDefault": 0,        → 1 défaut batterie actif, 0 défaut batterie inactif
      "tamper": 1,              → 1 AP ouvert (en alarme), 0 AP fermé (hors-alarme)
      "aux1": 1,                → 1 auxiliaire en alarme, 0 auxiliaire hors-alarme
      "aux2": 1
   }
}
```

Remarque : lorsque les entrées auxiliaires sont désactivées, l'état est en alarme.

2.3.2 Acquittement des alarmes

Cette requête permet l'acquittement des alarmes intrusions sur les zones.

Requête : **GET** **http://<adresse_ip>/update.cgi?type=11**

Exemple de requête : **GET** **http://192.168.105.202/update.cgi?type=11**
Exemple de réponse : **200**

2.4 GESTIONS DES CAS D'ERREURS

Exemple de cas d'erreurs :

- ➔ Utilisation de mauvais identifiants pour l'authentification Digest
  Réponse : **401 Unauthorized : password required**

- ➔ Mauvaise définition de l'URL de la requête
  Exemple : **GET** **http://192.168.105.202/alarms**
  (url attendu : http://192.168.105.202/API/alarms)

  Réponse: **404 File not found**

- ➔ Erreur interne de récupération de données
  Réponse : **500 Internal Server Error**

3 EXEMPLE D'ALGORITHME

Dans cette partie, nous allons proposer une idée d'algorithme pour gérer les alarmes intrusions.

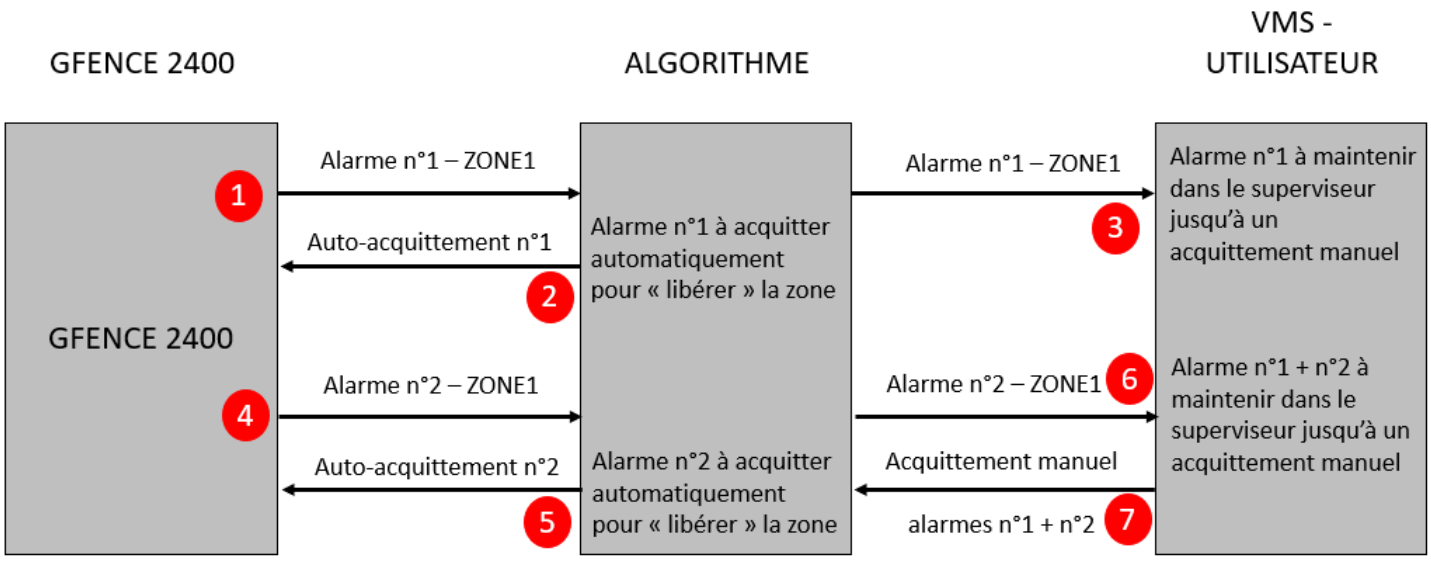
Les alarmes intrusions sur zones sont mémorisées dans l'API par zone pour permettre leur lecture. La mémorisation est maintenue jusqu'à la réception d'une commande d'acquittement de l'utilisateur. Les alarmes sur les autres zones continuent de remonter comme attendu.

Remarque : les alarmes les plus récentes sur une même zone sont stockées dans l'historique de l'unité de gestion indépendamment de l'API.

Pour une intégration de l'unité de gestion via ses API dans un système de supervision, nous proposons de mettre en place un acquittement « automatique » dès qu'une alarme intrusion apparait dans le but de « libérer » la zone qui a déclenché. Cet acquittement automatique permet de remettre rapidement l'unité de gestion « en écoute » de la prochaine alarme intrusion sur cette même zone.

Selon le besoin du client, il peut être nécessaire de maintenir l'alarme active dans le logiciel de supervision jusqu'à un acquittement manuel par l'utilisateur.

Ci-dessous un synoptique explicatif :



1. Récupération d'une première alarme sur la ZONE 1
   GET **http://<adresse_ip>/API/alarms**

2. Auto-acquittement de l'alarme sur la ZONE 1 pour acquitter la zone
   GET **http://<adresse_ip>/update.cgi?type=11**

3. Maintien de l'alarme dans le superviseur jusqu'à un acquittement manuel de l'utilisateur

4. Récupération d'une deuxième alarme sur la ZONE 1
