Cisco Secure Workload OpenAPI

OpenAPI fournit une API REST pour les fonctionnalités Cisco Secure Workload.

Authentification OpenAPI

OpenAPI utilise un schéma d'authentification basé sur un condensé. Le flux de travail est le suivant :

1. Connectez-vous au tableau de bord de l'interface utilisateur Cisco Secure Workload.
2. Générer une clé API et un code secret API avec les capacités souhaitées.
3. Utilisez le SDK API Cisco Secure Workload pour envoyer des requêtes REST au format JSON.
4. Pour utiliser le SDK Python, installez-le à l'aide de la `commande pip install tetpyclient`.
5. Une fois le SDK Python installé, voici un code standard pour l'instanciation de RestClient :

```
from tetpyclient import RestClient

API_ENDPOINT="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"

# ``verify`` is an optional param to disable SSL server authentication.
# By default, cluster dashboard IP uses self signed cert after
# deployment. Hence, ``verify=False`` might be used to disable server
# authentication in SSL for API clients. If users upload their own
# certificate to cluster (from ``Platform > SSL Certificate``)
# which is signed by their enterprise CA, then server side authentication
# should be enabled; in such scenarios, in the code below, verify=False
# should be replaced with verify="path-to-CA-file"
# credentials.json looks like:
# {
#   "api_key": "<hex string>",
#   "api_secret": "<hex string>"
# }

restclient = RestClient(API_ENDPOINT,
                credentials_file='<path_to_credentials_file>/credentials.json',
                verify=False)
# followed by API calls, for example API to retrieve list of agents.
# API can be passed /openapi/v1/sensors or just /sensors.
resp = restclient.get('/sensors')
```

Générer une clé API et un code secret

Procedure

Étape 1 Dans le coin supérieur droit de l'interface utilisateur Cisco Secure Workload, cliquez sur le compte connecté et sélectionnez **API Keys** (Clés API).

Étape 2 Cliquez sur **Create API Key** (créer une clé API).

Étape 3 (Facultatif) Saisissez une description pour la clé API.

Étape 4 Sélectionnez les fonctionnalités requises pour la clé et le code secret.

Sélectionnez l'ensemble limité de fonctionnalités destinées à l'utilisation de la paire clé API + code secret.

Note La disponibilité des fonctionnalités de l'API varie selon le rôle de l'utilisateur.

Table 1: Fonctionnalités de l'API

Capacité	Description
`sensor_management`	Pour configurer et surveiller l'état des agents logiciels
`software_download`	Pour télécharger des progiciels pour des agents ou des appliances virtuelles
`flow_inventory_query`	Pour interroger les flux et les articles de l'inventaire de la grappe Cisco Secure Workload
`user_role_scope_management`	Pour lire, ajouter, modifier ou supprimer des utilisateurs, des rôles et des portées
`user_data_upload`	Pour permettre aux utilisateurs de téléverser des données pour annoter les flux et les éléments d'inventaire ou de téléverser des condensés (hachages) de fichiers corrects ou non valides
`app_policy_management`	Pour gérer les espaces de travail (applications) et appliquer les politiques
`external_integration`	Pour permettre l'intégration avec des systèmes externes tels que vCenter et Kubernetes

Table 2: Capacités d'API pour les administrateurs de site

Capacité	Description
`appliance_management`	Pour gérer les appareils Cisco Secure Workload
`appliance_monitoring`	Pour surveiller les paramètres et la configuration des appareils Cisco Secure Workload (en lecture seule)

Étape 5 Cliquez sur **Create** (créer).

La clé API et le code secret sont générés, puis doivent être copiés dans un fichier et enregistrés dans un emplacement sûr. Sinon, vous pouvez télécharger le fichier JSON avec la clé et le code secret.

Note Si les options External Auth with LDAP (Authentification externe avec LDAP) et LDAP Authorization (Autorisation LDAP) sont activées, l'accès à OpenAPI à l'aide des clés API s'arrête parce que les rôles Cisco Secure Workload dérivés des groupes LDAP MemberOf (Membre de LDAP) sont réévalués après la fin de la session utilisateur. Par conséquent, pour assurer un accès OpenAPI ininterrompu, il est recommandé à tout utilisateur disposant de clés API d'activer l'option **Use Local Authentication** (Utiliser l'authentification locale) dans le flux Edit User Details (modifier les détails de l'utilisateur) de ce dernier.

Espaces de travail et politiques de sécurité

Les pages suivantes décrivent les points terminaux OpenAPI pour segmentation.

Espaces de travail

Les espaces de travail (anciennement « espaces de travail d'applications » ou « applications ») sont les conteneurs qui permettent de définir, d'analyser et d'appliquer les politiques pour les charges de travail dans une portée spécifique. Pour en savoir plus sur leur fonctionnement, consultez la documentation sur les espaces de travail. Cet ensemble d'API nécessite la capacité `app_policy_management` associée à la clé API.

Objet espace de travail

L'objet JSON d'espace de travail (« application ») est renvoyé sous forme d'objet unique ou de tableau d'objets selon le point terminal d'API. Les attributs de l'objet sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Un identifiant unique pour l'espace de travail.
name	chaîne	Nom spécifié par l'utilisateur de l'espace de travail.
description	chaîne	Description de l'espace de travail précisée par l'utilisateur.
app_scope_id	chaîne	ID de la portée auquel l'espace de travail est associé.
author	chaîne	Prénom et nom de famille de l'utilisateur qui a créé l'espace de travail.
primary	booléen	Indique si l'espace de travail est principal pour sa portée.

Attribut	Type	Description
alternate_query_mode	booléen	Indique si le « mode dynamique » est utilisé pour l'espace de travail. En mode dynamique, une exécution de découverte automatique des politiques crée une ou plusieurs requêtes admissibles pour chaque grappe. La valeur par défaut est « vrai ».
created_at	nombre entier	Horodatage Unix de la création de l'espace de travail.
latest_adm_version	nombre entier	La dernière version adm (v*) de l'espace de travail.
analysis_enabled	booléen	Indique si l'analyse est activée sur l'espace de travail.
analyzed_version	nombre entier	La version p* analysée de l'espace de travail.
enforcement_enabled	booléen	Indique si l'application est activée pour l'espace de travail.
enforced_version	nombre entier	La version p* appliquée de l'espace de travail.

Répertorier les applications

Ce point terminal renverra un tableau d'espaces de travail (« applications »).

```
GET /openapi/v1/applications
```

Table 3: Paramètres

Nom	Type	Description
app_scope_id	chaîne	Fait correspondre des espaces de travail associés à une portée d'application spécifique.
exact_name	chaîne	Fait correspondre les espaces de travail avec exactement la valeur fournie.

Objet de réponse : renvoie un tableau des objets espace de travail.

Exemple de code Python

```
restclient.get('/applications')
```

Récupérer un seul espace de travail

Ce point terminal renverra l'espace de travail demandé (« application ») en tant qu'objet JSON unique.

```
GET /openapi/v1/applications/{application_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l’espace de travail.

Objet de réponse : renvoie l’objet espace de travail pour l’ID spécifié.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s' % application_id)
```

Créer un espace de travail

Ce point terminal crée un espace de travail (« application »). Il est possible de définir des politiques en publiant un corps JSON contenant les définitions de la grappe et des politiques.

Note S’il existe un espace de travail principal pour la même portée et que de nouvelles politiques sont fournies, les politiques seront ajoutées en tant que nouvelle version à l’espace de travail existant.

```
POST /openapi/v1/applications
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
app_scope_id	chaîne	L’ID de la portée à affecter à l’espace de travail.
name	chaîne	(Facultatif) Un nom pour l'espace de travail.
description	chaîne	(Facultatif) La description mise à jour de l’espace de travail.
alternate_query_mode	booléen	(Facultatif) Indique si le « mode dynamique » est utilisé pour l’espace de travail. En mode dynamique, une exécution de découverte automatique des politiques crée une ou plusieurs requêtes admissibles pour chaque grappe. La valeur par défaut est « vrai ».
strict_validation	booléen	(Facultatif) Renvoie une erreur s'il y a des clés ou des attributs inconnus dans les données téléversées. Utile pour détecter les clés mal épelées. La valeur par défaut est faux.

Nom	Type	Description
primary	chaîne	(Facultatif) Définissez sur « vrai » si cet espace de travail doit être principal pour la portée associée. **La valeur par défaut est « vrai »**

Des paramètres facultatifs supplémentaires peuvent être inclus pour décrire les politiques à créer dans l'espace de travail.

Note Le schéma correspond à celui renvoyé lors de l'exportation par l'interface utilisateur et le point terminal **Details**.

Nom	Type	Description
clusters	tableau de grappes	Groupes de nœuds à utiliser pour définir des politiques
inventory_filters	tableau de filtres d'inventaire	Ressources du centre de données
absolute_policies	tableau de politiques	Ordonnancement des politiques à créer avec le rang absolu.
default_policies	tableau de politiques	Ordonnancement des politiques à créer avec le rang par défaut .
catch_all_action	chaîne	« AUTORISER » ou « REFUSER »

Attributs de l'objet de la grappe :

Nom	Type	Description
ID	chaîne	Identifiant unique à utiliser avec les politiques.
name	chaîne	Nom affiché de la grappe.
description	chaîne	Description de la grappe.
nodes	tableau de nœuds	Les nœuds ou les points terminaux qui font partie de la grappe.
consistent_uuid	chaîne	Doit être unique à un espace de travail donné. Après une exécution de découverte automatique des politiques, les grappes similaires/identiques de la prochaine version maintiendront l'attribut consistent_uuid.

Attributs de l'objet nœud :

Nom	Type	Description
ip	chaîne	IP ou sous-réseau du nœud Par exemple 10.0.0.0/8 ou 1.2.3.4
name	chaîne	Nom affiché du nœud.

Attributs de l'objet filtre d'inventaire :

Nom	Type	Description
ID	chaîne	Identifiant unique à utiliser avec les politiques.
name	chaîne	Nom affiché de la grappe.
query	objet	Représentation objet JSON d'une requête de filtre d'inventaire.

Attributs de l'objet politiques :

Nom	Type	Description
consumer_filter_id	chaîne	ID d'une grappe, d'un filtre d'inventaire d'utilisateurs ou de la portée de l'application.
provider_filter_id	chaîne	ID d'une grappe, d'un filtre d'inventaire d'utilisateurs ou de la portée de l'application.
action	chaîne	« AUTORISER » ou « REFUSER »
l4_params	tableau de l4 paramètres	Liste des ports et des protocoles autorisés.

Attributs de l'objet L4Params :

Nom	Type	Description
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
port	tableau	Plage de ports inclusive. Par exemple, [80, 80] ou [5000, 6000].
approved	booléen	(Facultatif) Indique si la politique est approuvée. La valeur par défaut est False.

Objet de réponse : renvoie l'objet espace de travail nouvellement créé.

Exemple de code Python

```
name = 'test'
scope_id = '5ce480cc497d4f1b4b9a9e8d'
filter_id = '5ce480cd497d4f1b4b9a9ea4'
application = {
    'app_scope_id': scope_id,
    'name': name,
    'absolute_policies': [
```

```
            {
        # consumer/provider filter IDs can be ID of a cluster identified during automatic
policy discovery (formerly known as ADM),
        # user inventory filter or app scope.
                'provider_filter_id': filter_id,
                'consumer_filter_id': filter_id,
                'action': 'ALLOW',
        # ALLOW policy for TCP on port 80.
                'l4_params': [
                    {
                        'proto': 6, # TCP
                        'port': [80, 80],  # port range
                    }
                ],
            }
        ],
        'catch_all_action': 'ALLOW'
    }
    restclient.post('/applications', json_body=json.dumps(application))
```

Importer une nouvelle version

Importe les politiques et crée une nouvelle version v* pour l'espace de travail (« application »).

`POST /openapi/v1/applications/{application_id}/import`

Les paramètres sont les mêmes que pour le point terminal de création d'espace de travail.

Objet de réponse : renvoie l'objet de l'espace de travail.

Valider un ensemble de politiques

Valide un ensemble de politiques sans créer de nouvelle version.

`POST /openapi/v1/applications/validate_policies`

Un *app_scope_id* est obligatoire. Les autres paramètres sont identiques à ceux du point terminal de création d'espace de travail.

Objet de réponse :

Attribut	Type	Description
valide	booléen	Indique si les politiques sont valides
erreurs	tableau	Si non valide, détails sur les erreurs

Supprimer un espace de travail

Supprime un espace de travail (une« application »).

`DELETE /openapi/v1/applications/{application_id}`

La mise en application doit être désactivée sur l'espace de travail avant de pouvoir le supprimer.

Si l'espace de travail, ou ses grappes, est utilisé par d'autres applications (par une relation de service fourni), ce point terminal renverra le message `422 Unprocessable Entity` (422 Entité non traitable). L'objet Erreur renvoyé contiendra un attribut `détails` avec le nombre d'objets dépendants ainsi que les ID des 10 premiers de chaque type. Ces renseignements peuvent être utilisés pour localiser et supprimer les dépendances bloquantes.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Objet de réponse : aucun

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
restclient.delete('/applications/%s' % application_id)
```

Mettre à jour un espace de travail

Ce point terminal met à jour un espace de travail existant (« application »).

```
PUT /openapi/v1/applications/{application_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
name	chaîne	(Facultatif) Le nom mis à jour de l'espace de travail.
descrip	chaîne	(Facultatif) La description mise à jour de l'espace de travail.
primary	chaîne	(Facultatif) Définissez à la valeur « vrai » pour en faire l'espace de travail principal. Définissez la valeur « faux » pour rendre l'espace de travail secondaire.

Objet de réponse : l'objet d'espace de travail mis à jour pour l'ID spécifié.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'name': 'Updated Name',
      'description': 'Updated Description',
      'primary': 'true'
}
resp = restclient.put('/applications/%s' % application_id,
                      json_body=json.dumps(req_payload))
```

Récupérer les détails de l'espace de travail

Ce point terminal renvoie un fichier JSON d'exportation complet pour l'espace de travail. Il comprend les définitions de politique et de grappe.

```
GET /openapi/v1/applications/{application_id}/details
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
version	chaîne	(Facultatif) Une version sous la forme « v10 » ou « p10 », par défaut, « latest » (dernière).

Objet de réponse : renvoie les grappes et les politiques dans la version d'espace de travail donnée.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
# For v* version v10 and for p* version p10
version = 'v10'
resp = restclient.get('/applications/%s/details?version=%s' % (application_id, version))
```

Répertorier les versions d'espace de travail

Ce point terminal renverra une liste de toutes les versions pour un espace de travail donné.

```
GET /openapi/v1/applications/{application_id}/versions
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
created_before	nombre entier	(Facultatif) Pour la pagination, définissez la valeur « created_at » de la dernière version de la réponse précédente.
limit	nombre entier	(Facultatif) Nombre maximal de résultats à renvoyer, la valeur par défaut est 50.

Objet de réponse : un tableau d'objets ayant les attributs suivants :

Attribut	Type	Description
version	chaîne	Une version sous la forme « v10 » ou « p10 ».

Attribut	Type	Description
created_at	nombre entier	Horodatage Unix de la création de l'espace de travail.
description	chaîne	Description fournie par l'utilisateur
name	chaîne	Nom d'affichage

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
created_before = 1612325705
limit = 10
resp = restclient.get('/applications/%s/versions?created_before=%s&limit=%s' %
                      (application_id, created_before, limit))
```

Supprimer la version de l'espace de travail

Ce point terminal supprimera la version donnée, y compris les grappes et les politiques. Les versions appliquées ou analysées ne peuvent pas être supprimées. Si des membres sont référencés par un autre espace de travail, par le biais d'une politique externe, la réponse renvoie une erreur avec une liste des références.

```
DELETE /openapi/v1/applications/{application_id}/versions/{version}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
version	chaîne	Une version sous la forme « v10 » ou « p10 ».

Objet de réponse : aucun

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
version = 'v10'
resp = restclient.delete('/applications/%s/versions/%s' %
                         (application_id, version))
```

Comparer les versions de l'espace de travail

Ce point terminal calcule la différence entre les versions de l'espace de travail fournies. Il retourne les politiques ajoutées, supprimées et éventuellement inchangées. Les modifications apportées à la grappe sont incluses si la grappe est présente dans les deux versions, définie par un _uuid cohérent correspondant, et que la requête a été modifiée.

```
GET /openapi/v1/applications/{application_id}/version_diff
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
base_version	chaîne	Version complète, par exemple « v10 » ou « p10 ».
draft_version	chaîne	Version complète, par exemple « v10 » ou « p10 ».
include_unchanged	booléen	La valeur par défaut est False. Renvoie les politiques inchangées dans la réponse.

Objet de réponse : renvoie un objet avec les attributs suivants :

Attribut	Type	Description
clusters	tableau	Les grappes qui ont été modifiées entre les versions.
politiques	tableau	Les politiques qui ont été modifiées entre les versions.

Analyser les dernières politiques

Activez l'analyse du dernier ensemble de règles dans l'espace de travail.

