# V1 - ( READ ) /api/v1/sla/ -- OPTIONNEL --

**Sommaire**



## Objectifs

Le module de type **livedata_module_sla_provider** permet par le biais d'une URL ( *Méthode POST de type READ* ) de recevoir la liste des données SLA de tous les éléments ( ***hôtes, clusters et checks*** ) :

- Filtrées ( *optionnel* ),
- Rangées,
  - En arbres ( *hôtes/clusters  checks* ),
  - Tous au même niveau,
  - Seulement les données SLA des éléments demandés.
- En choisissant,
  - les informations présentes dans le retour de la requête ( *optionnel* ),
  - la période sur laquelle les données SLA seront récupérées ( *optionnel* ),
    - les données SLA sont calculées à la fin de la journée, donc **la dernière donnée disponible est celle d'hier**,
  - le nombre d'éléments par page ( *optionnel* ).
- Les données SLA récupérées sont triées dans l'ordre chronologie ( du plus récent au plus vieux ).

**Pré-requis**

Le module de type **livedata_module_sla_provider** doit être activé sur le **broker-module-livedata** pour que la route **/api/v1/sla/** soit accessible.

La configuration du module se trouve par défaut dans le fichier suivant : **/etc/shinken/modules/livedata-module-sla-provider.cfg** : Module livedata-module-sla-provider

## Paramètres

Pour définir l'appel, 5 paramètres sont disponibles :

| Nom et format | Valeur par défaut | Description et Syntaxe |
|---|---|---|
| ***filterX=expression~expression~...*** | Aucun filtre défini => tous les éléments sont retournés | • **~** ayant le sens de "et"<br>• ***expression*** de la forme : ***critère:valeur0^^valeur1***<br>  ◦ où **^^** a le sens de "ou" |
| ***output_format=format_de_sortie_choisi*** | Les éléments ( Hôtes, Clusters et Checks ) sont au même niveau ( *elements_on_same_level* ) | 3 valeurs disponibles :<br>• **checks_attached_to_father**<br>• **elements_on_same_level**<br>• **list_of_sla** |
| ***output_field=champs1~champ2~...*** | Spécifique à chaque output_format | • **~** ayant le sens de "et"<br>• ***champ*** étant un des champs disponible en sortie de la requête |
| ***period=start:start_date~end:end_date*** | Aucune période définie => Les données SLA de la veille sont retournées<br><br>Les données SLA sont calculées à la fin de la journée, donc **la dernière donnée disponible est celle d'hier** | • **~** ayant le sens de "et"<br>• date au format DD_MM_YYYY |
| ***page_settings=page:page_number~size:page_size*** | Aucune valeur précisée => tous les éléments sont retournés | • **~** ayant le sens de "et"<br>• la première page débute à l'index 0 |

### filterX ( Filtres )

Les filtres ont pour formes :

- ***filterX = expression~expression***
  - **~** ayant le sens de "et"
  - ***expression*** de la forme : ***critère:valeur0^^valeur1***
    - où **^^** a le sens de "ou"
- **X** vaut de **0 à 9.**
- Chaque élément correspondant à **au moins un des filtres** sera retourné.

Liste des **critères** et leurs **valeurs possibles:**

| Nom | Matching | Type | Valeur | Exemple d' expression |
|---|---|---|---|---|
| **type** | correspond exactement | *Tableau* | **HOST, CLUSTER, CHECK, CHECK_HOST, CHECK_CLUSTER** | **CHECK^^HOST** |
| **father_name** | contient ( insensible à la casse ) | *Tableau* | father_name est le critère contenant le filtre pour le nom des hôtes et des clusters. Si vous désirez filtrer des hôtes avec une chaîne et des clusters avec un autre, il vous faut faire 2 filtres avec type=HOST et type=CLUSTER | **bordeaux^^rennes** |

