API ED3

Ce document présente les requêtes API de l'EcoDevices 3 de GCE Electronics.

L'API est disponible en ligne à l'url https://documenter.getpostman.com/view/34094433/2sBXc8o3Dq

Les requêtes API nécessitent une authentification pour fonctionner. L'authentification nécessaire est la clé API

La clé API se paramètre dans le menu Configuration de l'EcoDevices 3. Il faut récupérer la valeur paramétrée et ajouter aux requêtes API le paramètres "ApiKey={apikey}" où {apikey} désigne la valeur récupérée dans l'EcoDevices 3.

A noter : La clé API n'est pas lié à la session et peut donc être réutilisée sur différents périphériques.

Logs

L'EcoDevices 3 intègre une historisation de 30 compteurs avec :

- Enregistrement des index de compteurs et de prix toutes les 15 minutes sur 5 ans glissants.
- Enregistrement des valeurs moyennes, maximum et minimales des mesures instantanées toutes les 5 minutes sur 1 an glissant.

Les requêtes de cette catégorie permettent de modifier les compteurs et de consulter les logs.

Requête courbes

Ces requêtes permettent de récupérer, pour les 30 compteurs disponibles :

- Les indices de consommation et de prix sur 5 ans toutes les 15 minutes,
- Les valeurs minimums, maximums et moyennes sur 1 an toutes les 5 minutes.

Pour exploiter ces informations sont proposées 5 échelles de temps, représentées par le paramètre **preset**.

Ce preset comprend les valeurs **jour, semaine,mois,annee** et **total.**

Pour les preset de jour et de semaine, il est possible de consulter les valeurs instantanées en utilisant le paramètre **mode** avec les valeurs **min, max , moy** ou **index** pour obtenir les valeurs de consommation.

Pour les valeurs de consommation, il est possible de choisir entre la consommation énergétique et la dépense finançière avec le paramètre **unite** en écrivant **kwh ou euro.**

La date de début des enregistrements est à passer en tant que timestamp au format unix après le paramètre **debut.**

Les différents paramètres sont rappelés sous forme de tableau ici :

Paramètres	Valeurs possibles
preset	jour,semaine,mois,annee,total et default
mode	min,max,moy et index
unite	kwh et euro
debut	timestamp au format unix

Pour chaque preset, le nombre de points sont retournés en fonction du mode :

Preset	Mode instantanée	Mode Index
Jour	288	24
Semaine	287	21
Mois	/	24
Annee	/	13
Total	/	5
Default	288	288

Les paramètres sont à passer dans l'URL de la requête.

GET Requête moyenne jour

```
IP/api/object/core_logger/1?option=filter_logs&debut=1773739204&preset=jour&mode=moy&ApiKey=apikey
```

Cet exemple permet de récupérer les valeurs moyennes du compteur 1 de la journée du 17 mars 2026.

A noter : les logs sont retournés à partir de l'heure passée dans le timestamp, par exemple 10:20 ici.

PARAMS

option filter_logs

debut 1773739204

preset jour

mode moy

ApiKey=apikey

GET Requête index kwh jour

IP/api/object/core_logger/0?option=filter_logs&debut=1773739204&preset=jour&mode=index&unite=euro&ApiKey=apikey

Cet exemple permet de récupérer les logs d'indice de consommation du compteur 0 pour la journée du 17 mars 2026, à partir de 10h20.

PARAMS

option	filter_logs
debut	1773739204
preset	jour
mode	index
unite	euro
ApiKey=apikey	

GET Requête max semaine

IP/api/object/core_logger/0?option=filter_logs&debut=1773739204&preset=semaine&mode=max&AuthToken=ApiKey=apikey

Cette requête permet de récupérer les valeurs maximum instantanées du compteur 0 de la semaine, à partir du 17 mars 2026.

PARAMS

option	filter_logs
debut	1773739204
preset	semaine
mode	max
AuthToken	ApiKey=apikey

GET Requête total semaine

IP/api/object/core_logger/0?option=filter_logs&debut=1773739204&preset=semaine&mode=index&unite=kwh&ApiKey=apikey

Cet exemple permet de récupérer les logs d'index de consommation de la semaine, à partir du 17 mars 2026.

PARAMS

option	filter_logs
debut	1773739204
preset	semaine
mode	index
unite	kwh
ApiKey=apikey	

GET Requête mois

IP/api/object/core_logger/0?option=filter_logs&debut=1773739204&preset=mois&unite=kwh&ApiKey=apikey

Cet exemple permet de récupérer les indices de consommation à l'échelle du mois à partir du 17 mars 2026.

PARAMS

option	filter_logs
debut	1773739204
preset	mois
unite	kwh
ApiKey=apikey	

GET Requête annee

IP/api/object/core_logger/0?option=filter_logs&debut=1773739204&&mode=index&preset=annee&ApiKey=apikey

Cet exemple permet de récupérer les indices de consommation sur l'année , à partir du 17 mars 2026.

PARAMS

option	filter_logs

debut	1773739204
mode	index
preset	annee
ApiKey=apikey	

GET Request total

IP/api/object/core_logger/0?option=filter_logs&preset=total&unite=euro&ApiKey=apikey

Cet requête permet de récupérer les indices de consommation total sur 5 ans.