```
POST /openapi/v1/applications/{application_id}/enable_analysis
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Paramètres : le corps facultatif de la requête JSON contient les clés suivantes :

Nom	Type	Description
action_note	chaîne	(Facultatif) Raison de l'action de publication des politiques.
name	chaîne	(Facultatif) Nom dans la version de politique publiée.
description	chaîne	(facultatif) une description dans la version de politique publiée.

Objet de réponse : renvoie un objet avec les attributs suivants :

Attribut	Type	Description
data_set	objet	Représentation sous forme d'objet JSON de l'ensemble de données.
analyzed_policy_version	nombre entier	La version p* analysée de l'espace de travail.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'action_note': 'Policy analysis',
  'name': 'Test run 1',
  'description': 'New workloads added.'
}
resp = restclient.post('/applications/%s/enable_analysis' % application_id,
                       json_body=json.dumps(req_payload))
```

Désactiver l'analyse des politiques sur un seul espace de travail

Désactiver l'analyse des politiques sur l'espace de travail.

```
POST /openapi/v1/applications/{application_id}/disable_analysis
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Objet de réponse : renvoie un objet avec les attributs suivants :

Attribut	Type	Description
data_set	objet	Représentation sous forme d'objet JSON de l'ensemble de données.
analyzed_policy_version	nombre entier	Dernière version p* analysée de l'espace de travail.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_analysis' % application_id)
```

Appliquer un espace de travail unique

Activez l'application du dernier ensemble de politiques dans l'espace de travail.

```
POST /openapi/v1/applications/{application_id}/enable_enforce
```

Warning De nouvelles règles de pare-feu d'hôte seront insérées et toutes les règles existantes seront supprimées sur les hôtes concernés.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
version	chaîne	(Facultatif) La version de la politique à appliquer.

Si aucune version n'est fournie, les dernières politiques de l'espace de travail seront appliquées. « versions » doit être de manière préférentielle sous la forme « p* »; si seul un entier est fourni, la version « p* » correspondante sera appliquée.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
heure d'origine	chaîne	Identifiant unique du dernier profil d'application de la loi.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'version': 'p10'
}
resp = restclient.post('/applications/%s/enable_enforce' % application_id,
                       json_body=json.dumps(req_payload))
```

Désactiver l'application pour un seul espace de travail

Désactivez l'application sur l'espace de travail.

```
POST /openapi/v1/applications/{application_id}/disable_enforce
```

Warning De nouvelles règles de pare-feu d'hôte seront insérées et toutes les règles existantes seront supprimées sur les hôtes concernés.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
heure d'origine	chaîne	Identifiant unique du dernier profil d'application de la loi.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_enforce' %
                       application_id)
```

Initier la découverte automatique des politiques

Détectez automatiquement les politiques pour l'espace de travail. (anciennement dénommé « soumission d'une exécution ADM »).

```
POST /openapi/v1/applications/{application_id}/submit_run
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
start_time	chaîne	Heure de début de l'intervalle d'entrée pour une exécution de découverte automatique de politiques.
end_time	chaîne	Heure de fin de l'intervalle d'entrée pour une exécution de découverte automatique de politique.
clustering_granularity	chaîne	(Facultatif) La granularité de grappe permet à l'utilisateur de contrôler la taille des grappes générées par la découverte automatique des politiques. Valeurs attendues : VERY_FINE, FINE, MEDUM, COARSE ou VERY_COARSE

Nom	Type	Description
port_generalization	chaîne	(Facultatif) La généralisation de port contrôle le niveau de signification statistique requis lors de la généralisation de port. Valeurs attendues : DISABLED, CONSERVATIVE, MODERATE, AGRESSIVE ou VERY_AGGRESSIVE
policy_compression	chaîne	(Facultatif) la Compression de politiques lorsqu'elle est activée, les politiques qui sont suffisamment fréquentes, c'est-à-dire qui utilisent le même port fournisseur, parmi les grappes générées à l'intérieur d'un espace de travail peuvent être "factorisées" vers le parent, c'est-à-dire remplacées par une ou plusieurs politiques applicables à l'ensemble de la portée parente. Valeurs attendues : DISABLED, CONSERVATIVE, MODERATE, AGRESSIVE ou VERY_AGGRESSIVE
auto_accept_policy_connectors	booléen	(facultatif) Acceptation automatique des connecteurs de politique Toutes les demandes de politique sortantes créées lors de la découverte automatique des politiques sont automatiquement acceptées.
enable_exclusion_filter	booléen	(Facultatif) L'option Activer le filtre d'exclusion offre la possibilité d'ignorer toutes les conversations correspondant à l'un des filtres d'exclusion définis par l'utilisateur (le cas échéant). Pour en savoir plus, consultez Filtres d'exclusion.
enable_default_exclusion_filter	booléen	(Facultatif) L'option Activer le filtre d'exclusion par défaut offre la possibilité d'ignorer toutes les conversations correspondant à l'un des filtres d'exclusion par défaut (le cas échéant). Consultez la section Filtres d'exclusion par défaut pour en savoir plus.

Nom	Type	Description
enable_service_discovery	booléen	(Facultatif) Lorsque l'option Enable service discovery on agent (Activer la découverte de services sur l'agent) est définie, des informations éphémères de plage de ports sont fournies sur les services présents sur le nœud de l'agent. Des politiques sont ensuite générées en fonction des informations de plage de ports signalées.
carry_over_policies	booléen	(facultatif) Lorsque Carry over Approved Policies (report des politiques approuvées) est défini, toutes les politiques marquées comme approuvées par l'utilisateur dans l'interface utilisateur ou OpenAPI seront conservées.
skip_clustering	booléen	(Facultatif) Lorsque l'option Skip clustering(Ignorer la mise en grappe) est définie, aucune nouvelle grappe n'est générée, et les politiques sont générées à partir de toutes les grappes approuvées existantes ou des filtres d'inventaire et impliquent toutes les charges de travail de la portée.
deep_policy_generation	booléen	(Facultatif) Vous pouvez générer des politiques pour une branche de l'arborescence de la portée plutôt que pour une seule portée. Pour en savoir plus, consultez Découvrir les politiques relatives à une portée ou à une branche de l'arborescence de la portée et les sous-sections.
use_default_config	booléen	(Facultatif) Lorsque cette option est définie, la découverte automatique des politiques utilise la configuration de la découverte des politiques par défaut au lieu de la configuration d'exécution précédente. Pour en savoir plus, consultez Configuration de découverte des politiques par défaut.

Note Les valeurs par défaut des paramètres facultatifs non spécifiés seront issues de la configuration précédente de la découverte automatique des politiques si cette dernière a été effectuée plus tôt dans l'espace de travail, sinon les valeurs par défaut seront tirées de la configuration de découverte des politiques par défaut.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
message	chaîne	Message concernant la réussite ou l'échec de l'exécution de la découverte automatique des politiques.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'start_time': '2020-09-17T10:00:00-0700',
      'end_time': '2020-09-17T11:00:00-0700',
      # Optional Parameters.
      'clustering_granularity': 'FINE',
      'port_generalization': 'AGGRESSIVE',
      'policy_compression': 'AGGRESSIVE',
      'auto_accept_policy_connectors': False,
      'enable_exclusion_filter': True,
      'enable_default_exclusion_filter': True,
      'enable_service_discovery': True,
      'carry_over_policies': True,
      'skip_clustering': False,
      'deep_policy_generation': True,
      'use_default_config': False
}
resp = restclient.post('/applications/%s/submit_run' % application_id,
                       json_body=json.dumps(req_payload))
```

Obtenir l'état d'une exécution de découverte de politiques

Recherchez l'état d'une exécution de découverte automatique de politique dans l'espace de travail.

```
GET /openapi/v1/applications/{application_id}/adm_run_status
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
status	chaîne	État de l'exécution de la découverte automatique des politiques. Valeurs : PENDING (EN COURS), COMPLETE (TERMINÉ) ou FAILED (ERREUR)

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/applications/%s/adm_run_status' % application_id)
```

Politiques

Cet ensemble d'API peut être utilisé pour gérer l'ajout, la modification ou la suppression de politiques. Le paramètre de `version` est requis pour les actions Collecte de toutes les créations et mises à jour. Ces renseignements nécessitent la capacité `user_role_scope_management` associée à la clé API.

Objet politique

Les attributs de l'objet de politique sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique de la politique.
application_id	chaîne	ID de l'espace de travail auquel la politique appartient.
consumer_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe, tout filtre défini par l'utilisateur ou toute portée peut être utilisé comme consommateur d'une politique.
provider_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe, tout filtre défini par l'utilisateur ou toute portée peut être utilisé comme fournisseur d'une politique.
version	chaîne	Indique la version de l'espace de travail auquel la politique appartient.
rank	chaîne	Rang de politique, valeurs possibles : DEFAULT (PAR DÉFAUT), ABSOLUTE (ABSOLUE) ou CATCHALL COLLECTRICE).

Attribut	Type	Description
policy_action	chaîne	Les valeurs possibles peuvent être ALLOW (AUTORISER) ou DENY (REFUSER). Indique si le trafic doit être autorisé ou abandonné pour un port de service ou un protocole entre le consommateur et le fournisseur de service.
priority	nombre entier	Utilisé pour trier les politiques.
l4_params	tableau de l4 paramètres	Liste des ports et des protocoles autorisés.

Attributs de l'objet L4Params :

Nom	Type	Description
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
port	tableau	Gamme de ports inclusive. ex., [80, 80] ou [5000, 6000].
description	chaîne	Chaîne courte concernant ce protocole et ce port.
approved	booléen	Si la politique a été approuvée par l'utilisateur.

Obtenir des politiques

Ce point terminal renvoie une liste des politiques dans un espace de travail particulier. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/policies
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
version	chaîne	Indique la version de l'espace de travail à partir duquel obtenir les politiques.
consumer_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID du filtre du consommateur.
provider_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID du filtre du consommateur.

Les ID de politiques peuvent changer d'une version à l'autre. Pour obtenir la liste des politiques d'une version publiée, le numéro de version doit être précédé d'un « p ». Par exemple, pour récupérer toutes les politiques dans la version publiée 3, nous pouvons effectuer une requête comme :

```
GET /openapi/v1/applications/{application_id}/policies?version=p3
```

Renvoie un objet de toutes les politiques de cet espace de travail particulier, comme indiqué ci-dessous

```
{
  absolute_policies: [ ... ],
  default_policies: [ ... ],
  catch_all_action:
}
```

Exemple de code Python

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/policies' % application_id, params={'version': '1'})
```

Obtenir les politiques par défaut

Ce point terminal renvoie une liste des politiques par défaut pour un espace de travail donné. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/default_policies
```

Paramètres :

Nom	Type	Description
ID	chaîne	Identifiant unique de la politique.
version	chaîne	Indique la version de l'espace de travail pour lequel obtenir les politiques.
limit	nombre entier	Limite le nombre de politiques par demande.
offset	nombre entier	(Facultatif) Nombre de décalage reçu de la réponse précédente, doit toujours être utilisé avec `limite`.
consumer_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID du filtre du consommateur.
provider_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID de filtre du fournisseur.

Renvoie la liste des politiques par défaut dans la version fournie de cet espace de travail. La réponse contient le nombre de politiques demandé et un `décalage`, pour obtenir le prochain ensemble de politiques, utilisez ce `décalage` dans les demandes suivantes. L'absence de `décalage` dans la réponse indique que toutes les politiques sont déjà extraites.

Exemple de code Python

```
  application_id = '5f88c996755f023f3bafe163'
  restclient.get('/applications/%s/default_policies' % application_id, params={'version':
'1', 'limit': 3, 'offset': 3})
```

Exemple de réponse

```
    {
      "results": [
      PolicyObject4,
      PolicyObject5,
      PolicyObject6
      ],
      "offset": 6
    }
```

Obtenir les politiques absolues

Ce point terminal renvoie une liste des politiques absolues dans un espace de travail donné. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/absolute_policies
```

Paramètres :

Nom	Type	Description
version	chaîne	Indique la version de l'espace de travail à partir duquel obtenir les politiques.
limit	nombre entier	Limite le nombre de politiques par demande.
offset	nombre entier	(Facultatif) Nombre de décalage reçu de la réponse précédente, doit toujours être utilisé avec `limite`.
consumer_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID du filtre du consommateur.
provider_filter_id	chaîne	(Facultatif) Filtre la sortie en fonction de l'ID de filtre du fournisseur.

Renvoie la liste des politiques absolues dans la version fournie de cet espace de travail. La réponse contient le nombre de politiques demandé et un `décalage`, pour obtenir le prochain ensemble de politiques, utilisez ce `décalage` dans les demandes suivantes. L'absence de `décalage` dans la réponse indique que toutes les politiques sont déjà extraites.

Exemple de code Python

```
  application_id = '5f88c996755f023f3bafe163'
  restclient.get('/applications/%s/absolute_policies' % application_id, params={'version':
 '1', 'limit': 3})
```

Exemple de réponse

```
{
  "results": [
  PolicyObject1,
  PolicyObject2,
  PolicyObject3
  ],
  "offset": 3
}
```

Obtenir les politiques Catch All (collectrices)

Ce point terminal renvoie une politique Catch All (collectrice) pour un espace de travail donné. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/catch_all
```

Paramètres :

Nom	Type	Description
version	chaîne	Indique la version de l'espace de travail à partir duquel obtenir les politiques.

Renvoie un objet de politique unique représentant la politique collectrice d'une version donnée de l'espace de travail.

Exemple de code Python

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/catch_all' % application_id, params={'version': '1'})
```

Obtenir une politique spécifique

Ce point terminal renvoie une instance d'une politique.

```
GET /openapi/v1/policies/{policy_id}
```

Renvoie l'objet de politique associé à l'ID spécifié.

Exemple de code Python

```
policy_id = '5f88ca1e755f0222f85ce85c'
restclient.get('/policies/%s' % policy_id)
```

Rechercher une politique spécifique avec un identifiant de politique

Ce point terminal recherche la politique spécifiée en utilisant les paramètres d'identifiant de politique comme clé composée.

```
POST /openapi/v1/policies/search
```

Le corps de la requête se compose d'un corps JSON comportant le schéma suivant :

Nom	Type	Description
application_id	chaîne	ID de l'espace de travail de l'application.

Nom	Type	Description
policy_identifier	objet	Champs qui constituent l'identifiant cohérent de la politique.