   GET **http://<adresse_ip>/API/alarms**

5. Auto-acquittement de l'alarme sur la ZONE 1 pour acquitter la zone
   GET **http://<adresse_ip>/update.cgi?type=11**

6. Maintien de l'alarme 1 + alarme 2 dans le superviseur jusqu'à un acquittement manuel de l'utilisateur

7. Acquittement manuel de l'utilisateur des alarmes 1 et 2

Conformément aux directives européennes sur l'environnement, ce produit ne doit pas être jeté mais recyclé dans une filiale appropriée.

CONTENTS

1 INTRODUCTION

This user manual is dedicated to software developers familiar with the concepts of API, Web requests, JSON…

The GFENCE 2400 control unit features a HTTP API to ease the device integration into supervising systems.
This API is based on a REST architecture and data are transmitted into JSON format.

This user manual will present the HTTP requests that can be used to communicate with the control unit.

The control unit configuration is readable through GET requests:
- Retrieve alarm states
- Retrieve cables configurations
- Etc…

There is only one writing request used to acknowledge intrusion alarms.

HTTP requests describe below are compatible with V1.0 of the GFENCE 2400.

Note: it is advisable to read the NT429 to familiarize yourself with the terms control unit, zones, cables…

2 HTTP REQUESTS AVAILABLE

This first part will describes the HTTP requests that a user can implement to interact with the GFENCE 2400 control unit.

2.1 AUTHENTICATION

The authentication method is Digest.
The user identifies himself to the control unit using a username and a password.

➔ Refer to part 6.2 in NT429.

2.2 RETRIEVE CONTROL UNIT CONFIGURATION

2.2.1 Retrieve general configuration

This request allows the user to retrieve the general configuration of the control unit:
- Cables configuration states
- Number of detected sensors on each configured cables
- Auxiliaries input and buzzer configuration state
- Technical response time

Request: **GET** **http://<ip_address>/API/config**
Response content:

Data	Values	Comments
configCable1	[0 ; 1] ⇔ [unconfigured ; configured]	
configCable2	[0 ; 1] ⇔ [unconfigured ; configured]	
nbSensorCable1	[0 → 120]	Max 120 sensors
nbSensorCable2	[0 → 120]	Max 120 sensors
enableAux1	[0 ; 1] ⇔ [deactivated ; activated]	
enableAux2	[0 ; 1] ⇔ [deactivated ; activated]	
enableBuzzer	[0 ; 1] ⇔ [deactivated ; activated]	
technicalDefaultDelay	[30 ; 120 ; 300 ; 900 ; 1800 ; 3600 ; 7200 ; 14400 ; 86400]	Technical default response time in seconds

Request example: **GET** **http://192.168.105.202/API/config**
Response example: **200**