| | | | | |
|---|---|---|---|---|
| **father_name_contains** | contient ( insensible à la casse ) | *Tableau* | Permet de lister des chaines que l'on cherche dans les hôtes ou des clusters.<br><br>Si vous désirez filtrer des hôtes avec une chaîne et des clusters avec un autre, il vous faut faire 2 filtres avec type=HOST et type=CLUSTER. | **bor^^renn** |
| **father_uuid** | correspond exactement | *Tableau* | UUID des hôtes / clusters | **4a893fbbcb0047b8a1922bce91e3dfdg** |
| **check_name** | contient ( insensible à la casse ) | *Tableau* | Permet de lister les noms de checks exacts recherchés. | **CPU Usage^^Disk Usage** |
| **check_name_contains** | contient ( insensible à la casse ) | *Tableau* | Permet de lister les chaines que l'on cherche dans les noms de check. | **CPU^^Disk** |
| **check_uuid** | correspond exactement | *Tableau* | UUID des checks | **4a893fbbcb0047b8a1922bce91e3decf** |
| **address** | contient ( insensible à la casse ) | *Chaîne* | | **192.168.1.20** |
| **realm** | correspond exactement | *Tableau* | nom complet des royaumes | **Paris^^Bordeaux^^CORSE** |
| **notification_contacts** | correspond exactement | *Tableau* | nom complet de modèle de l'utilisateur | **user1^^user2** |
| **notification_contact_groups** | correspond exactement | *Tableau* | nom complet de groupe d'utilisateur | |
| **father_templates** | correspond exactement | *Tableau* | nom complet de modèle d'hôte ou de cluster | **linux^^http** |
| **host_groups** | correspond exactement | *Tableau* | nom complet de groupe d'hôte | **ERP_bordeaux** |
| **business_impact** | correspond exactement | *Entier* | 0, 1, 2, 3, 4, 5 | **4** |

Exemple filtrant les hôtes et les clusters ayant pour royaume ( **realm** ) Paris ou Bordeaux et comme impact métier ( **business_impact** ) 5

```
"filter0=type:cluster^^host~realm:Paris^^Bordeaux~business_impact:5"
```

## output_format ( Format de retour de la requête )

Ce paramètre permet de définir quelle format de retour est utilisé ( Il en existe 3 ):

- **checks_attached_to_father** : les checks sont accrochés à leurs hôtes / clusters ( *forme d'arbre* )
- **elements_on_same_level** : les checks sont listés au niveau des hôtes / clusters ( *une liste* )
- **list_of_sla** : Seul les données SLA sont listées ( *une liste* )

REMARQUE : Dans le cas ou le filtre vaut uniquement **type**=check ( *donc pas d'hôtes ou clusters* )

- Si le **output_format** = *checks_attached_to_father* , les hôtes / clusters seront quand même présents pour les checks correspondant à ce filtre.
- Si le **output_format** = *elements_on_same_level* ou **output_format** = *list_of_sla,* les hôtes et clusters ne sont pas présents.

Par défaut, la valeur est "**elements_on_same_level"**

## output_field ( Informations présentes dans le retour de la requête )

Ce paramètre permet de lister les champs qui seront affichés dans le résultat.

- Les champs présents par défaut sont :

| Nom | | Format | Description |
|---|---|---|---|
| **father_name** | 1 valeur | | **"bordeaux-storage"** |
| **father_uuid** | 1 valeur | | **"4a893fbbcb0047b8a1922bce91e3dfdg"** |
| **check_name** | 1 valeur | | **"CPU Stats"** |
| **check_uuid** | 1 valeur | | **"4a893fbbcb0047b8a1922bce91e3decf"** |

| | | | |
|---|---|---|---|
| **sla_total** | | Secondes | Temps total de SLA ( *86400 secondes étant 1 journée complète* ) |
| **sla_missing** | | Secondes | Temps en statut **Données manquantes** |
| **sla_ok** | | Secondes | Temps en statut **OK** |
| **sla_inactive** | | Secondes | Temps en statut **Shinken Inactive** |
| **sla_unknown** | | Secondes | Temps en statut **Inconnu** |
| **sla_crit** | | Secondes | Temps en statut **Critique** |
| **sla_warn** | | Secondes | Temps en statut **Attention** |
| **sla_thresholds** | | Liste de pourcentages | Deux pourcentages :<br>• la première valeur est le seuil d'avertissement<br>• la deuxième valeur est le seuil de critique<br>Les pourcentages ont une précision à 3 chiffres ( *ex: 90.001* ) |
| **sla_date** | | Chaîne de caractères | au format **aaaa_mm_jj** ( *ex: 2021_05_12* ) |
| | | | |

- Les champs présents peuvent être les suivants :

| Nom | | Valeur possible | Exemple d'expression |
|---|---|---|---|
| **type** | 1 valeur | **HOST**, **CLUSTER**, **CHECK_HOST**, **CHECK_CLUSTER** | **"check_host"** |

Le champ type fait partie des champs présents par défaut pour les formats de retour "*elements_on_same_level*" et "*list_of_sla*".