PARAMS

option	filter_logs
preset	total
unite	euro
ApiKey=apikey	

GET Requête all

IP/api/object/core_logger/11?option=filter_logs&debut=1773739204&preset=default&mode=index&unite=kwh&ApiKey=apikey

Cette requête permet de récupérer 288 logs à partir du timestamp passé en en paramètres.
Cette requête permet ainsi de récupérer tout les logs.

PARAMS

option	filter_logs
debut	1773739204
preset	default
mode	index
unite	kwh
ApiKey=apikey	

Objet Core Logger

Description de l'objet

Les Core Logger sont des objets faisant office de compteurs dans l'EcoDevice 3 pour la TIC, les entrées Tore et les entrées impulsionnelles.

A chaque compteur est associé une énergie. Les énergies possibles sont :

- L'électricité,
  - L'énergie consommée,
  - L'énergie produite,
  - L'énergie injectée,
  - L'énergie stockée.
- L'eau,
- Le gaz.

Les compteurs d'électricité peuvent prendre comme source la trame de téléinformation des compteurs LINKY (pour la consommation globale et l'énergie injectée sur le réseau) ainsi que les 8 entrées Tore.

Les compteurs d'eau et de gaz peuvent être liés uniquement aux entrées impulsionnelles.

Pour chaque compteurs, les enregistrements sont :

- En log long, toutes les 15 minutes, sur 5 ans glissant :
  - L'indice de consommation (en mL ou en wH),
  - L'indice de prix (en centimes).
- En log court, toutes les 5 minutes sur 1 an glissant :
  - La valeur moyenne de la puissance ou du débit depuis le dernier log,
  - La valeur minimal,
  - La valeur maximal.

Utilisation de l'objet

Pour pouvoir utiliser le Core Logger, il est nécessaire de suivre les étapes suivantes :

1. Dans le menu Configurations :
   a. Renseigner son tarif dans le menu Configurations.
   b. Paramétrer la source de la mesure en entrée.
2. Dans la page du poste dédiée, créer l'objet Core Logger en renseignant
   a. La fonction (voir ci-dessus),
   b. Le tarif associé,
   c. La mesure en source.

Corps de l'objet

Côté API, les objets Core Logger sont organisés de la manière suivante :

En configuration :

Champ	Type	Détail
name	Caractères	Taille max de 32
sourceMesure	Entier	ID de la source liée
linkTar	Entier	ID du tarif lié
icon	Entier	Numéro de l'icone

En Etats :

Champ	Type	Détail
index	entier	Indice totalisateur de consommation en wH
indexPrix	Entier	Indice totalisateur de prix en centimes d'euros
moyenne	Entier	Moyenne sur 5 minutes en W
instMax	Entier	Instantané Max, sur 5 minutes
instMin	Entier	Instantané Min, sur 5 minutes
unit	Entier	0 pour Wh, 1 pour litres
type	Entier	De 0 à 5, décrit ci dessous
tarif	Entier	Numéro du tarif lié

A la création d'un Core Logger, il est nécessaire de renseigner en corps le champ "type" permettant de définir la catégorie du compteurs. La correspondance est expliqué ci-dessous avec le nombre de ressource disponible pour chaque compteur.

Contrairement aux autres objets, l'attribution des ID est prédéfinie pour les Core Logger en fonction du champ type.

Numéro de type	Description	Nombre de compteurs	Plage d'ID
0	Consommation	8	11 à 18
1	Injection	1	19
2	Production	2	20 et 21
3	Stockage	3	22 à 24
4	Eau	3	25 à 27

5	Gaz	2	28 et 29

Enfin, 11 autres compteurs sont disponibles pour le traitement de la Téléinformation Linky, de l'indice 0 à 10.
Le compteur 0 sert de compteur totalisateur, les autres permettent d'accéder aux indices uniques propres aux options tarifaires.

La source de la mesure est un numéro d'identification GCE permettant de lier le compteur à une mesure physique (TORE, impulsion ou sortie TIC).

La clé sourceMesure permet de paramètrer cette mesure avec la correspondance suivante :

Element	ID du premier élément	**Nombre d'élément**
Tore	11534336	8
Pulse	11599872	4

La clé linkTar permet de lier un objet tarif via un numéro d'identification GCE à la consommation relevé par un Core Logger afin de calculer l'indice de prix.

GET Get Core Logger

IP/api/object/core_logger/32?ApiKey=apikey

Cette requête permet de récupérer la configuration d'un objet Core Logger avec :

- Son id,
- Son nom,
- La source associée,
- L'icon.

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
        "_id": 851968,
        "name": "BASE",
        "sourceMesure": 10354688,
        "linkTar": 1114112,
        "icon": 18,
        "unit": 0
```

