

V2 - (READ) /api/v2/hosts

Sommaire

- Objectifs
- Paramètres
- Réponse
 - Codes de retour
 - Retour du code 200
 - Retour du code 400
 - Paramètres POST incorrects
 - Paramètre inconnu
 - Messages d'erreurs des filtres (filterX)
 - Filtre inexistant
 - Filtre incomplet
 - Filtre incorrect
 - Valeur incorrecte pour ce type de filtre
 - Opérateur incorrect
 - Argument incorrect
 - Messages d'erreurs liés aux paramètres de tri (sort)
 - Propriété inconnu
 - Ordre de tri incorrect
 - Messages d'erreurs liés au paramètre de format du résultat (output_format)
 - Valeur incorrecte
 - Messages d'erreurs lors du paramétrage des propriétés présentes dans la sortie (output_field)
 - Propriété de sortie inexistante
 - Erreurs communes lors de l'envoi de la requête
 - Messages d'erreurs liés au protocole HTTPS
 - Le certificat SSL a été refusé
 - Requête HTTP sur un serveur en HTTPS
 - Requête HTTPS sur un serveur en HTTP

Objectifs

Méthode POST de type READ qui permet de récupérer des données d'hôtes et éventuellement de ses checks, comme sur l'interface de visualisation :

- Filtrées (optionnel)
- Triées
- Rangées :
 - En arbres (hôtes/clusters => checks)
 - Tous au même niveau

Paramètres

Cet appel utilise les 4 paramètres suivants :

- **filterX**
 - Pour la propriété **type**, le fait d'utiliser des valeurs différentes de **host** et **check_host** est interdit sur cette route.
- **sort**
- **output_format**
- **output_field**

Voir la page [V2 - Les paramètres des API du broker-module-livdata](#) , pour leur description complète.

Réponse

Codes de retour

Codes de retour	Explications
200	OK
400	Paramètre invalide
401	Accès nécessite une authentification ou un Token valide.
403	Authentification de l'utilisateur OK, mais droits non suffisant.
500	L'appel est valide, mais un problème d'exécution est rencontré.

Retour du code 200

Les propriétés retournées doivent être choisies avec l'option **output_field**.

Mais les propriétés suivantes seront au minimum automatiquement retournées :

- **nb_element**
- **type** (si le paramètre **output_format** vaut *elements_on_same_level*)
- **father_uuid**
- **father_name**
- par check :
 - **check_uuid**
 - **check_name**

Voir la page [V2 - Les propriétés présentes dans le retour 200 des API du broker-module-livedata](#) , pour la description complète de toutes les propriétés pouvant être renvoyée.

Suivant le paramètre **output_format** (*checks_attached_to_father* / *elements_on_same_level*)

- (présentation du format de retour ci-dessus, mais se référer à l'exemple pour le format exact)

checks attached to father:

- **request_statistics** :
 - ...
- **elements_found** :
 - **hosts** :
 - **host 1:**
 - **father_uuid** : *text*
 - **father_name** : *text*
 - ...
 - **checks** :
 - **check_uuid1** : *text*, **check_name1** : *text*, ...
 - **check_uuid2** : *text*, **check_name2** : *text*, ...
 - ...
 - ...

```
curl -s -S -H 'x-api-token: XYZ' \  
-d "output_format=checks_attached_to_father" \  
http://broker-module-livedata:50100/api/v2/hosts
```

Exemple de sortie attendue :

elements on same level:

- **request_statistics** :
 - ...
- **elements_found**:
 - **elem1**:
 - **type**: *host*
 - **father_name**: *text*
 - **father_uuid** : *text*
 - ...
 - **elem2**:
 - **type**: *check*
 - **check_uuid** : *text*
 - **check_name**: *text*
 - **father_name** : *text*
 - **father_uuid** : *text*
 - ...
 - **elem3**:
 - **type**: *check*
 - **check_uuid** : *text*
 - **check_name**: *text*
 - **father_name** : *text*
 - **father_uuid** : *text*
 - ...
 - ...

```
curl -s -S -H 'x-api-token: XYZ' \  
-d "output_format=elements_on_same_level" \  
http://broker-module-livedata:50100/api/v2/hosts
```