Exemple de définition du paramètre ouput_field :

```
"output_field=type"
```

**L'ordre dans lequel sont cités les champs ne change pas le format de sortie.**

## period ( entre quelles dates de début et de fin, prendre les données SLA )

| Nom | Valeur par défaut | Description et syntaxe |
|---|---|---|
| **period=start:date~end:date** | La date de la veille | Défini la période où collecter les données SLA<br>• Les dates sont au format **aaaa_mm_jj ( ex: 2021_05_12 )**<br>• Si **start** n'est **pas précisé**, cela signifie que le début de la période est la date de la veille.<br>• Si **end** n'est **pas précisé**, la fin de la période est égal à la date de début. |

Trois règles devront être respectées :

- La date de départ ne peut pas être antérieur à la date de la première donnée SLA disponible.
- La date de départ ne peut pas être supérieur à la date de la veille.
- La date de fin ne peut pas être supérieur à la date de début.

## page_settings ( combien d'éléments par page et quelle page retourner )

L'API peut, grâce à ce paramètre, définir le nombre d'éléments par page et le numéro de la page retournée, ce qui permet de contrôler le volume d'échange de données. Ceci est possible vu que les données SLA sont figées en base de données.

Le champ **has_next_page** dans la partie ***pagination*** du retour permet de savoir s'il y a une page suivante.

| Nom | Valeur par défaut | Info |
|---|---|---|
| **page_settings=*page*:*page_index*~*nb_element*:*size*** | Le nombre d'éléments par défaut d'une page est **100** | • ***nb_element*** étant la taille de la page<br>• ***page*** étant l'index de page demandée. Les indexes de page commencent à 0 |

Si **output_format** est à *checks_attached_to_father* , le nombre d'éléments par page correspondra aux hôtes / clusters.

Si **output_format** est à *elements_on_same_level* , le nombre d'éléments par page correspondra aux hôtes / clusters / checks.

Si **output_format** est à *list_of_sla*, le nombre d'éléments par page correspondra aux données SLA des hôtes / clusters / checks.

### Exemple

Exemple permettant d'obtenir la première page d'une requête renvoyant 100 éléments avec leurs données SLA, du début de l'année 2021 au 1er mai 2021.

```
curl -s -S -H "x-api-token: XYZ" \
 -d "period=start:2021_01_01~end:2021_05_01" \
 -d "page_settings=page:0~nb_element:100" \
http://broker-module-livedata:50100/api/v1/sla
```

Exemple permettant d'obtenir la quatrième page d'une requête renvoyant 100 éléments avec leurs données SLA, du début de l'année 2021 au 1er mai 2021.

```
curl -s -S -H "x-api-token: XYZ" \
 -d "period=start:2021_01_01~end:2021_05_01" \
 -d "page_settings=page:4~nb_element:100" \
http://broker-module-livedata:50100/api/v1/sla
```

## Exemple complet d'appel

Exemple par Appel curl :

```
curl -s -S -H "x-api-token: XYZ" \
 -d "filter0=check_name:CPU Stats" \
 -d "filter1=check_name:Disks" \
 -d "output_format=checks_attached_to_father" \
 -d "output_field=type" \
 -d "period=start:2021_05_01~end:2021_05_25" \
 -d "page_settings=page:4~nb_element:10" \
http://broker-server:50100/api/v1/sla
```

```
-s, alias de --silent,  ne pas afficher les barres de progression, n'affiche que les données récupérées
-S, alias de --show-error, afficher quand même les messages d'erreurs
-H, alias de --header, inclure ce header à la requête HTTP
-d, alias de --data, envoie les données spécifiées en requête POST au serveur HTTP
```

# 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 suffisants. |
| 500 | L'appel est valide, mais un problème d'exécution est survenu. |

## Retour du code 200

### Données statistiques

En premier apparaîtra des informations donnant le nombre d'éléments :