```
}
```

GET Get Core Logger State

IP/api/object/core_logger?option=filter_state&ApiKey=apikey

Cette requête permet de récupérer les valeurs de tout les compteurs avec la correspondance suivante :

Clé JSON	Description
"_id"	Code d'identification GCE
"index"	Index, strictement croissant en wH ou en mL
"indexprix"	Index de prix, strictement croissant en centimes
"moyenne"	Valeur instantanée moyenne, sur 5 minutes
"instMax"	Valeur instantanée max, sur 5 minutes
"instMin"	Valeur instantanée min, sur 5 minutes

A noter : les valeurs instantanées sont mises à 0 toutes les 5 minutes. Les valeurs retournées ici ne sont pas sur 5 minutes glissantes.

PARAMS

option filter_state

ApiKey=apikey

DELETE Delete logger

IP/api/object/core_logger?ApiKey=apikey

Cette requête permet de supprimer un Core Logger.

Elle nécessite un redémarrage pour être appliquée.

PARAMS

ApiKey=apikey

POST Create Core Logger Conso with Tore 1

IP/api/object/core_logger?ApiKey=apikey

Cette requête permet de créer un objet Core Logger de type consommation, lié au tarif avec l'id "1114113" et lié à la source "11599872" qui correspond à l'entrée Tore 1.

Elle nécessite un redémarrage pour être appliquée.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "name": "Test",
    "sourceMesure": 11534336,
    "linkTar": 1114113,
    "icon": 17,
    "unit": 0,
    "type": 0
}
```

POST Create Core Logger TIC

IP/api/object/core_logger?ApiKey=apikey

Cette requête permet de créer les Core Logger liés à la TIC.

Elle nécessite un redémarrage pour être appliquée.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "name": "TIC",
    "sourceMesure": 1048576
```

```
}
```

POST Create Core Logger TIC Inj

IP/api/object/core_logger?ApiKey=apikey

Cette requête permet de créer un Core Logger de type Injection et de le lié avec la Téléinformation afin de récupérer les valeurs du compteur depuis la trame TIC.

Elle nécessite un redémarrage pour être appliquée.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "name": "Injection,
    "sourceMesure": 1048588,
    "type" : 1
}
```

POST Delete Core Logger Logs

IP/api/object/core_logger/0/eraseLogs?ApiKey=apikey

Cette requête permet de supprimer les enregistrements d'un Core Logger.

Cette action est définitive.

PARAMS

ApiKey=apikey

POST Factory reset logs

IP/api/object/core_logger/0/eraseAllLogs?ApiKey=apikey

Cette requête permet de supprimer les logs de tout les compteurs.

Cette action est définitive.

PARAMS

ApiKey=apikey

POST Raz Core Logger Index

IP/api/object/core_logger/0/eraseIndex?ApiKey=apikey

Cette requête permet de remettre à zéro l'indice de consommation d'un Core Logger.

A noter : cette action est inutile pour les compteurs de la TIC si le compteur Linky n'est pas remis à zéro.

Cette action est définitive.

PARAMS

ApiKey=apikey

PUT Put Tore as Source

IP/api/object/core_logger/28?ApiKey=apikey

Cette requête permet de mettre comme source du compteur 28 l'entrée Tore 2.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
        "sourceMesure": 11599873
}
```

PUT Put Pulse as Source

IP/api/object/core_logger/20?ApiKey=apikey

Cette requête permet de mettre comme source du compteur 20 l'entrée impulsionnelle 3.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
"sourceMesure" : 11534338
}
```

Object

Les Objets sont des éléments paramétrables par l'utilisateur, conçus pour étendre les fonctionnalités de l'appareil et réaliser des actions automatisées ou des interactions avec d'autres systèmes.

Les objets peuvent être utilisés dans le moteur de scénario, dans les notifications, ou pour piloter des extensions et périphériques connectés.

Vous pouvez également passer des éléments variables (valeurs de compteurs, ANA virtuelles, etc.) dans les corps des notifications ou emails en utilisant le format ID, où ID correspond au code d'identification de l'élément.

Récupération de variables dynamiques

L'objet Push et l'objet Mail permettent de récupérer des valeurs d'autres objets de l'EcoDevices 3 en utilisant le format $_id$ dans le corps des deux objets.

Cette section détaille les valeurs à renseigner pour chaque objet ainsi que la méthode pour les obtenir.

Le tableau ci-dessous donne les valeurs pour chaque données

ED3

L'objet EcoDevices3 permet de représenter les mesures physiques réalisées sur les entrées de la carte.

Les champs modifiables sont :

Champ	Type	**Détails**
name	Caractères	Taille max de 32
ID_Logger_Tore	Entier	Numéro d'identification GCE du compteur associé à chaque TORE
ConsoTore	Entier	Facteur de conversion des TORE
ToreName	Caractères	Tableau de 10, 10 caractères par tore
ID_Logger_Pulse	Entier	Numéro d'identification GCE du compteur associé à chaque entrée impulsionnelle
ConsoPulse	Entier	Valeur de consommation associée à un pulse pour chaque entrée
PulseName	Caractères	Tableau de 8, 10 caractères par pulse

GET Get ED3

IP/api/object/ed3/0?ApiKey=apikey

Cette requête permet de récupérer les mesures du pulse.

PARAMS

ApiKey=apikey

PUT Modify ED3

IP/api/object/ed3/0?ApiKey=apikey

Cette requête permet de modifier l'objet EcoDevices 3.