Les champs d'identifiant de politique sont constitués selon le schéma suivant :

Nom	Type	Description
version	chaîne	(Facultatif) Indique la version de l'application pour laquelle obtenir les politiques. utilise par défaut la dernière version « v » de l'application lorsque cela n'est pas spécifié.
consumer_consistent_uuid	chaîne	UUID cohérent du consommateur ou de la source.
provider_consistent_uuid	chaîne	UUID cohérent du fournisseur ou de la destination.
rank	chaîne	Le rang de la politique doit être « DEFAULT (PAR DÉFAUT) » ou « ABSOLUTE (ABSOLUE) ».
action	chaîne	L'action de politique doit être « ALLOW (AUTORISER) » ou « DENY (REFUSER) ».
priority	nombre entier	Valeur de la priorité pour la politique.
protocol	nombre entier	Numéro de protocole IP (0 à 255).
start_port	nombre entier	(Facultatif) Début de la plage de ports (0 à 65 535) la valeur par défaut est 0 lorsqu'elle n'est pas précisée.
end_port	nombre entier	(facultatif) fin de la plage de ports (0 à 65 535); par défaut à 65535 si start_port est égal à 0 ou sinon à start_prot

Exemple de code Python

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
rank = 'DEFAULT'
action = 'ALLOW'
priority = 100
protocol = 6
start_port = 80
version = 'p3'

req_body = f'''
{{
    "application_id": "{application_id}",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{consumer_id}",
```

```
            "provider_consistent_uuid": "{provider_id}",
            "rank": "{rank}",
            "action": "{action}",
            "priority": {priority},
            "protocol": "{protocol}",
            "start_port": "{start_port}",
            "version": "{version}"
        }}
    }}'''
    restclient.post('/policies/search', json_body=req_body)
```

Créer une politique

Ce point terminal est utilisé pour créer de nouvelles politiques.

```
POST /openapi/v1/applications/{application_id}/policies
```

Paramètres :

Attribut	Type	Description
consumer_filter_id	chaîne	ID d'un filtre défini.
provider_filter_id	chaîne	ID d'un filtre défini.
version	chaîne	Indique la version de l'espace de travail dans lequel les politiques doivent être mises à jour.
rank	chaîne	les valeurs peuvent être DEFAULT, ABSOLUTE ou CATCHALL pour le classement
policy_action	chaîne	les valeurs peuvent être ALLOW ou DENY : signifie si nous devons autoriser ou abandonner le trafic du consommateur au fournisseur sur un port de service/protocole donné
priority	nombre entier	Utilisé pour trier les politiques.

Exemple de code Python

```
  req_payload = {
    "version": "v1",
    "rank" : "DEFAULT",
    "policy_action" : "ALLOW",
    "priority" : 100,
    "consumer_filter_id" : "123456789",
    "provider_filter_id" : "987654321",
  }
  resp = restclient.post('/openapi/v1/applications/{application_id}/policies',
json_body=json.dumps(req_payload))
```

Créer une politique par défaut

Ce point terminal est utilisé pour créer de nouvelles politiques par défaut. Ce point terminal crée une politique par défaut semblable à la création d'un point terminal de politique.

```
POST /openapi/v1/applications/{application_id}/default_policies
```

Créer une politique absolue

Ce point terminal est utilisé pour créer de nouvelles politiques absolues. Ce point terminal crée une politique absolue similaire à la création d'un point terminal de politique.

```
POST /openapi/v1/applications/{application_id}/absolute_policies
```

Mettre à jour une politique

Ce point terminal met à jour une politique.

```
PUT /openapi/v1/policies/{policy_id}
```

Paramètres :

Attribut	Type	Description
consumer_filter_id	chaîne	ID d'un filtre défini.
provider_filter_id	chaîne	ID d'un filtre défini.
policy_action	chaîne	Les valeurs possibles peuvent être ALLOW (AUTORISER) ou DENY (REFUSER). Indique si le trafic doit être autorisé ou abandonné pour un port de service ou un protocole entre le consommateur et le fournisseur de service.
priority	nombre entier	Utilisé pour trier les priorités des politiques

Renvoie l'objet de politique modifié associé à l'ID spécifié.

Mettre à jour une politique collectrice

Ce point terminal met à jour la politique Catch All (collectrice) pour un espace de travail particulier.

```
PUT /openapi/v1/applications/{application_id}/catch_all
```

Paramètres :

Attribut	Type	Description
version	chaîne	Indique la version de l'espace de travail dans lequel les politiques doivent être mises à jour.
policy_action	chaîne	Les valeurs possibles peuvent être ALLOW (AUTORISER) ou DENY (REFUSER). Indique si le trafic ne correspondant à aucune des politiques de cet espace de travail sera autorisé ou abandonné.

Ajout de ports de service à une politique

Ce point terminal est utilisé pour créer des ports de service pour une politique spécifique.

```
POST /openapi/v1/policies/{policy_id}/l4_params
```

Paramètres :

Attribut	Type	Description
version	chaîne	Indique la version de l'espace de travail à partir duquel obtenir les politiques.
start_port	nombre entier	Port de début de la plage.
end_port	nombre entier	Port de fin de la plage
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
description	chaîne	(Facultatif) Chaîne courte concernant ce protocole et ce port.

Mise à jour des ports de service d'une politique

Ce point terminal met à jour le port de service spécifié d'une politique.

```
PUT /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Paramètres :

Attribut	Type	Description
approved	booléen	Marque la politique comme approuvée.

Suppression des ports de service d'une politique

Ce point terminal supprime le port de service spécifié d'une politique. (Facultatif) Consultez la section Filtres d'exclusion pour en savoir plus.

```
DELETE /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Paramètres :

Attribut	Type	Description
create_exclusion_filter	booléen	(Facultatif) Si la valeur est Vrai (True), crée un filtre d'exclusion correspondant à la politique. Les flux correspondant à ce filtre seront exclus des futurs cycles de découverte automatique des politiques. Consultez la section Filtres d'exclusion pour en savoir plus.

Suppression d'une politique

Ce point terminal supprime la politique spécifiée. Aucun filtre d'exclusion n'est créé.

```
DELETE /openapi/v1/policies/{policy_id}
```

Suppression d'une politique avec identifiant

Ce point terminal supprime la politique spécifiée à l'aide des paramètres d'identifiant de politique. Aucun filtre d'exclusion n'est créé.

```
DELETE /openapi/v1/policies/destroy_with_identifier
```

Le corps de la requête se compose d'un corps JSON comportant le schéma suivant :

Nom	Type	Description
application_id	chaîne	ID de l'espace de travail de l'application.
policy_identifier	objet	Champs qui constituent l'identifiant cohérent de la politique.

Les champs d'identifiant de politique sont constitués selon le schéma suivant :

Nom	Type	Description
version	chaîne	(facultatif) la version « v » de l'espace de travail de l'application dans laquelle effectuer l'opération de suppression; utilise par défaut la dernière version « v » de l'espace de travail lorsqu'elle n'est pas précisée.
consumer_consistent_uuid	chaîne	UUID cohérent du consommateur ou de la source
provider_consistent_uuid	chaîne	UUID cohérent du fournisseur ou de la destination

Nom	Type	Description
rank	chaîne	Le rang de la politique doit être « DEFAULT (PAR DÉFAUT) » ou « ABSOLUTE (ABSOLUE) ».
action	chaîne	L'action de politique doit être « ALLOW (AUTORISER) » ou « DENY (REFUSER) ».
priority	nombre entier	Valeur de la priorité pour la politique
protocol	nombre entier	Numéro de protocole IP (0 à 255) de la politique
start_port	nombre entier	(Facultatif) Début de la plage de ports (0 à 65 535) la valeur par défaut est 0 lorsqu'elle n'est pas précisée
end_port	nombre entier	(facultatif) fin de la plage de ports (0 à 65 535); par défaut à 65535 si start_port est égal à 0 ou sinon à start_prot

Exemple de code Python

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100
version = '5'

req_body = f'''
{{
    "application_id": "{application_id}",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{consumer_id}",
        "provider_consistent_uuid": "{provider_id}",
        "rank": "{rank}",
        "priority": {priority},
        "action": "{action}",
        "protocol": "{protocol}",
        "start_port": "{start_port}",
        "version": "{version}"
    }}
}}'''
restclient.delete('/policies/destroy_with_identifier', json_body=req_body)
```

Analyse rapide de la politique

Ce point terminal peut être utilisé pour trouver l'ensemble de politiques correspondant à tout flux hypothétique par rapport aux politiques analysées ou appliquées dans une portée racine. Pour plus de détails, consultez l'analyse rapide

Cette API est uniquement disponible pour les utilisateurs disposant d'un accès en lecture minimal à la portée racine et nécessite la capacité `app_policy_management` associée à la clé API.

```
POST /openapi/v1/policies/{rootScopeID}/quick_analysis
```

Le corps de la requête se compose d'un corps JSON comportant le schéma suivant :

Nom	Type	Description
consumer_ip	chaîne	Adresse IP du client/consommateur.
provider_ip	chaîne	Adresse IP du serveur/fournisseur.
provider_port	nombre entier	(Facultatif) Port du fournisseur, pertinent uniquement pour les flux TCP ou UDP.
protocol	chaîne	Protocole du flux, par exemple TCP.
analysis_type	chaîne	Le type d'analyse peut être **analyzed (analysé)** ou **enforced (appliqué).** Le type d'analyse « analyzed (analysé) » prend la décision concernant le flux en faisant correspondre ce dernier à toutes les politiques analysées dans la portée racine. Le type d'analyse « enforced (appliqué) » prend la décision du flux en le faisant correspondre à toutes les politiques appliquées dans la portée racine.
application_id	chaîne	(Facultatif) L'ID de l'espace de travail principal, toujours accompagné dans la version « v » de l'espace de travail, le cas échéant, prend la décision de flux en utilisant les politiques dans la version spécifiée ainsi que les politiques analysées ou appliquées des autres espaces de travail dans la portée racine. Si ce champ est ignoré, la décision du flux est prise en tenant compte de toutes les politiques analysées ou appliquées dans la portée racine.

Nom	Type	Description
version	nombre entier	(Facultatif) La version « v » de l'espace de travail mentionnée ci-dessus. Elle doit être spécifiée si application_id est spécifié et doit être ignoré dans le cas inverse.

Exemple de requête

Le corps de la demande doit être une requête au format JSON.

Exemple de corps de requête où la décision de flux est basée sur toutes les politiques analysées :

```
    req_payload = {
        "consumer_ip": "4.4.1.1",
        "provider_ip": "4.4.2.1",
        "provider_port": 9081,
        "protocol": "TCP",
        "analysis_type": "analyzed"
    }
    resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

Exemple de corps de requête où la décision de flux est basée sur les politiques dans la version « v » de l'espace de travail ainsi que sur les politiques analysées de tous les autres espaces de travail de la portée racine :

```
    req_payload = {
        "consumer_ip": "4.4.1.1",
        "provider_ip": "4.4.2.1",
        "provider_port": 9081,
        "protocol": "TCP",
        "analysis_type": "analyzed",
        "application_id": "5e7e5f56497d4f0bc26c7bb3",
        "version": 1
    }
    resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

Exemple de réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes

Clés	Valeurs
policy_decision	La décision du flux hypothétique d'autoriser ou de refuser.
outbound_policy	La politique du consommateur qui autorise ou refuse le trafic sortant
inbound_policy	La politique du fournisseur qui autorise ou refuse le trafic entrant

```
    {
        "policy_decision": "ALLOW",
        "outbound_policy": {
            "policy_rank": "DEFAULT",
```

```
            "start_port": 9082,
            "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
            "src_filter_id": "5e7e600e497d4f7341f4f459",
            "end_port": 9082,
            "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
            "dst_filter_id": "5e7d0efc497d4f44b6b09351",
            "action": "ALLOW",
            "protocol": "TCP",
            "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
        },
        "inbound_policy": {
            "policy_rank": "DEFAULT",
            "start_port": 9082,
            "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
            "src_filter_id": "5e7e600e497d4f7341f4f459",
            "end_port": 9082,
            "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
            "dst_filter_id": "5e7d0efc497d4f44b6b09351",
            "action": "ALLOW",
            "protocol": "TCP",
            "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
        }
    }
```


Statistiques de la politique

Ce point terminal renvoie le nombre de paquets, d'octets et de conversations observés pour une politique sur un intervalle de temps. Une conversation peut être décrite dans les grandes lignes comme une observation de flux correspondant à une politique qui est agrégée avec une granularité d'une heure. Le nombre de conversations mesurées pour une politique donnée sur une heure représente le nombre de paires distinctes d'éléments d'inventaire client et fournisseur qui ont communiqué sur le réseau au cours de cette heure.

Bien que ce point terminal accepte les paramètres d'identifiant de politique comme entrées, nous vous recommandons d'utiliser les ID de politique et de paramètres L4 d'une version publiée de l'espace de travail.

Remarque Après la publication d'une nouvelle version de l'espace de travail d'application, six heures peuvent s'écouler avant que les résultats ne soient disponibles. Toutes les résolutions d'horodatage auront également une granularité minimale de 6 heures.

Pour obtenir les statistiques d'une politique sur les versions appliquées d'un espace de travail d'application, le chemin URL est le suivant :

```
POST /openapi/v1/policies/stats/enforced
```

Pour obtenir les statistiques d'une politique sur les versions analysées d'un espace de travail d'application, le chemin URL est le suivant :

```
POST /openapi/v1/policies/stats/analyzed
```

Le corps de la requête se compose d'un corps JSON comportant le schéma suivant :

Tableau 4 :

Nom	Type	Description
application_id	chaîne	ID de l'espace de travail de l'application.

Nom	Type	Description
t0	chaîne	Le début de l'intervalle de temps, au format RFC 3339.
t1	chaîne	(facultatif) La fin de l'intervalle de temps au format RFC-3339; correspond par défaut à l'heure actuelle, si elle n'est pas précisée.
policy_id	chaîne	l'ID de la politique; non obligatoire si l'identifiant de politique est présent.
l4_param_id	chaîne	ID du paramètre l4; non obligatoire si l'identifiant de politique est présent, ou pour les politiques « CATCH_ALL ».
policy_identifier	objet	Champs qui constituent l'identifiant cohérent de la politique.

Les champs d'identifiant de politique sont constitués selon le schéma suivant :

Nom	Type	Description
consumer_consistent_uuid	chaîne	UUID cohérent du consommateur ou de la source.
provider_consistent_uuid	chaîne	UUID cohérent du fournisseur ou de la destination.
rank	chaîne	Le rang de la politique doit être « DEFAULT (PAR DÉFAUT) » ou « ABSOLUTE (ABSOLUE) ».
action	chaîne	L'action de politique doit être « ALLOW (AUTORISER) » ou « DENY (REFUSER) ».
priority	nombre entier	Valeur de la priorité pour la politique.
protocol	nombre entier	Numéro de protocole IP (0 à 255) de la politique.
start_port	nombre entier	(Facultatif) Début de la plage de ports (0 à 65 535) la valeur par défaut est 0 lorsqu'elle n'est pas précisée

Nom	Type	Description
end_port	nombre entier	(facultatif) fin de la plage de ports (0 à 65 535); par défaut à 65535 si start_port est égal à 0 ou sinon à start_prot

Exemple de code Python

```
application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100

req_body = f'''
{{
    "application_id": "{application_id}",
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{consumer_id}",
        "provider_consistent_uuid": "{provider_id}",
        "rank": "{rank}",
        "priority": {priority},
        "action": "{action}",
        "protocol": "{protocol}",
        "start_port": "{start_port}"
    }}
}}'''
restclient.post('/policies/stats/analyzed', json_body=req_body)

# For CATCH_ALL policies:
root_app_scope_id = '6f88ca1e755f0222f85ce85e'
rank = 'CATCH_ALL'
action = 'DENY'
req_body = f'''
{{
    "application_id": "{application_id}",
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z",
    "policy_identifier": {{
        "consumer_consistent_uuid": "{root_app_scope_id}",
        "provider_consistent_uuid": "{root_app_scope_id}",
        "rank": "{rank}",
        "action": "{action}"
    }}
}}'''

restclient.post('/policies/stats/analyzed', json_body=req_body)
```

Exemple de réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Tableau 5 :

Clés	Valeurs
conversation_count	Le nombre de conversations observées pour la durée et la politique spécifiées.
packet_count	Le nombre de paquets observés pour la durée et la politique spécifiées.
byte_count	Le nombre d'octets observés pour la durée et la politique spécifiées.
first_seen_at	Horodatage (au format RFC-3339) de la première observation de flux pour cette politique.
last_seen_at	L'horodatage (au format RFC-3339) de la dernière observation des flux pour cette politique.
agg_start_version	La première version publiée de cette politique enregistrée à partir de l'instant t0.
agg_start_time	Horodatage de publication de agg_start_version.

```
{
  "conversation_count": 72,
  "packet_count": 800,
  "byte_count": 1960,
  "first_seen_at": "2022-09-09T11:00:00.000Z",
  "last_seen_at": "2022-09-09T11:00:00.000Z",
  "agg_start_version": 4,
  "agg_start_time": "2022-08-10T23:00:00.000Z"
}
```

Politiques inutilisées

Ce point terminal renvoie les identifiants de politique dans un espace de travail publié pour lequel aucune conversation n'est observée sur un intervalle de temps spécifié.