#### Compteurs obtenus avant d'effectuer la requête

| Nom | Type | Description |
|---|---|---|
| **nb_elements_total** | *Entier* | Nombre d'éléments supervisés visibles par l'utilisateur |
| **nb_hosts_total** | *Entier* | Nombre total d'hôtes visibles par l'utilisateur |
| **nb_clusters_total** | *Entier* | Nombre total de clusters visibles par l'utilisateur |
| **nb_checks_total** | *Entier* | Nombre total de checks visibles par l'utilisateur |

#### Compteurs obtenus après le filtrage des éléments

| Nom | Type | Description |
|---|---|---|
| **nb_elements_filtered** | *Entier* | Nombre d'éléments supervisés obtenus après application des filtres reçus en paramètres |
| **nb_hosts_filtered** | *Entier* | Nombre d'hôtes obtenus après application des filtres reçus en paramètres |
| **nb_clusters_filtered** | *Entier* | Nombre de clusters obtenus après application des filtres reçus en paramètres |
| **nb_checks_filtered** | *Entier* | Nombre de checks obtenus après application des filtres reçus en paramètres |

#### Compteurs liés à des filtres ne donnant pas le résultat attendu

| Nom | Type | Description |
|---|---|---|
| **nb_elements_not_found** | *Entier* | Nombre d'éléments dont un filtre explicite (voir ci-dessous) ne donne pas le retour attendu |
| **nb_father_not_found** | *Entier* | Nombre d'hôtes et de clusters pour lesquels un filtre **father_name** ou **father_uuid** ne donne pas d'élément en résultat |
| **nb_checks_not_found** | *Entier* | Nombre de checks pour lesquels un des filtres<br>• **check_uuid**<br>• **father_name** et **check_name**<br>• **father_uuid** et **check_name**<br>ne donne pas d'élément en résultat |

#### Compteurs liés au nombre d'éléments présent dans la page

| Nom | Type | Description |
|---|---|---|
| **nb_elements_in_page** | *Entier* | Nombre d'éléments présent dans la page |
| **nb_host_in_page** | *Entier* | Nombre d'hôtes présent dans la page |
| **nb_cluster_in_page** | *Entier* | Nombre de clusters présent dans la page |
| **nb_check_in_page** | *Entier* | Nombre de checks présent dans la page |
| **nb_sla_in_page** | *Entier* | Nombre de données SLA présent dans la page |

### Données de pagination

En deuxième, les données de pagination vont être retournées dans le format suivant :

| Nom | Type | Description |
|---|---|---|
| **has_next_page** | *Booléen* | Indication sur l'existence d'une page suivante |
| **nb_total_page** | *Entier* | Nombre total de pages |

| | | |
|---|---|---|
| **page** | *Entier* | Numéro de la page |
| **page_size** | *Entier* | Nombre d'éléments dans la page |

### Données propres aux éléments

| Nom | Type | Description |
|---|---|---|
| **type** | Chaîne de charactère | Type de l'élément ( *host, cluster, check_host, check_cluster* **)** |
| **father_name** | Chaîne de charactère | Nom de l'hôte / cluster |
| **father_uuid** | Chaîne de charactère | UUID de l'hôte |
| **check_name** | Chaîne de charactère | Nom du check |
| **check_uuid** | Chaîne de charactère | UUID du check |
| | | |

### Format de retour de la requête

Les champs présents pour chaque élément retourné possédant des données SLA doivent être choisis avec l'option **output_field,** mais les champs suivants sont au minimum automatiquement retourné :

**Output_format à checks_attached_to_father :**

- **elements_found**
  - **hosts**
    - **father_name**
    - **father_uuid**
    - **checks**
      - **check_name**
      - **check_uuid**
      - **sla**
        - **sla_total**
        - **sla_missing**
        - **sla_ok**
        - **sla_inactive**
        - **sla_unknown**
        - **sla_crit**
        - **sla_**

**Output_format à elements_on_same_level :**

- **elements_found**
  - **type**
  - **father_name**
  - **father_uuid**
  - **check_name**
  - **check_uuid**
  - **sla**
    - **sla_total**
    - **sla_missing**
    - **sla_ok**
    - **sla_inactive**
    - **sla_unknown**
    - **sla_crit**
    - **sla_warn**
    - **sla_thresholds**
    - **sla_date**
- **elements_not_found**