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "RelayName": [
        "Ouverture fermeture fenetre",
        "Porte"
    ]
}
```

PUT Relay 1 On

IP/api/object/ed3/0?ApiKey=apikey

Cette requête permet d'activer le relais 1 de l'EcoDevice 3.

A noter : la commande agit comme un ON et non comme un OFF.
Pour désactiver le relais, il faut donc transmettre une commande false.

L'état actuel du relais est transmis dans le champ "Relay_state".

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "Relay_command": [
        true,
        false
    ]
}
```

GET Get ED3 Plugin

IP/api/object/ed3/0?option=objDescriptor

Cette requête permet de récupérer les élements de l'objet ED3 utilisé par le plugin ED3 de l'IPX800 V5

PARAMS

option objDescriptor

Body raw (json)

json


```
{
    "_id": 1245184,
    "name": "ED30",
    "errorStatus": "0x0000",
    "ID_Logger_pulse": [
        0,
        0,
        0,
        0
    ],
    "ConsoPulse": [
```


GET Get ED3 State

IP/api/object/ed3/0?ApiKey=apikey&option=filter_state

Cette requête permet de filtrer les éléments de configuration et conserver uniquement les états des entrées et de la sortie.

PARAMS

ApiKey=apikey

option filter_state

Body raw (json)

json


```
{
    "_id": 1245184,
```

```
    "_id": 1245184,
    "name": "ED30",
    "errorStatus": "0x0000",
    "ID_Logger_pulse": [
        0,
        0,
        0,
        0
    ],
    "ConsoPulse": [
```


GET Toggle Relay 1

IP/api/object/ed3/0/ToggleR/1?ApiKey=apikey

Cette commande permet de faire commuter un relais.
L'exemple correspond au relais 1.
Pour faire commuter le relais 2, il faut remplacer le 1 par un 2

PARAMS

ApiKey=apikey

GET Set Relay 1 On

IP/api/object/ed3/0/SetOnR/1?ApiKey=apikey

Cette commande permet de passer l'état d'un relais à actif.
L'exemple correspond au relais 1.
Pour faire commuter le relais 2, il faut remplacer le 1 par un 2

PARAMS

ApiKey=apikey

GET Set Relay 1 Off

IP/api/object/ed3/0/SetOffR/1?ApiKey=apikey

Cette commande permet de passer l'état d'un relais à inactif.
L'exemple correspond au relais 1.
Pour faire commuter le relais 2, il faut remplacer le 1 par un 2

PARAMS

PARAMS

ApiKey=apikey

Tarif

Les objets tarifs permettent de paramétrer les tarifs appliqués sur les objets compteurs.

Ils sont paramétrables dans le menu Configuration.

Clé	Type	Détails
name	Caractères	Taille max = 32
value	Entier	A rentrer en centièmes de centimes (2531 pour 0,2135 €)
tarif	Entier	Unité du tarif, avec correspondance ci-dessous

Valeur de tarif	Type associé
0	Non défini
1	KWH
2	Litres

GET Get Tarif

IP/api/object/tarif?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
        "_id": 1114112,
        "name": "BASE EDF",
```

```
        "errorStatus": "0x0000",
        "value": 2516,
        "tarif": 1
    }
```


POST Create tarif

IP/api/object/tarif?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
        "name": "Test suppr",
        "value": 20
}
```


DELETE Delete Tarif

IP/api/object/tarif/1114122?ApiKey=apikey

PARAMS

ApiKey=apikey

PUT Modify Tarif

IP/api/object/tarif/14

Body raw (json)

json

```
{
    "name": "TARIF Freq1",
    "value": 10,
    "tarif": 4

}
```

Planning

Les objets planning sont des objets temporels qui activent leur sortie lorsque l'heure est située dans une plage horaire définie.

Le planning fonctionne selon deux récurrences :

- Soit une récurrence hebdomadaire (chaque semaine),
- Soit aucune récurrence.

Dans le cas de la récurrence hebdomadaire, l'utilisateur paramètre des créneaux horaires pour les jours de la semaine.
Si il ne paramètre pas de récurrence, il doit alors renseigner une date de début et une date de fin pour le planning.

Clé	**Type**	**Détails**
name	Caractères	Taille max = 32
errorstatus	Entier	A rentrer en centièmes de centimes (2531 pour 0,2135 €)
rec	Entier	Enumération avec correspondance ci-dessous
startTime	Tableau d'entier	Jour et heure de début du planning. Ex : Lundi 9h10 s'écrit 00910 Samedi 21h45 s'écrit 52145
endTime	Tableau d'entier	Jour et heure de fin du planning.
startDate	timestamp	Date de début du planning
endDate	timestamp	Date de fin du planning
Enable	Booléen	Activation du planning