Identifiant de la politique

Toutes les politiques et les grappes générées par ADM peuvent modifier leur ID dans les versions de l'espace de travail d'application même si les requêtes de filtre sous-jacentes ou le port et le protocole des politiques ne changent pas. Afin d'assurer le suivi du nombre de résultats pour une politique particulière dans les versions de l'espace de travail, nous utilisons des UUID cohérents pour les filtres qui ne changent pas d'une version à l'autre. Une clé composée appelée *identifiant de politique* comprend des UUID cohérents pour le fournisseur et le consommateur, ainsi que le rang, l'action, la priorité, le port et le protocole.

Ainsi, les identifiants de politiques servent de clé composée qui peut à la fois identifier et décrire les aspects importants d'une politique pour toutes les versions de l'espace de travail d'application, tandis que les ID de politique (comme ceux utilisés dans les points terminaux CRUD habituels) peuvent changer d'une version à l'autre.

Remarque Les UUID et identifiants de politiques cohérents avec le fournisseur ou le consommateur ne permettent pas d'identifier de façon unique un filtre ou une politique, car ils sont partagés entre différentes versions d'espace de travail d'application.

Pour effectuer des opérations CRUD sur une grappe ou une politique particulière, il est recommandé de résoudre l'identifiant à une politique concrète pour une version spécifique de l'espace de travail de l'application, le point d'arrivée de la recherche.

Les opérations CRUD classiques peuvent être effectuées à l'aide des ID de politique, tandis que seuls les statistiques de politique et l'API Détruire avec identifiant acceptent l'identifiant de politique comme entrée. Il s'agit principalement d'éviter l'appel intermédiaire à la recherche et de valider et détruire directement toutes les politiques inutilisées dans un espace de travail.

Il est fortement recommandé d'utiliser les ID de politiques et de filtres dans la mesure du possible et de ne pas générer manuellement des identifiants de politiques pour les points terminaux des statistiques de politique ou de l'API Destroy with Identifier (Détruire avec identifiant). Cependant, l'exemple suivant illustre une façon de générer des identifiants de politiques à partir de l'objet de politiques :

```
resp = restclient.get(f'/policies/631b0590497d4f09b537b973')
policy = resp.json() # policy object
policy_identifer = {
    'consumer_consistent_uuid': policy['consumer_filter']['consistent_uuid'],
    'provider_consistent_uuid': policy['provider_filter']['consistent_uuid'],
    'rank': policy['rank'],
    'action': policy['action'],
    'priority': policy['priority'],
    'protocol': policy['l4_params'][0]['proto'],
    'start_port': policy['l4_params'][0]['port'][0],
    'end_port': policy['l4_params'][0]['port'][1]
}
```

Remarque Après la publication d'une nouvelle version de l'espace de travail d'application, six heures peuvent s'écouler avant que les résultats ne soient disponibles. Toutes les résolutions d'horodatage auront également une granularité minimale de 6 heures.

Pour obtenir les politiques inutilisées des versions appliquées d'un espace de travail d'application, le chemin URL est :

```
POST /openapi/v1/unused_policies/{application_id}/enforced
```

Pour obtenir les politiques inutilisées des versions analysées d'un espace de travail d'application, le chemin URL est :

```
POST /openapi/v1/unused_policies/{application_id}/analyzed
```

Le corps de la requête se compose d'un corps JSON comportant le schéma suivant :

Nom	Type	Description
t0	chaîne	Le début de l'intervalle de temps, au format RFC 3339.

Nom	Type	Description
t1	chaîne	(Facultatif) La fin de l'intervalle de temps au format RFC-3339; correspond par défaut à l'heure actuelle, si elle n'est pas précisée.
limit	nombre entier	(Facultatif) Limite le nombre de politiques par demande.
offset	chaîne	(Facultatif) Décalage reçu de la réponse précédente - utile pour la pagination.

```
application_id = '62e1915e755f026f2bcdd805'
resp = restclient.post(f'/unused_policies/{application_id}/analyzed', json_body=f'''
{{
    "t0":"2022-07-06T00:00:00Z",
    "t1":"2022-07-28T19:00:00Z"
}}''')'
```

Exemple de réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Clés	Valeurs
application_id	ID de l'espace de travail de l'application.
policy_identifiers	Une liste des identifiants des politiques inutilisées
offset	Décalage de réponse à transmettre pour la page de résultats suivante.

Pour générer la page de résultats suivante, prenez l'objet reçu par la réponse dans « offset » et transmettez-le comme valeur pour le décalage de la prochaine requête.


```
{
  "application_id": "63054a97497d4f2dc113a9c4",
  "policy_identifiers": [
    {
      "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
      "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
      "version": "p1",
      "rank": "DEFAULT",
      "policy_action": "ALLOW",
      "priority": 10,
      "proto": 6,
      "start_port": 10000,
      "end_port": 10000,
      "agg_start_version": 1,
      "agg_start_time": "2022-08-10T23:00:00.000Z"
    },
    {
      "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
      "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
      "version": "p1",
      "rank": "DEFAULT",
```

```
        "policy_action": "ALLOW",
        "priority": 10,
        "protocol": 6,
        "start_port": 10001,
        "end_port": 10001,
        "agg_start_version": 1,
        "agg_start_time": "2022-08-10T23:00:00.000Z"
      }
    ],
    "offset": "eyJvZmZzZXQiOjZ9"
  }
```


Modèles de politiques

Cet ensemble d'API peut être utilisé pour ajouter, modifier ou supprimer des modèles de politique et nécessite la capacité app_policy_management associée à la clé API.

Obtenir des modèles de politiques

Ce point terminal renvoie une liste des modèles de politique pour une portée racine particulière. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/application_templates?root_app_scope_id={root_app_scope_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_app_scope_id	chaîne	Identificateur unique de la portée racine.

Objet de réponse : renvoie une liste des objets de modèle de politique pour la portée racine spécifiée.

Exemple de code Python

```
root_app_scope_id = '<root-app-scope-id>'
restclient.get('/application_templates?root_app_scope_id=%s' % root_app_scope_id)
```

Obtenir un modèle de politique spécifique

Ce point terminal renvoie une instance des modèles de politique.

```
GET /openapi/v1/application_templates/{template_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
template_id	chaîne	Identificateur unique du modèle de politique.

Objet de réponse : renvoie l'objet de modèle de politique avec l'ID spécifié.

Exemple de code Python

```
template_id = '<template-id>'
```

```
restclient.get('/application_templates/%s' % template_id)
```

Créer un modèle de politique

Ce point terminal est utilisé pour créer un nouveau modèle de politique.

```
POST /openapi/v1/application_templates
```

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
name	chaîne	Utilisé comme nom du modèle lors de l'importation.
description	chaîne	(Facultatif) Description du modèle affichée pendant le processus d'application
paramètres	objet Paramètres	Paramètres du modèle, voir ci-dessous.
absolute_policies	tableau d'objets politiques	(Facultatif) Tableau de politiques absolues.
default_policies	tableau d'objets politiques	(obligatoire) Le tableau de politiques par défaut peut être vide.

Objet de réponse : renvoie l'objet de modèle de politique créé.

Exemple de code Python

```
root_app_scope_id = '<root-app-scope-id>'
payload = {'root_app_scope_id': root_app_scope_id,
           'name': "policy_name",
           'default_policies': [
              {
                'action': 'ALLOW',
                'priority': 100,
                'l4_params': [
                  {
                    'proto': 17,
                    'port': [80, 90]
                  }
                ]
              }
            ]
          }
restclient.post('/application_templates',
                json_body=json.dumps(payload))
```

Mettre à jour un modèle de politique

Ce point terminal met à jour un modèle de politique.

```
PUT /openapi/v1/application_templates/{template_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
template_id	chaîne	Identificateur unique du modèle de politique.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
name	chaîne	(Facultatif) Utilisé comme nom du modèle lors de l'importation.
description	chaîne	(Facultatif) Description du modèle affichée pendant le processus d'application

Objet de réponse : renvoie l'objet de modèle de politique modifié avec l'ID spécifié.

Exemple de code Python

```
new_name = <new-name>
payload = {'name': new_name}
template_id = '<template-id>'
restclient.post('/application_templates/%s' % template_id,
                json_body=json.dumps(payload))
```

Suppression d'un modèle de politique

Ce point terminal supprime le modèle de politique spécifié.

```
DELETE /openapi/v1/application_templates/{template_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
template_id	chaîne	Identificateur unique du modèle de politique.

Objet de réponse : aucun

Exemple de code Python

```
template_id = '<template-id>'
restclient.delete('/application_templates/%s' % template_id)
```

Télécharger un modèle de politique

Ce point terminal télécharge un modèle de politique.

```
GET /openapi/v1/application_templates/{template_id}/download
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
template_id	chaîne	Identificateur unique du modèle de politique.

Objet de réponse : renvoie la définition complète du modèle de politique avec l'ID spécifié.

Exemple de code Python

```
template_id = '<template-id>'
restclient.get('/application_templates/%s/download' % template_id)
```

Grappes

Cet ensemble d'API peut être utilisé pour ajouter, modifier ou supprimer des grappes, qui sont membres d'espaces de travail (« applications »). Ces renseignements nécessitent la capacité `user_role_scope_management` associée à la clé API.

Objet grappe

Les attributs de l'objet grappe sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique de la grappe.
consistent_uuid	chaîne	Un ID cohérent dans toutes les exécutions de découverte automatique des politiques.
application_id	chaîne	ID de l'espace de travail auquel la grappe appartient.
version	chaîne	La version de l'espace de travail à laquelle la grappe appartient
name	chaîne	Nom de la grappe.
description	chaîne	La description de la grappe.
approved	booléen	Si la grappe a été « approuvée » par l'utilisateur .
query	JSON	Filtre (ou critères de correspondance) associé au filtre en conjonction avec les filtres des portées parentes.
short_query	JSON	Filtre (ou critères de correspondance) associé au filtre.

Attribut	Type	Description
alternate_queries	tableau de requêtes	Autres suggestions de requêtes générées par une découverte automatique des politiques exécutée en mode dynamique.
stocks	tableau de l'inventaire	Si demandé, renvoie l'inventaire des membres de la grappe, y compris l'adresse IP, le nom d'hôte, le vrf_id et l'UUID.

Obtenir des grappes

Ce point terminal renvoie la liste des grappes pour un espace de travail (« application ») particulier. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/clusters
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	ID de l'espace de travail auquel la grappe appartient.
version	chaîne	Indique la version de l'espace de travail pour lequel obtenir les grappes.
include_inventory	booléen	Inclure l'inventaire des grappes.

Objet de réponse : renvoie un tableau de toutes les grappes pour cet espace de travail et cette version spécifiques.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s/clusters' % application_id)
```

Obtenir une grappe spécifique

Ce point terminal renvoie une instance d'une grappe.

```
GET /openapi/v1/clusters/{cluster_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
cluster_id	chaîne	Identifiant unique de la grappe.
include_inventory	booléen	Inclure l'inventaire des grappes.

Objet de réponse : renvoie l'objet grappe associé à l'ID spécifié.

Exemple de code Python

```
cluster_id = '5d02d021497d4f0949ba74e4'
restclient.get('/clusters/%s' % cluster_id)
```

Créer une grappe

Ce point terminal est utilisé pour créer une nouvelle grappe.

```
POST /openapi/v1/applications/{application_id}/clusters
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	ID de l'espace de travail auquel la grappe appartient.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
name	chaîne	Nom de la grappe.
version	chaîne	Indique la version de l'espace de travail auquel la grappe sera ajoutée.
description	chaîne	(facultatif) La description de la grappe.
approved	booléen	(Facultatif) Une grappe approuvée ne sera pas mise à jour lors d'une recherche automatique des politiques. La valeur par défaut est faux.
query	JSON	Filtre (ou critères de correspondance) associé au filtre. L'autre mode de requête (également appelé mode dynamique) doit être activé sur l'espace de travail, sinon sera ignoré.
query	JSON	Filtre (ou critères de correspondance) associé au filtre. L'autre mode de requête (également appelé mode dynamique) doit être activé sur l'espace de travail, sinon sera ignoré.

Attribut	Type	Description
nodes	Tableau	Liste des adresses IP ou des points terminaux. Sera utilisé pour créer la requête correspondant à ces adresses IP, sauf si une requête est fournie et que l'espace de travail est en mode dynamique.

Attributs de l'objet nœud :

Nom	Type	Description
ip	chaîne	Adresse IP
name	chaîne	(Facultatif) Le nom du nœud.
prefix_len	nombre entier	(Facultatif) Masque de sous-réseau.

Note Les nœuds seront utilisés pour créer une requête, sauf si une requête est fournie et que l'espace de travail est en mode dynamique.

Objet de réponse : renvoie l'objet de grappe nouvellement créé.

Exemple de code Python

```
application_id = '5d02b493755f0237a3d6e078'
payload = {
  'name': 'test_cluster',
  'version': 'v2',
  'description': 'basic granularity',
  'approved': False,
  'query': {
      'type': 'eq',
      'field': 'host_name',
      'value': 'centos6001'
  }
}
restclient.post('/applications/%s/clusters' % application_id)
```

Mettre à jour une grappe

Ce point terminal met à jour une grappe.

```
PUT /openapi/v1/clusters/{cluster_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
cluster_id	chaîne	Identifiant unique de la grappe.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
name	chaîne	Nom de la grappe.
description	chaîne	(facultatif) La description de la grappe.
approved	booléen	Une grappe approuvée ne sera pas mise à jour lors d'une exécution de découverte automatique des politiques.
query	JSON	Filtre (ou critères de correspondance) associé au filtre. L'autre mode de requête (également appelé mode dynamique) doit être activé sur l'espace de travail, sinon sera ignoré.

Objet de réponse : renvoie l'objet de grappe modifié associé à l'ID spécifié.

Exemple de code Python

```
cluster_id = '5d02d2a4497d4f5194f104ef'
payload = {
  'name': 'new_test_cluster',
}
restclient.put('/clusters/%s' % cluster_id, json_body=json.dumps(payload))
```

Suppression d'une grappe

Ce point terminal supprime la grappe spécifiée. Si la grappe est utilisée par une politique, la grappe ne sera pas supprimée et une liste des entités dépendantes sera renvoyée.

```
DELETE /openapi/v1/clusters/{cluster_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
cluster_id	chaîne	Identifiant unique de la grappe.

Objet de réponse : aucun

Exemple de code Python

```
cluster_id = '5d02d2a4497d4f5194f104ef'
restclient.delete('/clusters/%s' % cluster_id)
```

Conversations

Les conversations sont des flux agrégés dans la plage temporelle d'une exécution de découverte automatique de politique où le port consommateur est supprimé. Vous trouverez une description plus détaillée des conversations dans la section Conversations.

Cette API vous permet de rechercher les conversations générées lors d'une exécution de découverte automatique des politiques pour un espace de travail donné. Elle nécessite la capacité `app_policy_management` associée à la clé API pour appeler cette API.

Rechercher des conversations dans une exécution de découverte de politiques

Ce point de terminaison vous permet de rechercher les conversations dans une exécution de découverte automatique des politiques pour un espace de travail donné. Vous pouvez également spécifier un sous-ensemble de dimensions et de mesures prises en charge que vous souhaitez peut-être voir dans les conversations téléchargées. Vous pouvez également rechercher un sous-ensemble de conversations en utilisant des filtres sur les dimensions et les statistiques prises en charge.