**Output_format à list_of_sla :**

- **elements_found**
  - **type**
  - **father_name**
  - **father_uuid**
  - **check_name**
  - **check_uuid**
  - **sla_total**
  - **sla_missing**
  - **sla_ok**
  - **sla_inactive**
  - **sla_unknown**
  - **sla_crit**
  - **sla_warn**
  - **sla_thresholds**
  - **sla_date**
- **elements_not_found**

- warn
    - sla_thresholds
    - sla_date
- clusters
    - father_name
    - father_uuid
    - checks
        - check_name
        - check_uuid
        - sla
            - sla_total
            - sla_missing
            - sla_ok
            - sla_inactive
            - sla_unknown
            - sla_crit
            - sla_warn
            - sla_thresh

    - **old_sla_date**
- **elements_not_found**

### Exemple 1 : output_format=checks_attached_to_father

```
curl -s -S -H "x-api-token: XYZ" \
 -d "period=start:2021_05_24~end:2021_05_25" \
 -d "output_format=checks_attached_to_father" \
 -d "filter01=type:check" \
 -d "page_settings=page:0~size:2" \
http://broker-module-livedata:50100/api/v1/sla
```


```
{
    "request_statistics": {
        "nb_elements_total": 34,
        "nb_hosts_total": 3,
        "nb_clusters_total": 1,
        "nb_checks_total": 30,
        "nb_elements_filtered": 4,
        "nb_hosts_filtered": 1,
        "nb_clusters_filtered": 1,
        "nb_checks_filtered": 2,
        "nb_elements_in_page": 4,
        "nb_hosts_in_page": 1,
        "nb_clusters_in_page": 1,
        "nb_checks_in_page": 2,
        "nb_sla_in_page": 4
    },
    "pagination": {
        "has_next_page": true,
        "nb_total_page": 2,
        "page": 0,
        "page_size": 2
    },
    "elements_found": {
        "clusters": [{
                "father_uuid": "12760f56bc6d11eb85a3080027c44e8f",
                "father_name": "Cluster 01",
                "checks": [{
                        "check_name": "Check Cluster 01",
                        "check_uuid": "12760f56bc6d11eb85a3080027c44e8f-9d86c522bd3511ebb58c080027c44e8f",
                        "sla": [{
                                "sla_date": "2021_05_24",
                                "sla_total": 86400,
                                "sla_warn": 0,
                                "sla_unknown": 0,
                                "sla_thresholds": [99.0, 97.0],
                                "sla_missing": 0,
                                "sla_ok": 0,
                                "sla_inactive": 86400,
                                "sla_crit": 0
                            }, {
                                "sla_date": "2021_05_25",
                                "sla_total": 47616,
                                "sla_warn": 0,
                                "sla_unknown": 0,
                                "sla_thresholds": [99.0, 97.0],
                                "sla_missing": 0,
                                "sla_ok": 0,
                                "sla_inactive": 47616,
```

```
                                "sla_crit": 0
                            }
                        ]
                    }
                ]
            }
        ],
        "hosts": [{
                "father_uuid": "f87c2e56b94b11ebaf7e080027c44e8f",
                "father_name": "Host 01",
                "checks": [{
                        "check_name": "Check 01",
                        "check_uuid": "f87c2e56b94b11ebaf7e080027c44e8f-fdd0c038b94b11ebb21f080027c44e8f",
                        "sla": [{
                                "sla_date": "2021_05_24",
                                "sla_total": 86400,
                                "sla_warn": 0,
                                "sla_unknown": 0,
                                "sla_thresholds": [99.0, 97.0],
                                "sla_missing": 0,
                                "sla_ok": 0,
                                "sla_inactive": 86400,
                                "sla_crit": 0
                            }, {
                                "sla_date": "2021_05_25",
                                "sla_total": 86400,
                                "sla_warn": 0,
                                "sla_unknown": 0,
                                "sla_thresholds": [99.0, 97.0],
                                "sla_missing": 0,
                                "sla_ok": 0,
                                "sla_inactive": 86400,
                                "sla_crit": 0
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```


**Exemple 2 : output_format=elements_on_same_level**

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_05_24~end:2021_05_25" \
-d "output_format=elements_on_same_level" \
-d "filter01=type:check" \
-d "page_settings=page:0~size:2" \
http://broker-module-livedata:50100/api/v1/sla
```


```
{
    "request_statistics": {
        "nb_elements_total": 34,
        "nb_hosts_total": 3,
        "nb_clusters_total": 1,
        "nb_checks_total": 30,
        "nb_elements_filtered": 30,
        "nb_hosts_filtered": 0,
        "nb_clusters_filtered": 0,
        "nb_checks_filtered": 30,
        "nb_elements_in_page": 2,
        "nb_hosts_in_page": 0,
        "nb_clusters_in_page": 0,
        "nb_checks_in_page": 2,
        "nb_sla_in_page": 4
```

```
    },
    "pagination": {
        "has_next_page": true,
        "nb_total_page": 15,
        "page": 0,
        "page_size": 2
    },
    "elements_found": [{
            "check_name": "Check Cluster 01",
            "type": "check_cluster",
            "father_uuid": "12760f56bc6d11eb85a3080027c44e8f",
            "father_name": "Cluster 01",
            "check_uuid": "12760f56bc6d11eb85a3080027c44e8f-9d86c522bd3511ebb58c080027c44e8f",
            "sla": [{
                    "sla_date": "2021_05_24",
                    "sla_total": 86400,
                    "sla_warn": 0,
                    "sla_unknown": 0,
                    "sla_thresholds": [99.0, 97.0],
                    "sla_missing": 0,
                    "sla_ok": 0,
                    "sla_inactive": 86400,
                    "sla_crit": 0
                }, {
                    "sla_date": "2021_05_25",
                    "sla_total": 51527,
                    "sla_warn": 0,
                    "sla_unknown": 24183,
                    "sla_thresholds": [99.0, 97.0],
                    "sla_missing": 3916,
                    "sla_ok": 0,
                    "sla_inactive": 23428,
                    "sla_crit": 0
                }
            ]
        }, {
            "check_name": "Check 01",
            "type": "check_host",
            "father_uuid": "f87c2e56b94b11ebaf7e080027c44e8f",
            "father_name": "Host 01",
            "check_uuid": "f87c2e56b94b11ebaf7e080027c44e8f-fdd0c038b94b11ebb21f080027c44e8f",
            "sla": [{
                    "sla_date": "2021_05_24",
                    "sla_total": 86400,
                    "sla_warn": 0,
                    "sla_unknown": 0,
                    "sla_thresholds": [99.0, 97.0],
                    "sla_missing": 165,
                    "sla_ok": 27344,
                    "sla_inactive": 58891,
                    "sla_crit": 0
                }, {
                    "sla_date": "2021_05_25",
                    "sla_total": 86400,
                    "sla_warn": 0,
                    "sla_unknown": 0,
                    "sla_thresholds": [58.654, 32.451],
                    "sla_missing": 245,
                    "sla_ok": 27881,
                    "sla_inactive": 58274,
                    "sla_crit": 0
                }
            ]
        }
    ]
}
```