Valeur de rec	Signification
0	Non défini
1	Pas de récurrence

2	Récurrence hebdomadaire

GET Get Planning

IP/api/object/planning?ApiKey=apikey

PARAMS

ApiKey=apikey

POST Create Planning

IP/api/object/planning?ApiKey=apikey

PARAMS

ApiKey=apikey

DELETE Delete Planning

IP/api/object/planning/0?ApiKey=apikey

PARAMS

ApiKey=apikey

PUT Modify PLanning

IP/api/object/planning/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
        "func": 2,
        "rec": 2,
        "startTime": [
        30000,
        65535,
        65535,
        65535,
        65535,
        65535,
        65535
```

Calendrier

Les objets Calendriers sont des objets temporels qui activent leur sortie lorsque la date est comprise dans un intervalle défini.

Il faut d'abord choisir la date et l'heure de début puis la date et l'heure de fin. A l'intérieur de cet interval, la sortie vaut 1.

Le planning fonctionne selon plusieurs récurrences :

- Soit une récurrence lié à une journée de la semaine,
- Soit une récurrence quotidienne,
- Soit une récurrence pendant les jours ouvrés,
- Soit les veilles de jours travaillés,
- Soit les Week End,
- Soit les veilles de jour de Week End,
- Soit chaque année,
- Soit aucune récurrence.

Clé	**Type**	**Détails**
name	Caractères	Taille max = 32
errorstatus	Entier	Enumération
mode	Entier	Enumération avec correspondance ci-dessous
start_date	String	Date au format JJ/MM/AAAA
startTime	String	Heure au format HH:MM:SS
endTime	String	Heure au format HH:MM:SS
end_date	String	Date au format JJ/MM/AAAA
ioEnable	Booléen	Activation du planning

ioState	Booléen	Valeur de sortie

Valeur de mode	Signification
0	Pas de répétition
1	Tous les lundi
2	Tous les mardi
3	Tous les mercredi
4	Tous les jeudi
5	Tous les vendredi
6	Tous les samedi
7	Tous les dimanche
8	Tous les jours
10	Tous les jours
20	Tous les jours travaillés (Lundi a Vendredi)
21	Tous les jours avant les jours travaillés (Du

GET Get Calendar

IP/api/object/calendar?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
        "_id": 1114112,
        "name": "BASE EDF",
        "errorStatus": "0x0000",
        "value": 2516,
        "tarif": 1
    }
```

POST Create Calendar

IP/api/object/calendar?ApiKey=apikey

PARAMS

ApiKey=apikey

DELETE Delete Calendar

IP/api/object/calendar/0?ApiKey=apikey

PARAMS

ApiKey=apikey

PUT Modify Calendar

IP/api/object/calendar/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "name": "Calendar Z",
    "mode": 1,
    "start_date": "17/09/2025",
    "start_time": "16:00:49",
    "end_date": "20/09/2025",
    "end_time": "19:03:49",
    "ioEnable": true,
    "ioState": false
}
```

TIC

L'objet TIC permet d'enregistrer et de traiter les informations récupérées par la trame de téléinformation des compteurs Linky.

Cet objet détecte automatiquement le mode de fonctionnement du compteur, récupére les indices, les instantanées et les informations sur le tarif en cours.

Les valeurs numériques sont transmises sous forme d'entiers (index, tension,...).
Les autres éléments sont transmis sous forme de texte (Abonnement, tarif).

GET Get TIC State

IP/api/object/tic/0?option=filter_state?ApiKey=apikey

Cette requête permet de consutler la trame de téléinformations détectée.

PARAMS

option	filter_state?ApiKey=apikey

Body raw (json)

json


```
{
    "_id": 1048576,
    "ADCO": "021628336005",
    "OPTARIF": "EJP.",
    "ISOUSC": 15,
    "BASE": 0,
    "HCHC": 0,
    "HCHP": 0,
    "EJPHN": 261673,
    "EJPHPM": 720831,
    "PPPHCJB": 0
```


GET Get TIC config

IP/api/object/tic?ApiKey=apikey

Cette requête permet d'obtenir les informations de configuration de l'objet téléinformation.

Elle permet de récupérer les informations sur le tarif avec le type de tarif choisi et l'option tarifaire en cours.

Le tableau ID Compteur permet de lier les données de la téléinformations avec les compteurs de l'EcoDevice 3.

PARAMS

ApiKey=apikey

Body raw (json)

json


```
[
    {
        "_id": 1048576,
        "name": "TIC N°0",
        "Tarif": "EJP",
        "Option": "EJPHN",
        "mode": "HISTORIQUE",
        "ID_Compteur": [
            0,
            0,
            0
```


Push

L'objet Push permet d'envoyer des requêtes HTTP ou Push vers d'autres systèmes ou API.

Les champs à paramétrer sont :

- l'adresse IP de la cible du push,
- le port de la cible,
- l'username et le password si nécessaire,
- la méthode d'accès,
- le mode d'authentificiation,
- l'activation de la sécurité,
- l'activation de la réduction du retour,
- l'URI de la requête,
- Le corps de la requête.

Les champs du push sont décrits ci - dessous :

Champ	Type	**Détails**
name	Caractères	Max = 32
addressIP	Caractères	Max = 256

port	Entier	Port du destinaire du push
username	Caractères	Taille max = 32
password	Caractères	Taille max = 32
Method	Caractères	Taille max = 8, type de méthode : GET,POST,PUT,DELETE
authMode	Caractères	Taille max = 8, mode d'authentification : NONE,BASIC, DIGEST
tlsMode	booléen	Activation de la sécurité TLS
minify	booléen	Paramètres permettant de réduire la taille de la réponse en supprimant les éléments

Les valeurs des codes d'erreur sont définies ici :

Code	Correspondance
0	En attente
1	Envoi réussi
2	Elements manquants
3	Erreur d'attribution MemPool
4	Erreur initialisation TLS
5	Erreur envoi de la requête

POST Create Push

IP/api/object/push?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
```

```
        "addressIP": "192.168.1.106",
        "SendPush": true,
        "Uri": "api/object/core_logger?ApiKey=CGWPaGLaePdnqL3"
    }
```