```
{
    "configCable1": 1,          → 1 the cable is configured, 0 the cable isn't configured
    "configCable2": 1,
    "nbSensorCable1": 120,      → it contains the number of detected sensors
    "nbSensorCable2": 120,
    "enableAux1": 0,            → 1 AUX1 is enable, 0 AUX1 is disable
    "enableAux2": 0,
    "enableBuzzer": 1,          → 1 buzzer is enable, 0 is disable
```

"technicalDefaultDelay": 30 → response time value unit is seconds
}

Note: if configCableX has the value 0, nbSensorCableX isn't showed in the information list.

2.2.2 Retrieve zones configuration

This request allows the user to retrieve the created zones configuration.

Request: **GET** **http://<ip_address>/API/zoning**
Response content:

Data	Values	Comments
nbZone	[0 → 10]	Number of created zones
zoneSettings	Tab containing a list of all the created zones	
zone	[0 → 10]	Zone value
startCable	[1 ; 2] ⇔ [Cable 1 ; Cable 2]	
startSensor	[1 → 120]	Max 120 sensors
endCable	[1 ; 2] ⇔ [Cable 1 ; Cable 2]	
endSensor	[1 → 120]	

Request example: **GET** **http://192.168.105.202/API/zoning**
Response example: **200**

```
{
   "nbZone": 10,                  → number of created zones
   "zoneSettings": [
      {
         "zone": 1,               → zone value (⇔ ZONE1)
         "startCable": 1,         → value of the cable where the zone starts
         "startSensor": 120,      → value of the sensor where the zone starts
         "endCable": 1,           → value of the cable where the zone ends
         "endSensor": 105         → value of the sensor where the zone ends
      },
      {
         "zone": 2,
         "startCable": 1,
         "startSensor": 104,
         "endCable": 1,
         "endSensor": 97
      },
        .
        .
        .
      {
         "zone": 10,              → zone value (⇔ ZONE10)
         "startCable": 2,
         "startSensor": 102,
         "endCable": 2,
         "endSensor": 120
      }
```

```
   ]
}
```

Note: the list matches the number of created zones. For example, if there are 3 created zones (ZONE1, ZONE3 and ZONE5), the list will contain 3 zones identified by zone 1, zone 3 and zone 5.

2.2.3 Retrieve number of impacts configuration

This request allows the user to retrieve the number of impacts configured for each zones.

Request: **GET** **http://<ip_address>/API/impact-settings**
Response content:

Data	Values	Comments
nbZone	[0 → 10]	Number of created zones
impactSettings	Tab containing the list of impact values settled on each zones	
impactValue	[1 → 10]	
zones	Tab containing the list of zones configured with a specific impact value	

Request example: **GET** **http://192.168.105.202/API/impact-settings**
Response example: **200**

```
{
   "nbZones": 10,                    → number of created zones
   "impactSettings": [
      {
         "impactValue": 1,           → it contains the number of impacts configured
         "zones": [1, 2…10]          → it contains the zones values configured to 1 impact
      },
      {
         "impactValue": 2,           → it contains the zones values configured to 2 impacts
         "zones": [4]
      },
      {
         "impactValue": 7,           → it contains the zones values configured to 7 impacts
         "zones": [8]
      }
   ]
}
```

2.2.4 Retrieve sensors sensitivity

This request allows the user to retrieve the sensors sensitivities configuration.

Request: **GET** **http://<ip_address>/API/sensitivity-settings**
Responses content:

Data	Values	Comments
Cable1 / Cable2	Details about the cable 1 or 2 sensitivity configuration	
configState	[0 ; 1] ⇔ [unconfigured ; configured]	
nbSensor	[1 → 120]	
sensitivitySettings	Tab containing all the sensitivity settled on each sensors	
sensitivityValue	[1 → 16]	1 is the less sensitive, 16 the most sensitive
sensors	List of sensors settled to this sensitivity value [1 → 120]	Max 120 sensors

Request example: **GET** **http://192.168.105.202/API/sensitivity-settings**
Response example: **200**