```
POST /openapi/v1/conversations/{application_id}
```

La requête consiste en un corps JSON avec les clés suivantes.

Nom	Type	Description
version	nombre entier	Version de l'exécution de découverte automatique des politiques
filter	JSON	(Facultatif) Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à toutes les conversations. Des conversations plus spécifiques peuvent être téléchargées en utilisant des filtres sur les dimensions et les mesures prises en charge. Pour la syntaxe des filtres, reportez-vous à la section Filtres .
dimensions	tableau	(facultatif) Liste des dimensions à renvoyer pour les conversations téléchargées. La liste des dimensions prises en charge se trouve Dimensions prises en charge .
metrics	tableau	(Facultatif) Liste des paramètres à renvoyer pour les conversations téléchargées. La liste des mesures prises en charge se trouve Mesures prises en charge .
limit	nombre entier	(Facultatif) Nombre de conversations à afficher dans une seule réponse d'API.
offset	chaîne	(Facultatif) Décalage reçu de la réponse précédente : utile pour la pagination.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "version": 1,
    "filter": {
        "type": "and",
        "filters":[
            {
                "type": "eq",
                "field": "excluded",
                "value": False
            },
            {
                "type": "eq",
                "field": "protocol",
                "value": "TCP"
            },
        ]
    },
    "dimensions": ["src_ip", "dst_ip", "port"],
    "metrics": ["byte_count", "packet_count"],
    "limit" : 2,
    "offset": <offset-object>
}
```

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Clés	Valeurs
offset	Décalage de réponse à transmettre pour la page de résultats suivante
results	Liste des résultats

Pour générer la page de résultats suivante, prenez l'objet reçu par la réponse dans « `offset` » et transmettez-le comme valeur pour le `décalage` de la prochaine requête.

```
   req_payload = {"version": 1,
                  "limit": 10,
                  "filter": {"type": "and",
                      "filters": [
                          {"type": "eq", "field": "excluded", "value": False},
                          {"type": "eq", "field": "protocol", "value": "TCP"}
                       ]
                  }
    }

    resp = restclient.post('/conversations/{application_id}',
json_body=json.dumps(req_payload))
    print resp.status_code
    if resp.status_code == 200:
        parsed_resp = json.loads(resp.content)
        print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

N principales conversations dans une exécution de découverte de politiques

Ce point terminal vous permet de rechercher dans les principales conversations une découverte automatique des politiques exécutée pour un espace de travail donné en fonction d'une mesure et regroupées par dimension. Les mesures actuellement prises en charge sont Mesures prises en charge et les groupes par dimensions actuellement pris en charge sont Dimensions prises en charge vous pouvez interroger un sous-ensemble de conversations en utilisant des filtres sur les dimensions et les mesures prises en charge. Par exemple, vous pouvez rechercher l'adresse IP source avec le plus grand nombre de conversations de trafic d'octets en utilisant une requête avec la dimension `src_ip` et la mesure `byte_count`.

```
POST /openapi/v1/conversations/{application_id}/topn
```

La requête consiste en un corps JSON avec les clés suivantes.

Nom	Type	Description
version	nombre entier	Version de l'exécution de découverte automatique des politiques
dimension	chaîne	La dimension selon laquelle les conversations doivent être regroupées pour les N premières requêtes. Dimensions prises en charge : src_ip, dst_ip
metric	chaîne	La mesure de tri des N principales conversations. La liste des mesures prises en charge se trouve Mesures prises en charge .
filter	JSON	(Facultatif) Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à toutes les conversations. Des conversations plus spécifiques peuvent être téléchargées en utilisant des filtres sur les dimensions et les mesures prises en charge. Pour la syntaxe des filtres, consultez la section Filtres .
seuil	nombre entier	Nombre des N principaux résultats à renvoyer dans une seule réponse API.

Le corps de la demande doit être au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "version": 1,
    "dimension": "src_ip",
    "metric": "byte_count",
    "filter": {
```

```
            "type": "and",
            "filters":[
                {
                    "type": "eq",
                    "field": "excluded",
                    "value": False
                },
                {
                    "type": "eq",
                    "field": "protocol",
                    "value": "TCP"
                },
            ]
        },
        "threshold" : 10
    }
```

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Clés	Valeurs
results	Liste avec un objet JSON avec une clé de résultats et une valeur d'une liste d'objets de résultats avec des clés correspondant à la dimension et à la mesure de la requête.

```
  [ {"result": [
    {
      "byte_count": 1795195565,
      "src_ip": "192.168.1.6"
    },
    {
      "byte_count": 1781002379,
      "src_ip": "192.168.1.28"
    },
    ...
  ] } ]
```

```
    req_payload = {"version": 1, "dimension": "src_ip", "metric": "byte_count",
      "filter": {"type": "and",
        "filters": [
          {"type": "eq", "field": "excluded", "value": False},
          {"type": "eq", "field": "protocol", "value": "TCP"},
         {"type": "eq", "field": "consumer_filter_id", "value": "16b12a5614c5af5b68afa7ce"},

          {"type": "subnet", "field": "src_ip", "value": "192.168.1.0/24"}
        ]
      },
      "threshold" : 10
    }

    resp = restclient.post('/conversations/{application_id}/topn',
json_body=json.dumps(req_payload))
    print resp.status_code
    if resp.status_code == 200:
        parsed_resp = json.loads(resp.content)
        print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Dimensions prises en charge

Nom	Type	Description
src_ip	chaîne	Adresse IP du consommateur
dst-ip	chaîne	Adresse IP du fournisseur
protocol	chaîne	Protocole utilisé dans la communication. Ex : « TCP », « UDP », etc.
port	nombre entier	Port du fournisseur.
address_type	chaîne	« IPv4 » ou « IPv6 »
consumer_filter_id	chaîne	ID de grappe de la grappe si l'adresse IP du consommateur appartient à une grappe, sinon l'ID de portée auquel l'adresse IP du consommateur appartient.
provider_filter_id	chaîne	ID de grappe de la grappe si l'adresse IP du fournisseur appartient à une grappe, sinon l'ID de la portée auquel l'adresse IP du fournisseur appartient.
excluded	booléen	Indique si cette conversation est exclue lors de la génération des politiques.
confidence	double	Le niveau de confiance de la classification des consommateurs et des fournisseurs. La valeur varie de 0,0 à 1,0, 1,0 étant la classification la plus sûre.

Mesures prises en charge

Nom	Type	Description
byte_count	nombre entier	Nombre total d'octets dans la conversation
packet_count	nombre entier	Nombre total de paquets dans la conversation

Filtres d'exclusion

Cet ensemble d'API peut être utilisé pour ajouter, modifier ou supprimer des filtres d'exclusion et nécessiter la capacité `user_role_scope_management` associée à la clé API.

Les filtres d'exclusion excluent les flux de l'algorithme de mise en grappe de la découverte automatique des politiques. Consultez la section Filtres d'exclusion pour en savoir plus.

Objet filtre d'exclusion

Les attributs de l'objet du filtre d'exclusion sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique de la grappe.
application_id	chaîne	ID de l'espace de travail auquel le filtre d'exclusion appartient.
version	chaîne	La version de l'espace de travail à laquelle appartient le filtre d'exclusion.
consumer_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme consommateur d'une politique.
provider_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme fournisseur d'une politique.
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
port	tableau	Gamme de ports inclusive. ex., [80, 80] ou [5000, 6000]. NULL signifie tous les ports.
updated_at	nombre entier	Horodatage Unix de la mise à jour du filtre d'exclusion.

Obtenir les filtres d'exclusion

Ce point terminal renvoie une liste des filtres d'exclusion pour un espace de travail particulier. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/applications/{application_id}/exclusion_filters
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.
version	chaîne	Indique la version de l'espace de travail pour lequel obtenir les filtres d'exclusion.

Objet de réponse : renvoie une liste des objets filtre d'exclusion pour l'espace de travail et la version spécifiés.

Exemple de code Python

```
application_id = '<application-id>'
params = {'version': 'v10'}
restclient.get('/applications/%s/exclusion_filters' % application_id,
               params=params)
```

Obtenir un filtre d'exclusion spécifique

Ce point terminal renvoie une instance de filtres d'exclusion.

```
GET /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion.

Objet de réponse : renvoie l'objet de filtre d'exclusion avec l'ID spécifié.

Exemple de code Python

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.get('/exclusion_filters/%s' % exclusion_filter_id)
```

Créer un filtre d'exclusion

Ce point terminal est utilisé pour créer un nouveau filtre d'exclusion.

```
POST /openapi/v1/applications/{application_id}/exclusion_filters
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
application_id	chaîne	Identificateur unique de l'espace de travail.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
version	chaîne	La version de l'espace de travail à laquelle appartient le filtre d'exclusion.

Attribut	Type	Description
consumer_filter_id	chaîne	(Facultatif) ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme consommateur d'une politique.
provider_filter_id	chaîne	(Facultatif) ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme fournisseur d'une politique.
proto	nombre entier	(Facultatif) Valeur entière du protocole (NULL signifie tous les protocoles).
start_port	nombre entier	(Facultatif) Port de début de la plage.
end_port	nombre entier	(Facultatif) Port de fin de la plage.

Les paramètres facultatifs manquants seront considérés comme des caractères génériques (correspondent à n'importe lequel).

Objet de réponse : renvoie l'objet filtre d'exclusion créé.

Exemple de code Python

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'version': 'v0',
           'consumer_filter_id': consumer_filter_id,
           'provider_filter_id': provider_filter_id,
           'proto': 6,
           'start_port': 800,
           'end_port': 1000}
application_id = '<application-id>'
restclient.post('/applications/%s/exclusion_filters' % application_id,
                json_body=json.dumps(payload))
```

Mettre à jour un filtre d'exclusion

Ce point terminal met à jour un filtre d'exclusion.

```
PUT /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
consumer_filter_id	chaîne	(Facultatif) ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme consommateur d'une politique.
provider_filter_id	chaîne	(Facultatif) ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme fournisseur d'une politique.
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
start_port	nombre entier	(Facultatif) Port de début de la plage.
end_port	nombre entier	(Facultatif) Port de fin de la plage.

Objet de réponse : renvoie l'objet de filtre d'exclusion modifié avec l'ID spécifié.

Exemple de code Python

```
payload = {'proto': 17}
exclusion_filter_id = '<exclusion-filter-id>'
restclient.post('/exclusion_filters/%s' % exclusion_filter_id,
                json_body=json.dumps(payload))
```

Suppression d'un filtre d'exclusion

Ce point terminal supprime le filtre d'exclusion spécifié.

```
DELETE /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion.

Objet de réponse : aucun

Exemple de code Python

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.delete('/exclusion_filters/%s' % exclusion_filter_id)
```

Filtres d'exclusion par défaut

Cet ensemble d'API peut être utilisé pour ajouter, modifier ou supprimer des filtres d'exclusion par défaut et nécessiter la capacité `app_policy_management` associée à la clé API.

Les filtres d'exclusion excluent les flux de l'algorithme de mise en grappe de la découverte automatique des politiques. Consultez la section Filtres d'exclusion pour en savoir plus.

Objet filtre d'exclusion par défaut

Les attributs de l'objet du filtre d'exclusion sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique du filtre d'exclusion par défaut.
consumer_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme consommateur d'une politique.
provider_filter_id	chaîne	ID d'un filtre défini. Actuellement, toute grappe appartenant à l'espace de travail, au filtre défini par l'utilisateur ou à la portée peut être utilisée comme fournisseur d'une politique.
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
port	tableau	Gamme de ports inclusive. ex., [80, 80] ou [5000, 6000]. NULL signifie tous les ports.
updated_at	nombre entier	Horodatage Unix de la mise à jour du filtre d'exclusion.

Obtenir les filtres d'exclusion par défaut

Ce point terminal renvoie une liste de filtres d'exclusion par défaut. Cette API est disponible pour les clés API avec la capacité `app_policy_management`.

```
GET /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_app_scope_id	chaîne	Identificateur unique de la portée racine.

Objet de réponse : renvoie une liste des objets de filtre d'exclusion par défaut pour la portée racine.

Exemple de code Python

```
root_app_scope_id = '<root-app-scope-id>'
 restclient.get('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id)
```

Obtenir un filtre d'exclusion par défaut spécifique

Ce point terminal renvoie une instance de filtres d'exclusion par défaut.

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'

restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
default_exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion.

Objet de réponse : renvoie l'objet de filtre d'exclusion par défaut avec l'ID précisé.

Exemple de code Python

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Créer un filtre d'exclusion par défaut

Ce point terminal est utilisé pour créer un nouveau filtre d'exclusion par défaut.

```
POST /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_app_scope_id	chaîne	Identificateur unique de la portée racine.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
consumer_filter_id	chaîne	(facultatif) ID d'une portée ou d'un inventaire défini.
provider_filter_id	chaîne	(facultatif) ID d'une portée ou d'un inventaire défini.
proto	nombre entier	(facultatif) Valeur entière de protocole (NULL signifie tous les protocoles).
start_port	nombre entier	(Facultatif) Port de début de la plage.

Attribut	Type	Description
end_port	nombre entier	(Facultatif) Port de fin de la plage.

Objet de réponse : renvoie l'objet filtre d'exclusion par défaut créé.

Exemple de code Python

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'consumer_filter_id': consumer_filter_id,
           'provider_filter_id': provider_filter_id,
           'proto': 6,
           'start_port': 800,
           'end_port': 1000}
root_app_scope_id = '<root-app-scope-id>'
restclient.post('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id,
                json_body=json.dumps(payload))
```

Mettre à jour un filtre d'exclusion par défaut

Ce point terminal met à jour un filtre d'exclusion par défaut.

```
PUT /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
default_exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion par défaut.

Le corps de la requête JSON contient les clés suivantes :

Attribut	Type	Description
consumer_filter_id	chaîne	(facultatif) L'ID d'une portée ou d'un filtre d'inventaire défini.
provider_filter_id	chaîne	(facultatif) L'ID d'une portée ou d'un filtre d'inventaire défini.
proto	nombre entier	Valeur entière du protocole (NULL signifie tous les protocoles).
start_port	nombre entier	(Facultatif) Port de début de la plage.
end_port	nombre entier	(Facultatif) Port de fin de la plage.

Objet de réponse : renvoie l'objet de filtre d'exclusion par défaut modifié avec l'ID précisé.

Exemple de code Python

```
payload = {'proto': 17}
default_exclusion_filter_id = '<default-exclusion-filter-id>'
```

```
restclient.post('/default_exclusion_filters/%s' % default_exclusion_filter_id,
                json_body=json.dumps(payload))
```

Suppression d'un filtre d'exclusion par défaut

Ce point terminal supprime le filtre d'exclusion par défaut spécifié.

```
DELETE /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
default_exclusion_filter_id	chaîne	Identificateur unique du filtre d'exclusion.

Objet de réponse : aucun

Exemple de code Python

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.delete('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Analyse en temps réel

L'analyse en direct (dite aussi en temps réel) ou l'analyse des politiques est un aspect important de la génération de politiques de sécurité. Elle vous permet d'évaluer l'incidence d'un ensemble de politiques (lorsqu'elles sont générées par la découverte automatique des politiques ou ajoutées manuellement par les utilisateurs) avant d'appliquer réellement ces politiques sur les charges de travail. L'analyse en direct permet aux utilisateurs d'exécuter une analyse de simulation sur le trafic en direct sans perturber le trafic des applications.

L'ensemble d'API disponibles dans cette section permet le téléchargement de flux et de l'effet de l'ensemble actuel de politiques publiées dans un espace de travail sur ces flux. La capacité `app_policy_management` de la clé API est nécessaire pour appeler cet ensemble d'API.

Les flux disponibles via l'analyse en direct ont certains attributs (dimensions et mesures) et l'API de téléchargement permet à l'utilisateur de filtrer les flux en fonction de différents critères sur les dimensions.

Dimensions de flux disponibles dans l'analyse en temps réel

Ce point terminal est utile pour connaître les colonnes sur lesquelles les critères de recherche (ou *filtres*) peuvent être spécifiés pour le téléchargement des flux disponibles via l'analyse en direct. Le scénario d'utilisation le plus courant serait de télécharger des flux *permitted (autorisés)*, *escaped (échappés)* ou *rejected (rejetés)* . Il est possible d'y parvenir en transmettant à l'API de téléchargement un critère de recherche sur la dimension de la catégorie. Lorsque l'API est utilisée avec **type: eq**, les catégories entrantes et sortantes du flux doivent correspondre. Utilisé avec **type: contains** la catégorie des flux entrants ou sortants doit correspondre

```
GET /openapi/v1/live_analysis/dimensions
```

Indicateurs de flux disponibles dans l'analyse en temps réel

Ce point terminal renvoie la liste des mesures (p. ex., nombre d'octets, nombre de paquets) associées à l'analyse en direct. Un scénario d'utilisation de ce point terminal consisterait à projeter un sous-ensemble de mesures

dans l'API de téléchargement, c'est-à-dire qu'au lieu de télécharger toutes les mesures, les utilisateurs peuvent spécifier un sous-ensemble plus restreint de mesures qui les intéressent.