GET Get Push

IP/api/object/push?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
    {
        "_id": 1179648,
        "name": "Push N°0",
        "addressIP": "192.168.1.227",
        "port": 80,
        "username": "",
        "password": "",
        "method": "PUT",
        "authMode": "NONE",
        "tlsMode": false
```


PUT Modify Push

IP/api/object/push/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
```

```
{
        "_id": 1179649,
        "name": "Push N°1",
        "addressIP": "",
        "port": 80,
        "username": "",
        "password": "",
        "method": "GET",
        "authMode": "NONE",
        "tlsMode": false,
        "minify": false
```


DELETE Delete Put

IP/api/object/push/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
    "_id": 1179648,
    "name": "Push N°1",
    "addressIP": "http://192.168.1.227",
    "port": 80,
    "username": "",
    "password": "",
    "method": "GET",
    "authMode": "BASIC",
    "tlsMode": false,
    "minify": false
```


PUT Send Push

IP/api/object/push/0?ApiKey=apikey

Cette requête permet d'activer l'envoi d'un Push.

PARAMS

ApiKey=apikey

Body raw (json)

```
json

{
    "SendPush": true
}
```

Mail

Cet objet permet d'envoyer des mails depuis un serveur SMTP à paramétrer dans la partie Système.

Les champs paramétrables sont le nom, le ou les destinataires, le sujet et le corps du mail.

Le mail peut être paramétré avec un sujet par défaut intitulé "Mon EcoDevice 3".

Un même mail peut être envoyé à 4 destinataires différents.

Champ	Type	Détails
name	Caractères	Taille max : 32
SendMail	Booléen	Envoi un mail quand vaut vrai
defaultSub	Booléen	Activation du sujet par défaut
Destinataire 1	Caractères	Taille max : 50
Destinataire 2	Caractères	Taille max : 50
Destinataire 3	Caractères	Taille max : 50
Destinataire 4	Caractères	Taille max : 50
Body	Caractères	Taille max : 100, corps du message
subject	Caractères	Taille max : 30, sujet du message
mailStatus	Code de statut	Code de statut du mail

Les valeurs de code de mailStatus possible sont détaillées ici :

Valeur	Signification
0	Attente
1	Envoi OK
2	Erreur nom du serveur

3	Erreur initialisation TLS
4	Erreur initialisation timeout
5	Erreur ID/Mot de passe
6	Erreur type de contenu
7	Erreur sujet
8	Erreur Entête
9	Erreur écriture du corps
10	Erreur fermeture du corps
11	Erreur connection au serveur SMTP

POST Create Mail

IP/api/object/mail?ApiKey=apikey

PARAMS

ApiKey=apikey

GET Get Mail

IP/api/object/mail?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
[
    {
        "_id": 1376256,
        "name": "MAIL test",
        "errorStatus": "0x0000",
```

```
        "SendMail": false,
        "defaultSub": false,
        "Destinataire 1": "",
        "Destinataire 2": "",
        "Destinataire 3": "",
        "Destinataire 4": ""
```

PUT Modify Mail

IP/api/object/mail/0

Body raw (json)

json

```
{
        "name": "MAIL test",
        "defaultSub": false,
        "SendMail": true,
        "Destinataire 1": "monMail@hebergeur.com",
        "Body":"Ce test permet de tester le mail",
        "Subject":"Test ED3"
}
```

DELETE Delete Mail

IP/api/object/mail/0?ApiKey=apikey

PARAMS

ApiKey=apikey

PUT Send Mail

IP/api/object/mail/0?ApiKey=apikey

Cette requête permet d'envoyer le mail paramétré.

PARAMS

ApiKey=apikey

Body raw (json)

```
json

{
        "SendMail": true
}
```

AnaVirtuelles

Les AnaVirtuelles sont des objets permettant d'enregistrer des valeurs.

POST Create ANA

IP/api/object/anaVirtuelle?ApiKey=apikey

PARAMS

ApiKey=apikey

GET Get ANA

IP/api/object/anaVirtuelle?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

```
json

    {
        "_id": 1572865,
        "name": "ANA_VIRT N°1",
        "errorStatus": "0x0000",
        "value": 0.0
```

```
}
```