Exemple de sortie attendue :

<http://localhost:50100/api/v2/hosts/> 

<http://localhost:50100/api/v2/hosts/>

```
{
  "request_statistics": {
    "nb_elements_total": 9,
    "nb_hosts_total": 4,
    "nb_checks_total": 4,
    "nb_elements_filtered": 7,
    "nb_hosts_filtered": 4,
    "nb_checks_filtered": 3
  },
  "elements_found": {
    "clusters": [],
    "hosts": [
      {
        "father_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe",
        "father_name": "Bordeaux",
        "checks": [
          {
            "check_name": "CPU Stats",
            "check_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe-
c296d75e5ad911e58cc5080027f08538"
          },
          {
            "check_name": "Disks Stats",
            "check_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe-
c29735965ad911e58cc5080027f08538"
          }
        ]
      },
      {
        "father_uuid":
"8db8b5f0ba1611eba60d0800277faebe",
        "father_name": "Lyon",
        "checks": []
      },
      {
        "father_uuid":
"76f45d80ba1e11eba2670800277faebe",
        "father_name": "Nantes",
        "checks": [
          {
            "check_name": "CPU Stats",
            "check_uuid":
"76f45d80ba1e11eba2670800277faebe-
c296d75e5ad911e58cc5080027f08538"
          }
        ]
      },
      {
        "father_uuid":
"e5460dc2ba1611eb81580800277faebe",
        "father_name": "Rennes",
        "checks": []
      }
    ]
  }
}
```

```
{
  "request_statistics": {
    "nb_elements_total": 9,
    "nb_hosts_total": 4,
    "nb_checks_total": 4,
    "nb_elements_filtered": 7,
    "nb_hosts_filtered": 4,
    "nb_checks_filtered": 3
  },
  "elements_found": [
    {
      "father_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe",
      "father_name": "Bordeaux",
      "type": "host"
    },
    {
      "check_name": "CPU Stats",
      "check_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe-
c296d75e5ad911e58cc5080027f08538",
      "father_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe",
      "father_name": "Bordeaux",
      "type": "check_host"
    },
    {
      "check_name": "Disks Stats",
      "check_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe-
c29735965ad911e58cc5080027f08538",
      "father_uuid":
"2c6dcf1aba1611ebaa7d0800277faebe",
      "father_name": "Bordeaux",
      "type": "check_host"
    },
    {
      "father_uuid":
"8db8b5f0ba1611eba60d0800277faebe",
      "father_name": "Lyon",
      "type": "host"
    },
    {
      "father_uuid":
"76f45d80ba1e11eba2670800277faebe",
      "father_name": "Nantes",
      "type": "host"
    },
    {
      "check_name": "CPU Stats",
      "check_uuid":
"76f45d80ba1e11eba2670800277faebe-
c296d75e5ad911e58cc5080027f08538",
      "father_uuid":
"76f45d80ba1e11eba2670800277faebe",
      "father_name": "Nantes",
      "type": "check_host"
    },
    {
      "father_uuid":
"e5460dc2ba1611eb81580800277faebe",
      "father_name": "Rennes",
      "type": "host"
    }
  ]
}
```

Retour du code 400

Paramètres POST incorrects

Paramètre inconnu

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "parametre_inconnu=is_status:true" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: POST parameter [ parametre_inconnu ] is unknown
```

ERROR 400: POST parameter [parametre_inconnu] is unknown

Messages d'erreurs des filtres (filterX)

(voir la page [V2 - Les paramètres des API du broker-module-livadata](#))

Filtre inexistant

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "filter01=is_status:true" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: filtering[0]: invalid field name [ is_status_ ]
```

ERROR 400: filtering[0]: invalid field name [is_status_]

Filtre incomplet

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "filter0=next_check" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: filtering[0]: missing value for field [ next_check ]
```

ERROR 400: filtering[0]: missing value for field [next_check]

Filtre incorrect

Valeur incorrecte pour ce type de filtre

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "filter0=status:9" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: filtering[0]: field [ status ] => wrong value ['9']
```