```
GET /openapi/v1/live_analysis/metrics
```

Télécharger les flux disponibles via l'analyse en temps réel

Ce point terminal renvoie la liste des flux correspondant aux critères de filtre. Chaque objet de flux dans le résultat possède des attributs qui sont une union des dimensions d'analyse en direct (renvoyées par l'API des dimensions d'analyse en direct ci-dessus) ainsi que des mesures d'analyse en direct (renvoyées par l'API des métriques d'analyse en direct ci-dessus). Si vous le souhaitez, l'utilisateur peut également préciser un petit sous-ensemble de dimensions ou de métriques s'il n'est pas intéressé par l'ensemble complet de dimensions et de métriques disponibles. Cette prévision d'un sous-ensemble plus petit de dimensions ou de métriques a également pour effet secondaire d'accélérer les appels d'API.

```
POST /openapi/v1/live_analysis/{application_id}
```

Le corps de la requête se compose d'un corps JSON avec les clés suivantes.

Nom	Type	Description
t0	entier ou chaîne	Début de l'intervalle de temps (heure d'origine ou ISO 8601)
t1	entier ou chaîne	Fin de l'intervalle de temps (heure d'origine ou ISO 8601)
filter	JSON	Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à tous les flux. Reportez-vous à la section sur les Filtres dans la recherche de flux pour connaître la syntaxe des filtres.
dimensions	tableau	(facultatif) Liste des dimensions de flux à renvoyer pour les flux téléchargés, disponible par le biais de l'analyse en direct. Si ce n'est pas spécifié, toutes les dimensions disponibles sont affichées.
metrics	tableau	(facultatif) Liste des mesures de flux à renvoyer pour les flux téléchargés, disponible par le biais de l'analyse en direct.
limit	nombre entier	(facultatif) Nombre de flux à renvoyer en une seule réponse API.
offset	chaîne	(Facultatif) Décalage reçu de la réponse précédente : utile pour la pagination.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "category",
                "value": "escaped"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "limit": 100,
    "offset": <offset-object>
}
```

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Clés	Valeurs
offset	Décalage de réponse à transmettre pour la page de résultats suivante
results	Liste des résultats

Pour générer la page de résultats suivante, prenez l'objet reçu par la réponse dans « `offset` » et transmettez-le comme valeur pour le `décalage` de la prochaine requête.

Exemple de code Python

```
   req_payload = {"t0": "2016-11-07T09:00:00-0700",
                  "t1": "2016-11-07T19:00:00-0700",
                  "limit": 10,
                  "filter": {"type": "and",
                      "filters": [
                          {"type": "contains", "field": "category", "value": "escaped"},
                          {"type": "regex", "field": "src_hostname", "value": "web*"}
                       ]
                  }
    }

    resp = restclient.post('/live_analysis/{application_id}',
json_body=json.dumps(req_payload))
    print resp.status_code
    if resp.status_code == 200:
        parsed_resp = json.loads(resp.content)
        print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Portées

Cet ensemble d'API peut être utilisé pour gérer les portées (ou AppScopes) au sein de déploiement de grappes Cisco Secure Workload. Ces renseignements nécessitent la capacité `user_role_scope_management` associée à la clé API. L'API pour obtenir la liste des portées est également disponible pour les clés API avec la capacité `app_policy_management` ou `sensor_management`.

Objet portée

Les attributs de l'objet portée sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique de la portée.
short_name	chaîne	Nom spécifié par l'utilisateur de la portée.
name	chaîne	Nom complet du domaine de la portée. Il s'agit d'un nom complet, c'est-à-dire qu'il comporte le nom des portées parentes (le cas échéant) jusqu'à la portée racine.
description	chaîne	Description de la portée précisée par l'utilisateur.
short_query	JSON	Filtre (ou critères de correspondance) associé à la portée.
query	JSON	Le filtre (ou les critères de correspondance) associé à la portée conjointement avec les filtres des portées parents (jusqu'à la portée racine).
vrf_id	nombre entier	ID du VRF auquel la portée appartient.
parent_app_scope_id	chaîne	ID de la portée parente.
child_app_scope_ids	tableau	Un tableau d'ID d'enfants de la portée.
policy_priority		Utilisé pour trier les priorités de l'espace de travail. Reportez-vous à la section Sémantique et affichage.
dirty	booléen	Indique qu'une requête parent ou enfant a été mise à jour et que les modifications doivent être validées.

Attribut	Type	Description
dirty_short_query	JSON	Non NULL si la requête pour cette portée a été mise à jour mais n'est pas encore validée.

Obtenir les portées

Ce point terminal renvoie une liste des portées connues de l'appareil Cisco Secure Workload. Cette API est disponible pour les clés API avec la capacité `app_policy_management` ou `user_role_scope_management`.

```
GET /openapi/v1/app_scopes
```

Paramètres :

Nom	Type	Description
vrf_id	nombre entier	Correspondance des portées d'application en fonction de l'identifiant vrf_id.
root_app_scope_id	chaîne	Correspondance des portées d'application en fonction de l'identifiant de la portée de l'application racine.
exact_name	chaîne	Renvoie la portée correspondant au nom exact, en respectant la casse.
exact_short_name	chaîne	Renvoie les portées correspondant exactement à short_name, en respectant la casse.

Renvoie la liste des objets de la portée.

Créer une portée

Ce point terminal est utilisé pour créer de nouvelles portées.

```
GET /openapi/v1/app_scopes
```

Paramètres :

Nom	Type	Description
short_name	chaîne	Nom spécifié par l'utilisateur de la portée.
description	chaîne	Description de la portée précisée par l'utilisateur.
short_query	JSON	Filtre (ou critères de correspondance) associé à la portée.

Nom	Type	Description
parent_app_scope_id	chaîne	ID de la portée parente.
policy_priority	nombre entier	La valeur par défaut est « dernier ». Utilisé pour trier les priorités de l'espace de travail. Voir l'ordonnancement des politiques sous Politiques.

Exemple de code Python

```
req_payload = {
   "short_name": "App Scope Name",
   "short_query": {
       "type":"eq",
       "field":"ip",
       "value": <....>
   },
   "parent_app_scope_id": <parent_app_scope_id>
}
resp = restclient.post('/app_scopes', json_body=json.dumps(req_payload))
```

Pour créer une portée en fonction du sous-réseau, utilisez la commande short_query suivante :

```
 "short_query":
   {
     "type":"subnet",
     "field":"ip",
     "value": "1.0.0.0/8"
   },
```

Obtenir une portée spécifique

Ce point terminal renvoie une instance d'une portée.

`GET /openapi/v1/app_scopes/{app_scope_id}`

Renvoie l'objet de portée associé à l'ID spécifié.

Mettre à jour une portée

Ce point terminal met à jour une portée. Les modifications apportées au `nom` et à la `description` sont appliquées immédiatement. Les modifications apportées à short_query marquent la portée comme « dirty » (sale) et définissent l'attribut « `dirty_court_query` ». Une fois que toutes les modifications de requête de portée, dans une portée racine donnée, sont effectuées, il faut envoyer un message ping au point de terminaison Valider les modifications de requête de portée (Valider les modifications de requête de portée) pour valider toutes les mises à jour requises.

`PUT /openapi/v1/app_scopes/{app_scope_id}`

Paramètres :

Nom	Type	Description
short_name	chaîne	Nom spécifié par l'utilisateur de la portée.
description	chaîne	Description de la portée précisée par l'utilisateur.
short_query	JSON	Filtre (ou critères de correspondance) associé à la portée.

Renvoie l'objet de portée modifié associé à l'ID spécifié.

Supprimer une portée spécifique

Ce point terminal supprime la portée spécifiée.

```
DELETE /openapi/v1/app_scopes/{app_scope_id}
```

Si la portée est associée à un espace de travail, à une politique, à un filtre d'inventaire utilisateur, etc., ce point terminal renverra `422 Unprocessable Entity` (422 Entité non traitable). L'objet Erreur renvoyé contiendra un attribut `détails` avec le nombre d'objets dépendants ainsi que les identifiants des 10 premiers de chaque type. Ces renseignements peuvent être utilisés pour localiser et supprimer les dépendances bloquantes.

Obtenir les portées par ordre de priorité des politiques

Ce point terminal répertorie les portées dans l'ordre dans lequel leur espace de travail principal correspondant sera mis en application.

```
POST /openapi/v1/app_scopes/ {root_app_scope_id}/politique_commande
```

Renvoie un tableau des objets de la portée.

Mettre à jour l'ordre de la politique

Ce point terminal mettra à jour l'ordre dans lequel les politiques sont appliquées.

Warning Ce point terminal modifie l'ordre dans lequel les politiques sont appliquées. Par conséquent, de nouvelles règles de pare-feu hôte seront insérées et toutes les règles existantes seront supprimées sur les hôtes concernés.

```
POST /openapi/v1/app_scopes/ {root_app_scope_id}/politique_commande
```

Paramètres :

Nom	Type	Description
root_app_scope_id	chaîne	Portée racine ou dont l'ordre est en cours de modification.

Nom	Type	Description
ids	tableau	un tableau de chaînes d'ID de portée dans l'ordre dans lequel elles doivent être appliquées.

Le paramètre de tableau `ids` doit inclure tous les membres de la portée racine, y compris la racine.

Valider les modifications de requête de portée

Ce point terminal déclenche une tâche en arrière-plan asynchrone pour mettre à jour tous les enfants « modifiés » d'une portée racine donnée. Cette tâche met à jour les portées et les espaces de travail. Consultez Scopes (Portées) pour plus de détails.

```
POST /openapi/v1/app_scopes/commit_dirty
```

Paramètres :

Nom	Type	Description
root_app_scope_id	chaîne	ID d'une portée racine dont tous les enfants seront mis à jour.
sync	booléen	(Facultatif) Indiquez si la requête doit être synchrone.

Renvoie 202 pour indiquer que la tâche a été mise en file d'attente. Pour vérifier si la tâche est terminée, interrogez l'attribut « dirty » de la portée racine pour voir s'il a été défini sur faux.

Les utilisateurs peuvent transmettre le paramètre `sync` (synchronisation) pour que la tâche soit exécutée immédiatement. La demande sera renvoyée une fois terminée avec un code d'état 200. Cette demande peut prendre un certain temps si de nombreuses mises à jour doivent être appliquées.

Envoyer une demande de suggestions de groupe

Envoyer une demande de suggestions de groupe pour une porter.

```
PUT /openapi/v1/app_scopes/{app_scope_id}/suggest_groups
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
app_scope_id	chaîne	Identificateur unique pour la portée.

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
start_time	chaîne	Heure de début de l'intervalle de saisie de suggestions de groupe.
end_time	chaîne	Heure de fin de l'intervalle de saisie de suggestions de groupe.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
message	chaîne	Message concernant la réussite ou l'échec de l'envoi de la demande de suggestions de groupe.

Exemple de code Python

```
app_scope_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'start_time': '2020-09-17T10:00:00-0700',
      'end_time': '2020-09-17T11:00:00-0700',
}
resp = restclient.put('/app_scopes/%s/suggest_groups' % app_scope_id,
                      json_body=json.dumps(req_payload))
```

Obtenir l'état de la proposition de groupe

Interroger l'état de proposition de groupe de la portée.

```
GET /openapi/v1/app_scopes/{app_scope_id}/suggest_groups_status
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
app_scope_id	chaîne	Identificateur unique pour la portée.

Objet de réponse : renvoie un objet avec les attributs suivants :

Nom	Type	Description
status	chaîne	État de la proposition de groupe Valeurs : PENDING (EN COURS), COMPLETE (TERMINÉ) ou FAILED (ERREUR)

Exemple de code Python

```
app_scope_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/app_scopes/%s/suggest_groups_status' % app_scope_id)
```

Configurer les alertes

Cet ensemble d'API peut être utilisé pour gérer les alertes d'utilisateurs. Elles nécessitent la capacité `user_alert_management` associée à la clé API.

Objet alerte

Chaque objet de configuration d'alerte contient les champs suivants :

Attribut	Type	Description
app_name	chaîne	Nom de l'application associée à la configuration d'alerte.
règles	objet	Ensemble de conditions qui doivent être respectées pour que la configuration des alertes déclenche une alerte.
subjects	objet	Liste des utilisateurs qui doivent recevoir l'alerte.
gravité	chaîne	Indique le niveau de gravité associé à une configuration d'alerte.
individual_alert	booléen	Indique si des alertes individuelles doivent être envoyées pour déclencher la configuration des alertes.
summary_alert_freq	chaîne	Fréquence des résumés d'alertes à envoyer pour une configuration d'alerte particulière.
alert_type	chaîne	Identifiant unique de la configuration d'alerte.
app_instance_id	chaîne	Identifiant unique d'une instance particulière associée à la configuration d'alerte

Recevoir des alertes

Ce point terminal récupère la liste des configurations d'alertes pour un utilisateur. Les alertes peuvent être filtrées selon une portée racine donnée. Si aucune portée n'est fournie, toutes les alertes sont renvoyées pour toutes les portées auxquelles l'utilisateur a accès. Les alertes du fournisseur de services ne seront renvoyées que si l'utilisateur est un administrateur du site.

```
GET /openapi/v1/alert_confs
```

Paramètres :

Nom	Type	Description
root_app_scope_id	chaîne	(Facultatif) ID d'une portée racine pour renvoyer les alertes uniquement affectées à cette portée.

Objet de réponse : renvoie une liste des objets d'alerte de l'utilisateur.

Créer une alerte

Ce point terminal est utilisé pour créer une nouvelle alerte.

```
POST openapi/v1/alert_confs
```

Paramètres :

Attribut	Type	Description
app_name	chaîne	Nom de l'application associée à la configuration d'alerte.
règles	objet	Ensemble de conditions qui doivent être respectées pour que la configuration des alertes déclenche une alerte.
subjects	objet	Liste des utilisateurs qui doivent recevoir l'alerte.
gravité	chaîne	Indique le niveau de gravité associé à une configuration d'alerte.
individual_alert	booléen	Indique si des alertes individuelles doivent être envoyées pour déclencher la configuration des alertes.
summary_alert_freq	chaîne	Fréquence des résumés d'alertes à envoyer pour une configuration d'alerte particulière.
alert_type	chaîne	Identifiant unique de la configuration d'alerte.
app_instance_id	chaîne	Identifiant unique d'une instance particulière associée à la configuration d'alerte.

L'utilisateur demandeur doit avoir accès à la portée fournie. Une alerte sans portée est une « alerte de fournisseur de services », et seul un administrateur de site peut en créer.

Objet de réponse : renvoie l'objet alerte nouvellement créé.

Obtenir une alerte spécifique

Ce point terminal renvoie un objet d'alerte spécifique.