PUT Modify ANA

IP/api/object/anaVirtuelle/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "value": 1.61
}
```

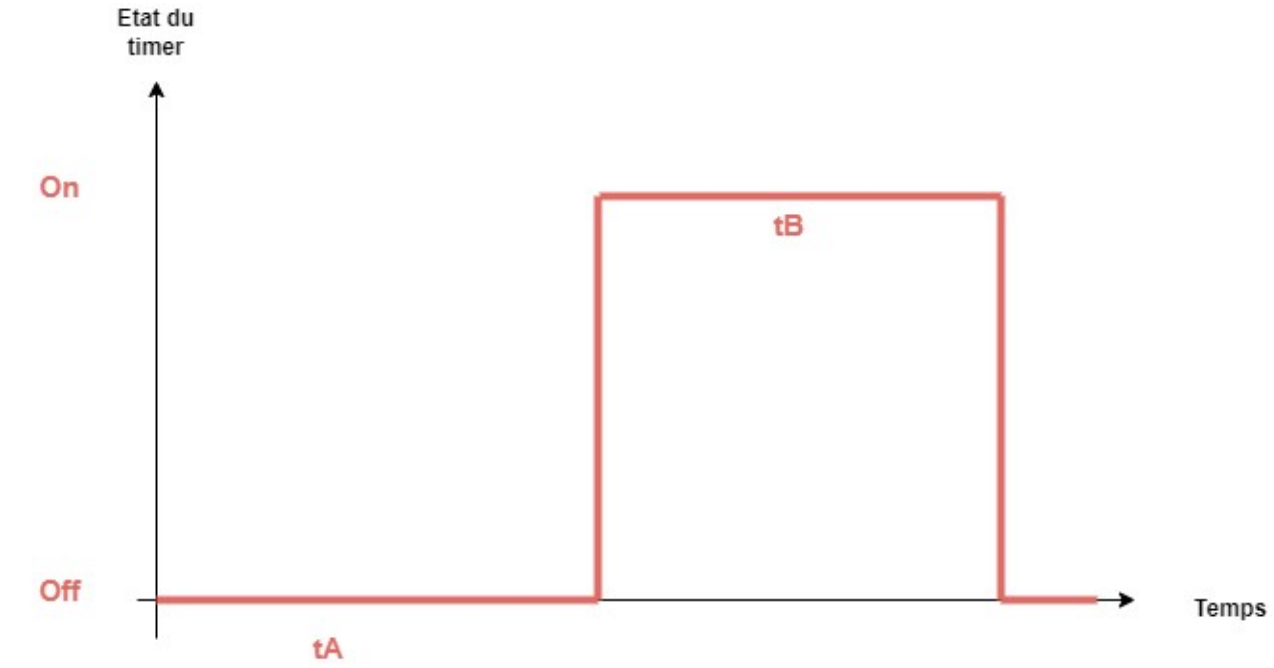
DELETE Delete ANA

IP/api/object/anaVirtuelle/0

Timer

L'objet Timer permet de générer un signal bas (0 logique ou OFF) pendant un certain temps puis un signal haut (1 logique ou ON) pendant un second temps . Les paramètres de cet objet sont listés ci-dessous :

Paramètres	Type	Signification
bSecond	Booléen	Echelle de temps en secondes si vrai, en millisecondes sinon
bStart	Booléen	Activation du timer
bState	Booléen	Etat du timer. Vaut false pour OFF et true pour ON.
taValue	Entier	Temps du signal bas
tbValue	Entier	Temps du signal haut



A la fin du signal haut, l'objet se désactive : il faut remettre le champ bStart à Vrai pour l'activer.

GET Get Timer

IP/api/object/timer/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
    "_id": 1638400,
    "name": "PULSE N°0",
    "errorStatus": "0x0000",
    "bSecond": false,
    "bStart": false,
    "bState": false,
    "taValue": 20,
    "tbValue": 2
}
```

POST Create Timer

IP/api/object/timer?ApiKey=apikey

PARAMS

ApiKey=apikey

PUT Modify Timer

IP/api/object/timer/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "name": "PULSE test",
    "bSecond": false,
    "bStart": true,
    "bState": false,
    "taValue": 5000,
    "tbValue": 5000
}
```

DELETE Delete Timer

IP/api/object/timer/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Extensions

Cette section est dédiée aux objets permettant d'utiliser les extensions GCE Electronics.

Les extensions inclues sont :

- le XDisplay,
- le XTHL,
- le X8R.

X8R

Le X8R est une extension permettant de piloter 8 relais.

Les champs API de l'objet sont les suivants :

Information	Type	**Description**
codeExt	Texte	Code extension unique
pushDelais	Entier	Temps nécessaire pour qu'un appui soit considéré comme appui long
bConnect	Booléen	Paramètres X8R Connect
LongPush	Booléen	Appui long détecté
state	Booléen	Etat des sorties
command	Booléen	Commande des sorties

GET Get X8R

IP/api/ebx/x8r?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
[
    {
        "_id": 4259840,
        "name": "X8R-N80"
```

```
"name": "X8R N°0",
"codeExt": "20ab1211",
"errorStatus": "0x0000",
"pushDelais": [
    10,
    0,
    0,
    0
```


PUT Modify X8R

IP/api/ebx/x8r/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
    "codeExt": "2000047E",
    "bConnect": true
}
```


POST Create X8R

IP/api/ebx/x8r

DELETE Delete X8R

IP/api/ebx/x8r

GET Toggle X8R Relay

IP/api/ebx/x8r/0/ToggleR/1?ApiKey=apikey

Cette commande permet de faire commuter un relais.
L'exemple correspond au relais 1.
Les valeurs possibles vont de 1 à 8

Les valeurs possibles vont de 1 à 8

PARAMS

ApiKey=apikey

GET Set X8R Relay On

IP/api/ebx/x8r/0/SetOnR/1?ApiKey=apikey

Cette commande permet de passer à l'état actif un relais.
L'exemple correspond au relais 1.
Les valeurs possibles vont de 1 à 8

PARAMS

ApiKey=apikey

GET Set X8R Relay Off

IP/api/ebx/x8r/0/SetOffR/1?ApiKey=apikey

Cette commande permet de passer à l'état inactif un relais.
L'exemple correspond au relais 1.
Les valeurs possibles vont de 1 à 8

PARAMS

ApiKey=apikey

XTHL

Le XTHL est une extension intégrant un capteur de température, de luminosité et d'humidité.

Le seul paramètre à renseigner est le code d'extension unique pour le paramétrer.

Les différentes mesures sont retournées avec des double.

GET Get XTHL

IP/api/ebx/xthl?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
[
    {
        "_id": 4325376,
        "name": "XTHL N°35",
        "codeExt": "0",
        "errorStatus": "0x0000",
        "bVersion2": false,
        "Humidite": 0,
        "Luminosite": 0,
        "Temperature": 0.0,
        "isFault": false
```