**Exemple 3 : output_format=list_of_sla**

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_05_241~end:2021_05_25" \
-d "output_format=list_of_sla" \
-d "filter01=type:check" \
-d "page_settings=page:0~size:2" \
http://broker-module-livedata:50100/api/v1/sla
```


```
{
    "request_statistics": {
        "nb_elements_total": 34,
        "nb_hosts_total": 3,
        "nb_clusters_total": 1,
        "nb_checks_total": 30,
        "nb_elements_filtered": 30,
        "nb_hosts_filtered": 0,
        "nb_clusters_filtered": 0,
        "nb_checks_filtered": 30,
        "nb_elements_in_page": 1,
        "nb_hosts_in_page": 0,
        "nb_clusters_in_page": 0,
        "nb_checks_in_page": 1,
        "nb_sla_in_page": 2
    },
    "pagination": {
        "has_next_page": "true",
        "nb_total_page": 30,
        "page": 0,
        "page_size": 2
    },
    "elements_found": [{
            "sla_total": 86400,
            "sla_thresholds": [99.0, 97.0],
            "check_name": "Check Cluster 01",
            "check_uuid": "12760f56bc6d11eb85a3080027c44e8f-9d86c522bd3511ebb58c080027c44e8f",
            "father_uuid": "12760f56bc6d11eb85a3080027c44e8f",
            "sla_ok": 0,
            "sla_inactive": 86400,
            "sla_warn": 0,
            "sla_date": "2021_05_24",
            "father_name": "Cluster 01",
            "sla_unknown": 0,
            "sla_missing": 0,
            "type": "check_cluster",
            "sla_crit": 0
        }, {
            "sla_total": 51527,
            "sla_thresholds": [99.0, 97.0],
            "check_name": "Check Cluster 01",
            "type": "check_cluster",
            "father_uuid": "12760f56bc6d11eb85a3080027c44e8f",
            "sla_ok": 0,
            "sla_inactive": 23428,
            "sla_crit": 0,
            "sla_date": "2021_05_25",
            "father_name": "Cluster 01",
            "sla_unknown": 24183,
            "sla_missing": 3916,
            "check_uuid": "12760f56bc6d11eb85a3080027c44e8f-9d86c522bd3511ebb58c080027c44e8f",
            "sla_warn": 0
        }
    ]
}
```


## 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-livedata:50100/api/v1/sla
```