```
GET /openapi/v1/alert_confs/
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
alert_id	chaîne	Identifie de façon unique l'alerte.

Objet de réponse : renvoie un objet d'alerte associé à l'ID précisé.

Mettre à jour une alerte

Ce point terminal est utilisé pour mettre à jour une alerte existante.

```
PUT /openapi/v1/alert_confs/
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
alert_id	chaîne	Pour récupérer ou modifier les paramètres de configuration de l'alerte,

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Nom spécifié par l'utilisateur pour l'alerte.
description	chaîne	Description fournie par l'utilisateur pour l'alerte.

L'utilisateur demandeur doit avoir accès à la portée fournie. Une alerte sans portée est appelée une « alerte de fournisseur de services » et seul l'administrateur du site peut la mettre à jour.

Objet de réponse : objet d'alerte mis à jour avec l'ID spécifié.

Supprimer une alerte spécifique

Ce point terminal supprime l'alerte spécifiée.

```
DELETE /openapi/v1/alert_confs/{alert_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
alert_id	chaîne	Identifie de façon unique l'alerte.

Objet de réponse : aucun.

Rôles

Cet ensemble d'API peut être utilisé pour gérer les rôles d'utilisateur. Ces renseignements nécessitent la capacité `user_role_scope_management` associée à la clé API.

Note Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

Objet rôle

Les attributs d'objets de rôle sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique du rôle
app_scope_id	chaîne	Portée à laquelle la portée est définie, peut-être vide pour « Rôles de fournisseur de services ».
name	chaîne	Nom indiqué par l'utilisateur pour le rôle.
description	chaîne	Description définie par l'utilisateur pour le rôle.

Obtenir des rôles

Ce point terminal renvoie une liste des rôles accessibles à l'utilisateur. Les rôles peuvent être filtrés selon une portée racine donnée. Si aucune portée n'est fournie, tous les rôles sont retournés, pour toutes les portées auxquelles l'utilisateur a accès. Les rôles de fournisseur de services ne seront renvoyés que si l'utilisateur est un administrateur de site.

```
GET /openapi/v1/roles
```

Paramètres :

Nom	Type	Description
app_scope_id	chaîne	(Facultatif) ID d'une portée racine pour renvoyer les rôles uniquement affectés à cette portée.

Objet de réponse : renvoie une liste des objets rôle d'utilisateur.

Exemple de code Python

```
resp = restclient.get('/roles')
```

Créer un rôle

Ce point terminal est utilisé pour créer un nouveau rôle.

```
POST /openapi/v1/roles
```

Paramètres :

Nom	Type	Description
name	chaîne	Nom indiqué par l'utilisateur pour le rôle.
description	chaîne	Description définie par l'utilisateur pour le rôle.
app_scope_id	chaîne	(Facultatif) L'ID de portée sous lequel le rôle est créé. Si aucun ID de portée n'est mentionné, le rôle est considéré comme un rôle de fournisseur de services.

L'utilisateur demandeur doit avoir accès à la portée fournie. Un rôle sans portée est appelé « rôle de fournisseur de services » et seul l'administrateur du site peut le créer.

Objet de réponse : renvoie l'objet rôle nouvellement créé.

Exemple de code Python

```
app_scope_id = '<app-scope-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
      'app_scope_id': app_scope_id
}
restclient.post('/roles', json_body=json.dumps(req_payload))
```

Obtenir un rôle spécifique

Ce point terminal renvoie un objet de rôle spécifique.

```
GET /openapi/v1/roles/{role_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	Permet d'identifier le rôle de façon unique.

Objet de réponse : renvoie un objet de rôle associé à l'ID spécifié.

Exemple de code Python

```
role_id = '<role-id>'
restclient.get('/roles/%s' % role_id)
```

Mettre à jour un rôle

Ce point terminal est utilisé pour mettre à jour un rôle existant.

```
PUT /openapi/v1/roles/{role_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	Permet d'identifier le rôle de façon unique.

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Nom indiqué par l'utilisateur pour le rôle.
description	chaîne	Description définie par l'utilisateur pour le rôle.

L'utilisateur demandeur doit avoir accès à la portée fournie. Un rôle sans portée est appelé « rôle de fournisseur de services » et seul l'administrateur du site peut le mettre à jour.

Objet de réponse : l'objet rôle mis à jour avec l'ID précisé.

Exemple de code Python

```
role_id = '<role-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
}
restclient.put('/roles/%s' % role_id, json_body=json.dumps(req_payload))
```

Accorder l'accès à un rôle à la portée

Ce point terminal donne à un rôle le niveau d'accès spécifié à une portée.

```
POST /openapi/v1/roles/ {role_id}/capabilities
```

Les capacités ne peuvent être ajoutées qu'aux rôles auxquels l'utilisateur a accès. Si le rôle est affecté à une portée, les capacités doivent correspondre à cette portée ou à ses enfants. Les rôles de fournisseur de services (ceux qui ne sont affectés à aucune portée) peuvent ajouter des fonctionnalités à n'importe quelle portée.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	Permet d'identifier le rôle de façon unique.

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
app_scope_id	chaîne	ID de la portée pour laquelle l'accès est fourni.

Nom	Type	Description
ability	chaîne	Les valeurs possibles sont SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER

Pour obtenir une description plus détaillée des capacités, reportez-vous à la section Rôles .

Objet de réponse :

Nom	Type	Description
app_scope_id	chaîne	ID de la portée pour laquelle l'accès est fourni.
id_rôle	chaîne	ID du rôle.
ability	chaîne	Les valeurs possibles sont SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER
inherited	booléen	

Exemple de code Python

```
role_id = '<role-id>'
req_payload = {
  'app_scope_id': '<app-scope-id>',
  'ability': 'SCOPE_READ'
}
restclient.post('/roles/%s/capabilities' % role_id,
                json_body=json.dumps(req_payload))
```

Supprimer un rôle spécifique

Ce point terminal supprime le rôle spécifié.

```
DELETE /openapi/v1/roles/{role_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	Permet d'identifier le rôle de façon unique.

Objet de réponse : aucun.

Exemple de code Python

```
role_id = '<role-id>'
restclient.delete('/roles/%s' % role_id)
```

Utilisateurs

Cet ensemble d'API gère les utilisateurs. Ces renseignements nécessitent la capacité `user_role_scope_management` associée à la clé API.

Note Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

Objet utilisateur

Les attributs de l'objet utilisateur sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique du rôle
courriel	chaîne	Adresse de courriel associée au compte d'utilisateur.
first_name	chaîne	Prénom
last_name	chaîne	Nom
app_scope_id	chaîne	La portée à laquelle l'utilisateur est affecté. Peut-être vide si l'utilisateur est un « utilisateur de fournisseur de services ».
role_ids	liste	Liste des ID des rôles affectés au compte d'utilisateur.
by-pass_external_auth	booléen	Vrai pour les utilisateurs locaux et faux pour les utilisateurs d'authentification externes (LDAP ou SSO).
disabled_at	nombre entier	Horodatage Unix du moment où l'utilisateur a été désactivé. Zéro ou nul, sinon.

Obtenir des utilisateurs

Ce point terminal renvoie une liste des objets utilisateur connus de l'appareil Cisco Secure Workload.

```
GET /openapi/v1/users
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
include_disabled	booléen	(Facultatif) Pour inclure les utilisateurs désactivés, la valeur par défaut est False.
app_scope_id	chaîne	(Facultatif) Renvoie uniquement les utilisateurs affectés à la portée fournie.

Objet de réponse : renvoie une liste des objets utilisateur. Seuls les administrateurs de site peuvent voir les utilisateurs du fournisseur de services, c.-à-d. ceux qui ne sont affectés à aucune portée.

Exemple de code Python

```
GET /openapi/v1/users
```

Créer un nouveau compte utilisateur

Ce point terminal est utilisé pour créer un nouveau compte d'utilisateur.

```
POST /openapi/v1/users
```

Paramètres : le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
courriel	chaîne	Adresse de courriel associée au compte d'utilisateur.
first_name	chaîne	Prénom
last_name	chaîne	Nom
app_scope_id	chaîne	(Facultatif) Portée racine à laquelle l'utilisateur appartient.
role_ids	liste	(Facultatif) La liste des rôles à attribuer à l'utilisateur.

L'app_scope_id est l'ID de la portée racine à laquelle l'utilisateur doit être affecté. Si app_scope_id n'est pas présent, l'utilisateur est un « utilisateur de fournisseur de services ». Seuls les administrateurs de site peuvent créer des utilisateurs fournisseurs de services. Les `value_ids` sont les ID des rôles qui ont été créés dans la portée d'application spécifiée.

Objet de réponse : renvoie l'objet utilisateur nouvellement créé.

Exemple de code Python

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
    "email": "foo@bar.com"
    "app_scope_id": "root_appscope_id",
```

```
    "role_ids": ["roleid1", "roleid2"]
}
resp = restclient.post('/users', json_body=json.dumps(req_payload))
```

Obtenir un utilisateur spécifique

Ce point terminal renvoie un objet utilisateur spécifique.

```
GET /openapi/v1/users/{user_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur.

Objet de réponse : renvoie un objet utilisateur associé à l'ID spécifié.

Exemple de code Python

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/users/%s' % user_id)
```

Mettre à jour un utilisateur

Ce point terminal met à jour un utilisateur existant.

```
PUT /openapi/v1/users/{user_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur en cours de mise à jour.

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
courriel	chaîne	Adresse de courriel associée au compte d'utilisateur.
first_name	chaîne	Prénom
last_name	chaîne	Nom
app_scope_id	chaîne	ID de portée de l'application racine (autorisé uniquement pour les administrateurs de site)

Objet de réponse : renvoie l'objet utilisateur nouvellement mis à jour.

Exemple de code Python

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
```

```
    "email": "foo@bar.com"
    "app_scope_id": "root_appscope_id",
}
restclient.put('/users', json_body=json.dumps(req_payload))
```

Activer ou réactiver un utilisateur désactivé

Ce point terminal est utilisé pour réactiver un utilisateur désactivé.

```
POST /openapi/v1/users/{user_id}/enable
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur qui est activé.

Objet de réponse : renvoie l'objet utilisateur réactivé associé à l'ID spécifié.

Exemple de code Python

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.post('/users/%s/enable' % user_id)
```

Ajouter un rôle au compte d'utilisateur

Ce point terminal est utilisé pour ajouter un rôle à un compte d'utilisateur.

```
PUT /openapi/v1/users/{user_id}/add_role
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur en cours de modification.

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	ID de l'objet de rôle à ajouter.

Objet de réponse : renvoie l'objet utilisateur modifié associé à l'ID spécifié.

Exemple de code Python

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.put('/users/%s/add_role' % user_id,
                      json_body=json.dumps(req_payload))
```

Supprimer le rôle du compte d'utilisateur

Ce point terminal est utilisé pour supprimer un rôle d'un compte d'utilisateur.

```
DELETE /openapi/v1/users/{user_id}/remove_role
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur en cours de suppression.

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
id_rôle	chaîne	ID de l'objet rôle à supprimer.

Objet de réponse : renvoie l'objet utilisateur modifié associé à l'ID spécifié.

Exemple de code Python

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.delete('/users/%s/remove_role' % user_id,
                         json_body=json.dumps(req_payload))
```

Supprimer un utilisateur spécifique

Ce point terminal supprime le compte d'utilisateur spécifié.

```
DELETE /openapi/v1/users/{user_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
user_id	chaîne	ID de l'objet utilisateur en cours de suppression.

Objet de réponse : renvoie l'objet utilisateur supprimé associé à l'ID spécifié.

Exemple de code Python

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.delete('/users/%s' % user_id)
```

Filtres d'inventaire

Les filtres d'inventaire codent les critères de correspondance pour les requêtes de recherche d'inventaire. Cet ensemble d'API fournit des fonctionnalités similaires à celles décrites dans les filtres d'inventaire. Ils nécessitent la capacité `sensor_management ou app_policy_management` associée à la clé API.

Objet filtre d'inventaire

L'objet JSON du filtre d'inventaire est renvoyé sous forme d'objet unique ou de tableau d'objets selon le point de terminaison d'API. Les attributs de l'objet sont décrits ci-dessous :

Attribut	Type	Description
ID	chaîne	Identifiant unique du filtre d'inventaire.
name	chaîne	Nom spécifié par l'utilisateur pour le filtre d'inventaire.
app_scope_id	chaîne	ID de la portée associée au filtre.
short_query	JSON	Filtre (ou critères de correspondance) associé au filtre.
primary	booléen	Lorsque la valeur est « vrai », le filtre est limité à la portée de la propriété.
public	booléen	Lorsqu'il est défini sur « vrai », le filtre fournit un service pour sa portée. Doit également être principal ou de portée limitée.
query	JSON	Filtre (ou critères de correspondance) associé au filtre en conjonction avec les filtres des portées parentes. Ces conjonctions prennent effet si la case « restricted to ownership scope » (limité à la portée de la propriété) est cochée. Si le champ « primary » (principal) est faux, la requête est identique à short_query.

Obtenir des filtres d'inventaire

Ce point terminal renvoie une liste des filtres d'inventaire visibles par l'utilisateur.

```
POST /openapi/v1/filters/inventories
```

Paramètres :

Nom	Type	Description
vrf_id	nombre entier	Mettre en correspondance les filtres d'inventaire par ID de VRF.
root_app_scope_id	chaîne	Faire correspondre les filtres d'inventaire par ID de portée de l'application racine.
name	chaîne	Renvoie les filtres d'inventaire correspondant à une partie du nom, insensible à la casse.
exact_name	chaîne	Renvoie les filtres d'inventaire correspondant au nom exact, sensibles à la casse.

Créer un filtre d'inventaire

Ce point terminal est utilisé pour créer un filtre d'inventaire.

```
POST /openapi/v1/filters/inventories
```

Paramètres :

Nom	Type	Description
name	chaîne	Nom spécifié par l'utilisateur pour la portée de l'application.
query	JSON	Filtre (ou critères de correspondance) associé au filtre.
app_scope_id	chaîne	ID de la portée associée au filtre.
primary	booléen	Lorsque la valeur est « vrai », le filtre est limité à la portée de la propriété.
public	booléen	Lorsqu'il est défini sur « vrai », le filtre fournit un service pour sa portée. Il doit également s'agir d'un domaine principal ou d'une portée restreinte.

Exemple de code Python

```
req_payload = {
  "app_scope_id": <app_scope_id>,
  "name": "sensor_config_inventory_filter",
  "query": {
    "type": "eq",
    "field": "ip",
    "value": <sensor_interface_ip>
```

```
  },
}
resp = restclient.post('/filters/inventories', json_body=json.dumps(req_payload))
```

Valider une requête de filtre d'inventaire

Ce point terminal validera la structure d'une requête par rapport au schéma requis.

```
POST /openapi/v1/filters/inventories/validate_query
```

Paramètres :

Nom	Type	Description
query	JSON	Filtre (ou critères de correspondance) associé à la portée.

Objet de réponse :

Attribut	Type	Description
valide	booléen	Indique si la requête est valide
erreurs	tableau	Si non valide, détails sur les erreurs

Obtenir un filtre d'inventaire spécifique

Ce point terminal renvoie une instance d'un filtre d'inventaire.

```
GET /openapi/v1/filters/inventories/{inventory_filter_id}
```

Renvoie un objet de filtre d'inventaire associé à l'ID spécifié.

Mettre à jour un filtre d'inventaire spécifique

Ce point terminal est utilisé pour mettre à jour un filtre d'inventaire.