ERROR 400: filtering[0]: field [status] => wrong value ['9']

Opérateur incorrect

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "filter0=next_check:avant" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: filtering[0]: field [ next_check ]
unknown date constraint [ avant ]
```

ERROR 400: filtering[0]: field [next_check] unknown date constraint [avant]

Argument incorrect

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "filter0=next_check:in-less-than|hier" \
http://broker-module-livadata:50100/api/v2/hosts
ERROR 400: filtering[0]: field [ next_check ] => [ in-less-than ] => invalid literal for int() with base 10: 'hier'
```

ERROR 400: filtering[0]: field [next_check] => [in-less-than] => invalid literal for int() with base 10: 'hier'

Messages d'erreurs liés aux paramètres de tri (sort)

(voir la page [V2 - Les paramètres des API du broker-module-livadata](#))

Propriété inconnu

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "sort=host" \
http://broker-module-livedata:50100/api/v2/hosts
ERROR 400: sort: invalid field name [ host ]
```

ERROR 400: sort: invalid field name [host]

Ordre de tri incorrect

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "sort=father_name:big" \
http://broker-module-livedata:50100/api/v2/hosts
ERROR 400: sort: invalid sort direction [ big ]
for field [ father_name ]
```

ERROR 400: sort: invalid sort direction [big] for field [father_name]

Messages d'erreurs liés au paramètre de format du résultat (output_format)

(voir la page [V2 - Les paramètres des API du broker-module-livedata](#))

Valeur incorrecte

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "output_format=vrai" \
http://broker-module-livedata:50100/api/v2/hosts
ERROR 400: output_format: invalid value [ vrai ]
```

ERROR 400: output_format: invalid value [vrai]

Messages d'erreurs lors du paramétrage des propriétés présentes dans la sortie (output_field)

(voir la page [V2 - Les paramètres des API du broker-module-livedata](#))

Propriété de sortie inexistante

```
$ curl -s -S -H "x-api-token: XYZ" \
-d "output_field=is_status_" \
http://broker-module-livedata:50100/api/v2/hosts
ERROR 400: output_field: invalid field name [
is_status_ ]
```

ERROR 400: output_field: invalid field name [is_status_]

Erreurs communes lors de l'envoi de la requête

Messages d'erreurs liés au protocole HTTPS

Le certificat SSL a été refusé

```
$ curl -s -S -H "x-api-token: XYZ" \
https://broker-module-livedata:50100/api/v2/inventory
curl: (60) SSL certificate problem: unable to get
local issuer certificate
...
```



Cela peut se produire si le certificat du serveur est auto-signé.

Pour contourner la validité de l'émetteur du certificat, il faut utiliser l'option **--insecure** (*option courte* : **-k**).

```
curl --insecure -s -S -H "x-api-token: XYZ" \  
https://broker-module-livedata:50100/api/v2  
/inventory
```

Avec curl v7.29.0 :

```
curl: (60) Peer's Certificate issuer is not recognized.
```

...

Avec curl v7.60 et supérieur :

```
curl: (60) SSL certificate problem: unable to get local issuer certificate
```

...

Requête HTTP sur un serveur en HTTPS

```
$ curl -s -S -H "x-api-token: XYZ" \  
http://broker-module-livedata:50100/api/v2/inventory  
The client IP_SERVEUR:PORT_CLIENT sent a plain HTTP  
request,  
but this server only speaks HTTPS on this port.
```

```
The client IP_SERVEUR:PORT_CLIENT sent a plain HTTP request,  
but this server only speaks HTTPS on this port.
```

Requête HTTPS sur un serveur en HTTP

```
$ curl -s -S -H "x-api-token: XYZ" \  
https://broker-module-livedata:50100/api/v2/inventory  
curl: (35) error:1408F10B:SSL routines:  
ssl3_get_record:wrong version number
```

Avec curl v7.29.0 :

```
curl: (35) SSL received a record that exceeded the maximum  
permissible length.
```

Avec curl v7.60 et supérieur :

```
curl: (35) error:1408F10B:SSL routines:ssl3_get_record:wrong version  
number
```