```
{
   "Cable1": {                          → cable 1 configuration details
      "configState": 1,                 → 1 the cable is configured, 0 the cable isn't configured
      "nbSensor": 120,                  → it contains the number of detected sensors
      "sensitivitySettings": [
         {
            "sensitivityValue": 10,
            "sensors": [1,2,3...70]     → list of sensors with a sensitivity settled to 10
         },
         {
            "sensitivityValue": 12,
            "sensors": [71,72...120]    → list of sensors with a sensitivity settled to 12
         }
      ]
   },
   "Cable2": {
      "configState": 1,
      "nbSensor": 120,
      "sensitivitySettings": [
         {
            "sensitivityValue": 12,
            "sensors": [1...120]        → list of sensors with a sensitivity settled to 12
         }
      ]
   }
}
```

2.2.5 Retrieve network configuration

This request allows the user to retrieve the network configuration.

Request: **GET** **http://<ip_address>/API/network**
Responses content:

Data	Values	Comments
ipaddr	Expected type: string – Format X.X.X.X	
mask	Expected type: string – Format X.X.X.X	
gwaddr	Expected type: string – Format X.X.X.X	
gateway	[true ; false]	
speed	[10/100 Mbps ; 10 Mbps]	
HTTPport		80 is the default parameter

Request example: **GET** **http://192.168.105.202/API/network**
Response example: **200**

```
{
   "ipaddr": "10.15.112.175",          → control unit ip address
   "mask": "255.255.240.0",            → subnet mask value
   "gwaddr": "10.15.127.254",          → gateway ip address
   "gateway": true,                    → true the gateway is enable, false it is disable
   "speed": "10/100 Mbps",             → transmission speed
   "HTTPport": 80                      → it contains the HTTP port value
}
```

Note: if gateway is settled to false, gwaddr isn't showed in the information list.

2.2.6 Retrieve date and time

This request allows the user to retrieve the control unit date and time.

Request: **GET** **http://<ip_address>/API/date-time**
Response content:

Données	Valeurs	Commentaire(s)
date	Expected type: string – Format YYYY-MM-DD	
time	Expected type: string – Format HH:MM:SS	

Request example: **GET** **http://192.168.105.202/API/date-time**
Response example: **200**

```
{
   "date": "2021-06-10",               → expected format YYYY-MM-DD
   "time": "12:29:53"                  → expected format HH:MM:SS
}
```

2.2.7 Retrieve relay states

This request allows the user to retrieve the 12 relays states.

Request: **GET** **http://<ip_address>/API/relay-states**

Response content;

Data	Values	Comments
relays_status	List containing the 12 relays states	
zone1 → zone10	[0 ; 1] ⇔ [no-alarm ; alarm]	
technical	[0 ; 1] ⇔ [no-alarm ; alarm]	
AP	[0 ; 1] ⇔ [no-alarm ; alarm]]	

Request example: **GET** **http://192.168.105.202/API/relay-states**
Response example: **200**

```
{
   "relays_status": {
      "zone1": 1,                    → 1 the relay is in alarm mode, 0 it isn't
      "zone2": 1,
      "zone3": 1,
      "zone4": 1,
      "zone5": 1,
      "zone6": 1,
      "zone7": 1,
      "zone8": 1,
      "zone9": 1,
      "zone10": 1,
      "technical": 1,
      "AP": 0
   }
}
```

Note: when the zones aren't created, the relays are in alarm mode.

2.2.8 Retrieve firmware version

This request allows the user to retrieve the version and name of the control unit.

Request: **GET** **http://<ip_address>/API/version**
Response content :

Data	Values	Comments
version	Expected type: string	
name	Expected type: string	
type	"gfence2400"	Fixed value

Request example : **GET** **http://<ip_address>/API/version**
Response example: **200**