ERROR 400: POST parameter [ parametre_inconnu ] is unknown

**Messages d'erreurs des filtres ( filterX )**

Filtre inexistant

```
curl -s -S -H "x-api-token: XYZ" \
-d "filter01=is_status_:true" \
http://broker-module-livedata:50100/api/v1/sla
```

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-livedata:50100/api/v1/sla
```

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

Filtre existant, mais non autorisé

```
curl -s -S -H "x-api-token: XYZ" \
-d "filter01=is_status_confirmed:true" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: filtering[0]: field name [is_status_confirmed] not allowed in this route

Valeur incorrecte pour ce type de filtre

```
curl -s -S -H "x-api-token: XYZ" \
-d "filter0=business_impact:cinq" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: filtering[0]: field [ business_impact ] => wrong value [u'cinq']

**Messages d'erreurs du format de retour de la requête ( output_format )**

Valeur invalide

```
curl -s -S -H "x-api-token: XYZ" \
-d "output_format=all_elements" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: output_format: invalid value [ all_elements]

**Messages d'erreurs des champs présents dans la sortie ( output_field )**

Champ de sortie existant, mais non autorisé

```
curl -s -S -H "x-api-token: XYZ" \
-d "output_field=is_status_confirmed" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: output_field: field name [is_status_confirmed] not allowed in this route

Champ de sortie inexistant

```
curl -s -S -H "x-api-token: XYZ" \
-d "output_field=is_status_" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: output_field: invalid field name [is_status_]

**Messages d'erreurs de la période sur laquelle récupérer les données SLA ( period )**

Date de début inférieure à la date de la plus ancienne donnée SLA enregistrée dans la base MongoDB

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_01_01~end:2021_05_25" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: period: the start period is not valid, as there is no SLA data for this date. You can filter elements from 2021_02_13.

Date de fin supérieur à la date de la veille

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_01_01~end:2021_05_26" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: period: the end period is invalid, as the requested period is in the future. You can filter elements until 2021_05_25.

Date de fin inférieur à la date de début saisie

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_05_25~end:2021_05_24" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: period: the end period is invalid, as it's less than the start period.

Date de début est au mauvais format

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:01_05_2021~end:2021_05_01" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: period: start period wrong format. Expecting ISO format YYYY_MM_DD.

Date de fin est au mauvais format

```
curl -s -S -H "x-api-token: XYZ" \
-d "period=start:2021_05_01~end:25_05_2021" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: period: end period wrong format. Expecting ISO format YYYY_MM_DD.

**Messages d'erreurs de la pagination ( page_settings )**

Mauvaise valeur pour le paramètre page

```
curl -s -S -H "x-api-token: XYZ" \
-d "page_settings=page:cinq~size:2" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: Wrong value:[ cinq ] for page number parameter

Mauvaise valeur pour le paramètre size

```
curl -s -S -H "x-api-token: XYZ" \
-d "page_settings=page:5~size:deux" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: Wrong value:[ deux ] for page size parameter

Pas de valeur pour le paramètre size

```
curl -s -S -H "x-api-token: XYZ" \
-d "page_settings=page:5~size" \
http://broker-module-livedata:50100/api/v1/sla
```

ERROR 400: Missing page size parameter

Paramètre size manquant

ERROR 400: Wrong format for pagination parameters, expected: [name:integer~name:integer], got:[ page:2 ]

```
curl -s -S -H "x-api-token: XYZ" \
-d "page_settings=page:5" \
http://broker-module-livedata:50100/api/v1/sla
```