PUT Modify XTHL

IP/api/ebx/xthl/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
    "bVersion2": true
}
```


POST Create XTHL

IP/api/ebx/xthl?ApiKey=apikey

PARAMS

ApiKey=apikey

DELETE Delete XTHL

IP/api/ebx/xthl/0

XDISPLAY

L'extension XDisplay est un écran tactile permettant d'afficher des données et de piloter des sorties.

Couplé avec l'EcoDevices 3, il permet d'afficher 4 écrans :

- Deux écrans pour piloter chaque relais de l'EcoDevices 3,
- Un écran avec les indices totalisateurs d'énergie,
- Un écran avec les valeurs instantanées d'énergie.

Les clés API sont décrites ci-dessous :

Paramètre	Type	Description
codeExt	Texte	Code d'extension unique

L'EcoDevice 3 crée automatiquement ses écrans sur le XDisplay lorsqu'on l'ajoute. Une route API existe pour recharger les écrans si besoin.

GET Get XDISPLAY

IP/api/ebx/xdisplay?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
[
    {
        "_id": 4390912,
        "name": "XDISPLAY N°0",
        "codeExt": "0b70017b",
        "autoOff": 0,
        "sensitive": 15,
        "screensType": [
            11,
            11,
            0
```


PUT Modify XDISPLAY

IP/api/ebx/xdisplay/0?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json


```
{
    "bRefresh": true,
    "consoName": "Conso",
    "consoEnergy": "Energy"
}
```


POST Create XDISPLAY

IP/api/ebx/xdisplay?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

json

```
{
    "codeExt": "0b000101"
}
```

DELETE Delete XDISPLAY

IP/api/ebx/xdisplay/0?ApiKey=apikey

PARAMS

ApiKey=apikey

POST Recharger les écrans

IP/api/ebx/xdisplay/0/createScreens?ApiKey=apikey

Cette requête permet de recréer les écrans sur le XDisplay ciblé.

PARAMS

ApiKey=apikey

Plugins

Cette catégorie décrit les requêtes API pour accéder aux plugins intégrés dans l'EcoDevice 3

Weather

L'EcoDevices récupère les informations météo à partir de l'API gratuite d'OpenWeather Map. Les clés API sont présentées ici :

Paramètres	Description
apiKey	Clé api.
bEnable	Activation du plugin météo

position	Position GPS des relevés météo
sunriseTime	Heure du levée de soleil au format unix
sunsetTime	Heure du couchée du soleil au format unix
isDay	Information sur s'il fait Jour ou Nuit.
temperature	Température
pressure	Pression
humidity	Taux d'humidité
windSpeed	Vitesse du vent
clouds	Prévisions nuageuse.
levels	Statut météo (ensoleillé, pluvieux, etc)

Au niveau des entrées, l'utilisateur doit passer sa clé API OpenWeatherMap, la position GPS du relevés et il doit activer le plugin Weather.

Nom	Type de données	Détails
apiKey	40 Caractères	Clé API
bEnable	Booléen	Activation du plugin
Position	30 Caractères	Coordonnées GPS

GET Get Weather

IP/api/plugin/weather?ApiKey=apikey

Cette requête permet de récupérer les données traitées par GCE suite à un relevé météo.

PARAMS

ApiKey=apikey

PUT Modify Weather

IP/api/plugin/weather?ApiKey=apikey

PARAMS

ApiKey=apikey

Body raw (json)

```
json

{
    "apiKey": "MyApiKey",
    "bEnable": true,
    "position": "48.856614, 2.3522219"
}
```

Modbus

L'EcoDevice 3 intègre un gestionnaire Modbus, permettant d'interagir avec les équipements compatibles Modbus.
L'API permet d'accéder aux objets paramétrés et de modifier certaines caractéristiques.

GET Get Object

IP/api/object/modbus/0?ApiKey=apikey

La requête GET permet de récupérer :

- le nom de l'objet
- les paramétres de conversion,
- la fréquence de lecture
- Les états liés à chaque commande.

Ces états sont représentes par un tableau d'IO contenant des booléens et un tableau d'ANA contenant des valeurs.

PARAMS

ApiKey=apikey

Body raw (json)

```
json

{
```

```
    "ana": [
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0
```


GET Export

IP/api/object/modbus/0?ApiKey=apikey&option=filter_modbus

Cette requête permet de récupérer un fichier compatible avec l'utilitaire GCE pour le Modbus, permettant ainsi de modifier la configuration.

PARAMS

ApiKey=apikey

option	filter_modbus

Body raw (json)

json


```
{
    "io": [
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
```


PUT Ecriture

IP/api/object/modbus/0

Cette requête montre comment réaliser une écriture dans le cas où l'on veut transmettre la commande N°3 et qu'elle est de type IO.

Body raw (json)

json


```
{
    "io": [
        false,
        false,
        true,
        false,
        false,
        false,
        false,
        false,
        false
```