```
{
   "version": "V 1.0  04/06/21        ",      → value and date of the version
   "name": "GFENCE 2400 ",                    → control unit name
   "type": "gfence2400"                       → product type
}
```

2.3 ALARMS HANDLING

2.3.1 Retrieve alarms

This request allows the user to retrieve the control unit alarm states.

Request: **GET** **http://<ip_address>/API/alarms**
Response content;

Datas	Values	Comments
nbZone	[0 → 10]	
intrusions	Details of the 10 zones' alarm states	
zone	[1 → 10]	
alarm	[0 ; 1] ⇔ [no-alarm ; alarm]	
triggeredSensor	Détail du câble et du capteur déclencheur + valeur du choc	This field isn't showed if there is an active intrusion alarm on the zone
cable	[1 ; 2]	
sensor	[1 → 120]	
sensitivity	[1 → 16]	
technicalAlarms	Détails	
cable1 / cable2	[0 ; 1] ⇔ [no-alarm ; alarm]	
triggeredSensorCable1 / triggeredSensorCable2	Details of the sensor where the technical default occurred (according to cable 1 or cable 2)	This fiel dis showed only if there is active technical default on the cable 1 or 2
sensor	[1 → 120]	
VBattDefault	[0 ; 1] ⇔ [no-alarm ; alarm]	
tamper	[0 ; 1] ⇔ [no-alarm ; alarm]	
aux1 / aux2	[0 ; 1] ⇔ [no-alarm ; alarm]	

Request example: **GET** **http://192.168.105.202/API/alarms**
Response example: **200**

```
{
   "nbZone": 10,                  → number of created zones
   "intrusions": [                → list of all the intrusion alarms on zones
      {
         "zone": 1,               → first zone
         "alarm": 0               → 1 the zone is in alarm, 0 it isn't
      },
      {
         "zone": 2,
         "alarm": 0
      },
      {
         "zone": 3,
         "alarm": 0
      },
      {
         "zone": 4,
```

```
            "alarm": 0
        },
        {
            "zone": 5,
            "alarm": 0
        },
        {
            "zone": 6,
            "alarm": 0
        },
        {
            "zone": 7,
            "alarm": 0
        },
        {
            "zone": 8,
            "alarm": 0
        },
        {
            "zone": 9,                          → ninth zone
            "alarm": 1,                         → ZONE9 is in alarm
            "triggeredSensor": {                → localization of the triggered sensor in the zone
                "cable": 2,                     → cable value
                "sensor": 1,                    → sensor value
                "sensitivity": 11               → shock value
            }
        },
        {
            "zone": 10,
            "alarm": 0
        }
    ],
    "technicalAlarms": {                        → technical alarms list
        "cable1": 0,                            → 1 technical default is active, 0 it isn't
        "cable2": 1,
        "triggeredSensorCable2": {              → localization of the triggered sensor on the cable 1
            "sensor": 1                         → value of the defected sensor
        },
        "VBattDefault": 0,                      → 1 power supply default is active, 0 it isn't
        "tamper": 1,                            → 1 tamper is open (alarm mode), 0 tamper is closed
        "aux1": 1,                              → 1 auxiliary is in alarm mode, 0 it isn't
        "aux2": 1
    }
}
```

Note: when auxiliaries aren't activated, they are in alarm mode.

2.3.2 Alarms acknowledgment

This is the only write request. It allows to acknowledge the intrusion alarms.

Request: **GET** **http://<ip_address>/update.cgi?type=11**

Request example: **GET** **http://192.168.105.202/update.cgi?type=11**
Response example: **200**

2.4 ERRORS HANDLING

Examples of known errors :

➔ Using incorrect login or password
Response: **401 Unauthorized : password required**

➔ Bad URL construction
Response: **404 File not found**

Example : **GET** **http://192.168.105.202/alarms**
*(expected URL is http://192.168.105.202/**API/**alarms)*

➔ Control unit internal error
Response: **500 Internal Server Error**

3 ALGORITHM EXAMPLE

In this part, we will see an example of algorithm to handle intrusion alarms.

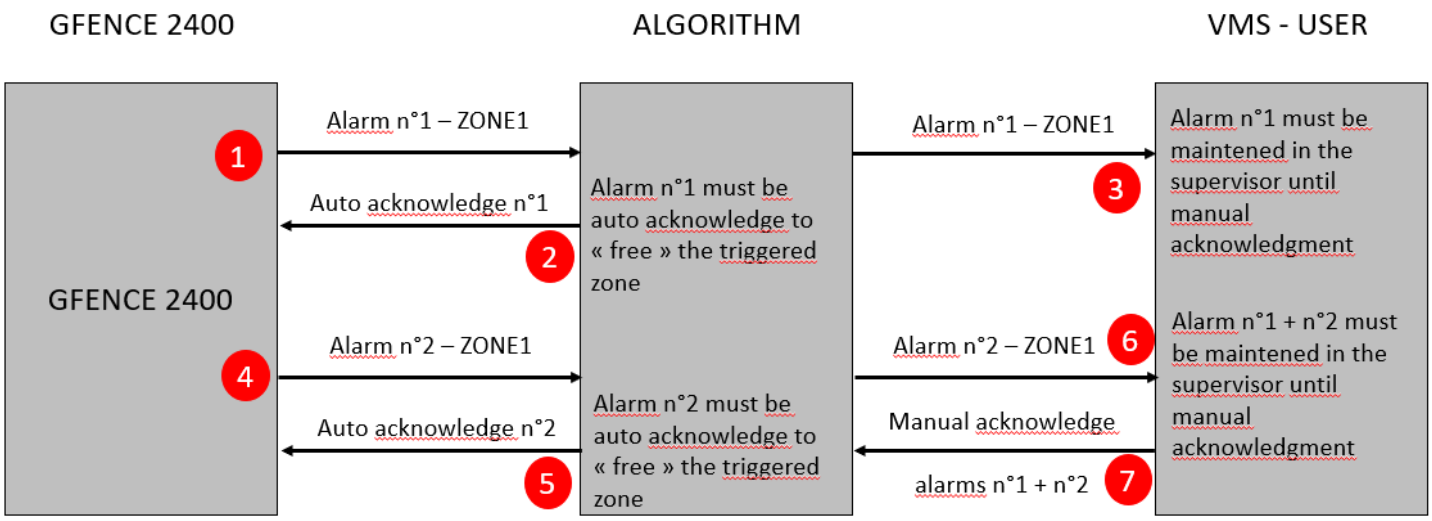
Zones intrusion alarms are memorized into the control unit until the user acknowledge the alarms from the Web pages. It means that in one specific zone, the most recent intrusion alarms won't be actualized until the oldest alarms is acknowledge. Other alarms on other zones will be detected as they are supposed to be.

Note: the most recent alarms are logged in the control unit events logs but not in the real time web page.

For an integration of the control unit, with HTTP requests, into a supervision system, it is advisable to create an automatic acknowledge as soon as an intrusion alarm appears in order to release the triggered zone. This automatic acknowledge allows to set the control unit back in to "listen mode" of a next intrusion alarm.

Therefore, the supervision system will have to maintain the alarm up until a real manual acknowledge from the user.

Below there is a scheme to explain the idea:



1. Retrieve a first alarm on ZONE1
   GET **http://<ip_address>/API/alarms**

2. Auto acknowledgment of the alarm on ZONE1 to release the zone
   GET **http://<ip_address>/update.cgi?type=11**

3. The first alarm must be hold into the supervisor until a manual acknowledgment

4. Retrieve a second alarm on ZONE 1
   GET **http://<ip_address>/API/alarms**

5. Auto acknowledgment of the alarm on ZONE1 to release the zone
   GET **http://<ip_address>/update.cgi?type=11**

6. Both alarms (1 + 2) must be hold until a manual acknowledgment by the user

7. Manual acknowledgment of both alarms by the user

In compliance with the European environmental directives, this product must not be thrown away but recycled in an appropriate subsidiary.