```
PUT /openapi/v1/filters/inventories/{inventory_filter_id}
```

Paramètres :

Nom	Type	Description
name	chaîne	Nom spécifié par l'utilisateur de la portée.
query	JSON	Filtre (ou critères de correspondance) associé à la portée.
app_scope_id	chaîne	ID de la portée associée au filtre.
primary	booléen	Lorsque la valeur est « vrai », le filtre est limité à la portée de la propriété.

Nom	Type	Description
public	booléen	Lorsqu'il est défini sur « vrai », le filtre fournit un service. Peut être utilisé dans le cadre de la génération de politiques. Il doit également s'agir d'un domaine principal ou d'une portée restreinte.
Utilisations	booléen	Collecte des statistiques de politique et de configuration des membres.

Supprimer un filtre d'inventaire en particulier

Ce point terminal supprime le filtre d'inventaire spécifié.

```
GET /openapi/v1/filters/inventories/{inventory_filter_id}
```

Recherche de flux

La fonction de recherche de flux offre des fonctionnalités similaires à celles décrites dans la section Flux. Ces ensembles d'API nécessitent la capacité `flow_inventory_query` associée à la clé API.

Requête de dimensions de flux

Ce point terminal renvoie la liste des colonnes de flux dans lesquelles des critères de recherche (ou *filtres*) peuvent être spécifiés pour les requêtes de recherche de flux (ci-dessous). Pour en savoir plus sur les descriptions de colonnes, consultez Colonnes et filtres.

```
GET /openapi/v1/flowsearch/dimensions
```

Paramètres : Aucun

Objet de réponse :

Nom	Type	Description
dimensions	Liste de chaînes	La liste des dimensions téléchargées par l'utilisateur et l'orchestrateur.

Exemple de code Python

```
restclient.get('/flowsearch/dimensions')
```

Requête de mesures de flux

Ce point terminal renvoie la liste des mesures, par exemple le nombre d'octets et le nombre de paquets, associées aux observations de flux.

```
GET /openapi/v1/flowsearch/métriques
```

Paramètres : Aucun

Objet de réponse :

Nom	Type	Description
metrics	Liste de chaînes.	Liste des mesures disponibles.

Exemple de code Python

```
restclient.get('/flowsearch/metrics')
```

Requête de flux

Ce point terminal renvoie la liste des flux correspondant aux critères de filtre. Chaque objet de flux dans le résultat possède des attributs qui sont une union des dimensions de flux (renvoyées par l'API des dimensions de flux ci-dessus) ainsi que des mesures de flux (renvoyées par l'API des mesures de flux ci-dessus).

```
POST /openapi/v1/flowsearch
```

La liste des colonnes qui peuvent être spécifiées dans les critères de filtre peut être obtenue à l'aide de l'API `/openapi/v1/flowsearch/ dimensions`.

Paramètres : le corps de la requête se compose d'un corps JSON avec les clés suivantes.

Nom	Type	Description
t0	entier ou chaîne	Heure de début de la recherche de flux (heure d'origine ou ISO 8601)
t1	entier ou chaîne	Heure de fin de la recherche de flux (heure d'origine ou ISO 8601)
filter	JSON	Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à tous les flux.
scopeName	chaîne	Nom complet de la portée à laquelle la requête est limitée.
dimensions	tableau	(Facultatif) Liste des noms de dimensions à renvoyer dans le résultat de l'API flowsearch. Il s'agit d'un paramètre facultatif. Si ce paramètre n'est pas spécifié, les résultats de la recherche de flux renvoient toutes les dimensions disponibles. Cette option est utile pour spécifier un sous-ensemble des dimensions disponibles lorsque l'appelant ne se soucie pas du reste des dimensions.

Nom	Type	Description
metrics	tableau	(Facultatif) Liste des noms de mesures à renvoyer dans le résultat de l'API flowsearch. Il s'agit d'un paramètre facultatif. Si ce paramètre n'est pas spécifié, les résultats de la recherche de flux renvoient toutes les mesures disponibles. Cette option est utile pour spécifier un sous-ensemble de mesures disponibles lorsque l'appelant n'est pas intéressé par le reste des mesures.
limit	nombre entier	(Facultatif) Limite du nombre de flux de réponse.
offset	chaîne	(Facultatif) Objet décalage reçu de la réponse précédente.
descending	booléen	(Facultatif) Si ce paramètre est défini sur « faux » ou non défini, les résultats sont classés dans l'ordre ascendant des horodatages. Si la valeur du paramètre est « vrai », les résultats sont classés dans l'ordre décroissant des horodatages.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "dst_hostname",
                "value": "prod"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "scopeName": "Default:Production:Web",
    "limit": 100,
    "offset": <offset-object>
}
```

Filtres

Le filtre prend en charge les filtres primaires et les filtres logiques (« not », « and », « or ») composés d'un ou de plusieurs filtres primaires. Le format du filtre primaire est le suivant :

```
{"type" : "<OPERATOR>", "field": "<COLUMN_NAME>", "value": "<COLUMN_VALUE>"}
```

Pour les filtres primaires , l'opérateur peut être un opérateur de comparaison comme `eq, ne, lt, lte, gt` ou `gte`. L'operateur peut également être `in, regex, subnet, contains` ou `range`.

Voici quelques exemples de filtres primaires :

```
{"type": "eq", "field": "src_address", "value": "7.7.7.7"}

{"type": "regex", "field": "src_hostname", "value": "prod.*"}

{"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}

# Note, 'in' clause uses 'values' key instead of 'value'
{"type": "in", "field": "src_port", "values": [80, 443]}
```

Vous pouvez également spécifier des filtres complexes à l'aide d'opérations booléennes telles que not", "and" ou "or". Voici quelques exemples de ces types de filtres :

```
# "and" and "or" operators need to specify list of "filters"
{"type": "and",
    "filters": [
        {"type": "in", "field": "src_port", "values": [80, 443]},
        {"type": "regex", "field": "src_hostname", "value": "prod.*"}
    ]
}

# "not" operator needs to specify a "filter"
{"type": "not",
    "filter": {"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}
}
```

Plus formellement, le schéma du `filtre` dans la demande de recherche de flux est le suivant :

Clés	Valeurs
type	Type de filtre
Champ	Colonne du champ de filtre pour les filtres primaires
filter	Objet filtre (utilisé uniquement pour le type de filtre `not`)
filtres	Liste des objets filtre (utilisés pour les types de filtres `and` et `or`)
valeur	Valeur des filtres primaires
values (valeurs)	Liste de valeurs pour les filtres primaires avec le type de filtre `in` ou `range`

Types de filtres primaires

eq, ne : recherche dans les flux l'égalité ou l'inégalité, respectivement, dans la colonne spécifiée par « `field` » (champ) avec la valeur spécifiée par « `value` » (valeur). Prend en charge les champs suivants : `src_hostname, dst_hostname, src_address, dst_address, src_port, dst_port, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name, proto`. Ces opérateurs fonctionnent également sur les colonnes étiquetées par l'utilisateur.

lt, lte, gt, gte : recherche les flux où les valeurs de la colonne spécifiées par « `field` » sont inférieures, égales, supérieures ou égales à (selon le cas) la valeur spécifiée par « `value` ». Prend en charge les champs suivants : `[src_port, dst_port]`.

range(plage) : recherche les flux pour les valeurs de la colonne spécifiées par « field » entre le début et la fin de la plage spécifiée par la liste « values » (cette liste doit être de taille 2 pour le type de filtre *« range »* : la première valeur est le début de la plage et la deuxième est la fin de plage). Prend en charge les champs suivants : [src_port, dst_port].

in : recherche les flux pour les membres dans la colonne spécifiée par « `field` » dans la liste de membres spécifiée par « `values` ». Prend en charge les champs suivants : `src_hostname, dst_hostname, src_address, dst_address, src_port, dst_port, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name, proto`. Cet opérateur fonctionne également sur les colonnes étiquetées par l'utilisateur.

regex, contains : recherche dans les flux les correspondances d'expressions régulières ou les correspondances de stockage, respectivement, dans la colonne spécifiée par « `field` » avec l'expression régulière spécifiée par « `value` ». Prend en charge les champs suivants : `src_hostname, dst_hostname, src_scope_name, dst_scope_name, vrf_name, src_enforcement_epg_name, dst_enforcement_epg_name`. Ces opérateurs fonctionnent également sur les colonnes étiquetées par l'utilisateur. Les filtres de type `regex (expression régulière)` doivent utiliser des modèles d'expression régulière de style Java comme « `valeur` ».

subnet : recherche les flux pour les membres de sous-réseau spécifiés par « `field` » sous forme de chaîne en notation CIDR. Prend en charge les champs suivants : `["src_address", "dst_address"]`

Types de filtres logiques

- **not** : filtre « non » logique de l'objet spécifié par « `filtre` ».
- **and** : filtre « et » logique de la liste des objets de filtre spécifiées par « `filtres` ».
- **or** : filtre « ou » logique de la liste des objets de filtre spécifiés par « `filtres` ».

Objet de réponse :

Clés	Valeurs
offset	Décalage de réponse à transmettre pour la page de résultats suivante
results	Liste des résultats

Pour générer la page de résultats suivante, prenez l'objet reçu par la réponse dans « `offset` » et transmettez-le comme valeur pour le `décalage` de la prochaine requête.

Exemple de code Python

```
req_payload = {"t0": "2016-11-07T09:00:00-0700",
```

```
                "t1": "2016-11-07T19:00:00-0700",
                "scopeName": "Default:Prod:Web",
                "limit": 10,
                "filter": {"type": "and",
                    "filters": [
                     {"type": "subnet", "field": "src_address", "value": "1.1.11.0/24"},

                        {"type": "regex", "field": "src_hostname", "value": "web*"}
                     ]
                }
}

resp = restclient.post('/flowsearch', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Requête TopN pour les flux

Ce point terminal renvoie une liste des N valeurs triées les plus hautes d'une dimension spécifiée, où le rang dans la liste est déterminé par l'agrégation de la métrique spécifiée.

`POST /openapi/v1/flowsearch/topn`

Paramètres :

La liste des colonnes qui peuvent être spécifiées dans les critères de filtre peut être obtenue à l'aide de l'API `/openapi/v1/flowsearch/dimensions`. Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous. Les paramètres `t0` et `t1` du corps de la demande peuvent être au format heure d'origine ou au format ISO 8601. L'API TopN permet uniquement d'interroger une plage temporelle maximale d'un jour. La dimension selon laquelle le regroupement doit être effectué doit être précisée dans le champ `Dimension`. La mesure selon laquelle les N premiers résultats doivent être classés doit être spécifiée dans le champ de `mesure` dans le corps JSON. Vous devez spécifier un `seuil` avec une valeur minimale de 1, ce qui signifie le « N » de « TopN ». La valeur maximale de ce `seuil` est 1 000. Même si l'utilisateur spécifie plus de 1000, l'API ne renvoie qu'un maximum de 1000 résultats. En outre, vous devez spécifier un paramètre appelé `scopeName` qui correspond au nom complet de la portée à laquelle vous souhaitez restreindre la recherche. Le `filtre` est le même que celui du filtre de recherche de flux Filtres, on page 86 . Si le `filtre` n'est pas mentionné, topN est appliqué à toutes les entrées d flux.e

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "dimension": "src_address",
  "metric": "fwd_pkts",
 "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},  #optional

  "threshold": 5,
  "scopeName": "Default"
}
```

Le corps de la requête se compose d'un corps JSON avec les clés suivantes.

Clés	Valeurs
t0	Heure de début du flux (heure d'origine ou ISO 8601)
t1	Heure de fin du flux (heure d'origine ou ISO 8601)

Clés	Valeurs
filter	Filtre de requête. Si le filtre est vide (c.-à-d. {}) ou le filtre est absent (facultatif), la requête topN est appliquée à toutes les entrées de flux
scopeName	Nom complet de la portée à laquelle la requête est limitée
dimension	La dimension est un champ sur lequel nous regroupons.
metric	La mesure est le nombre total de valeurs de la dimension.
seuil	Le seuil est N dans le N top.

Objet de réponse :

Clés	Valeurs
résultat	Tableau des N principales entrées

Exemple de code Python

```
req_payload = {
    "t0": "2017-06-07T08:20:00-07:00",
    "t1": "2017-06-07T14:20:00-07:00",
    "dimension": "src_address",
    "metric": "fwd_pkts",
    "filter": {"type": "ne", "field": "src_address", "value": "172.29.203.193"},
    "threshold": 5,
    "scopeName": "Default"
}
resp = rc.post('/flowsearch/topn',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Exemple de réponse

```
[
  { "result": [
    {"src_address": "172.31.239.163", "fwd_pkts": 23104},
    {"src_address": "172.31.239.162", "fwd_pkts": 22410},
    {"src_address": "172.31.239.166", "fwd_pkts": 16185},
    {"src_address": "172.31.239.168", "fwd_pkts": 15197},
    {"src_address": "172.31.239.169", "fwd_pkts": 15116}
    ]
  }
]
```

Nombre de flux

Ce point terminal renvoie le nombre d'observations de flux correspondant aux critères spécifiés.

```
POST /openapi/v1/flowsearch/count
```

Paramètres :

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous. Les paramètres `t0` et `t1` du corps de la demande peuvent être au format heure d'origine ou au format ISO 8601. Cette API permet uniquement d'interroger une plage temporelle maximale d'un jour. En outre, vous devez spécifier le paramètre `scopeName` qui correspond au nom complet de la portée à laquelle vous souhaitez restreindre la recherche. Si ce paramètre n'est pas spécifié, la demande API de comptage des observations de flux s'applique à toutes les portées auxquelles vous avez un accès en lecture. Le `filtre` est identique à celui du filtre de recherche de flux Filtres (Filtres).

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},
  "scopeName": "Default"
}
```

Le corps de la requête se compose d'un corps JSON avec les clés suivantes.

Clés	Valeurs
t0	Heure de début du flux (heure d'origine ou ISO 8601)
t1	Heure de fin du flux (heure d'origine ou ISO 8601)
filter	Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à tous les flux.
scopeName	Nom complet de la portée à laquelle la requête est limitée

Objet de réponse :

Clés	Valeurs
Nombre	Le nombre d'observations de flux correspondant aux critères de recherche de flux.

Exemple de code Python

```
req_payload = {
      "t0": "2017-07-20T08:20:00-07:00",
      "t1": "2017-07-20T10:20:00-07:00",
      "scopeName": "Tetration",
      "filter": {
                  "type": "eq",
                  "field": "dst_port",
                  "value": "5642"
          }
}
resp = rc.post('/flowsearch/count',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Exemple de réponse

```
{"count":508767}
```

Inventaire

Les API de recherche d'inventaire fournissent des fonctionnalités similaires à celles décrites dans la section Recherche d'inventaire. Ces ensembles d'API nécessitent la capacité `flow_inventory_query` associée à la clé API.

Requête de dimensions d'inventaire

Ce point terminal renvoie la liste des colonnes d'inventaire sur lesquelles des critères de recherche (ou des *filtres*) peuvent être spécifiés pour réaliser les requêtes de recherche d'inventaire.

```
GET /openapi/v1/inventory/search/dimensions
```

Recherche dans l'inventaire

Ce point terminal renvoie la liste des éléments de l'inventaire correspondant aux critères spécifiés.

```
POST /openapi/v1/inventory/search
```

La liste des colonnes qui peuvent être spécifiées dans les critères de filtre peut être obtenue à l'aide de l'API /openapi/v1/inventory/ search/dimensions.

Paramètres :

Nom	Type	Description
filter	JSON	Une requête de filtre.
scopeName	chaîne	(Facultatif) Nom de la portée par laquelle les résultats doivent être limités.
limit	nombre entier	(facultatif) Nombre maximal de résultats à renvoyer.
offset	nombre entier	(facultatif) Décalage par rapport à la demande précédente pour obtenir la page suivante.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "filter": {
        "type": "contains",
        "field": "hostname",
        "value": "collector"
    },
    "scopeName": "Default:Production:Web",  // optional
    "limit": 100,
```

```
    "offset": "<offset-object>"              // optional
}
```

Pour connaître les différents types de filtres pris en charge, reportez-vous à Filtres, on page 86

Le corps de la requête se compose d'un corps JSON avec les clés suivantes.

Clés	Valeurs
filtrer	Filtre de requête. Si le filtre est vide (c.-à-d. {}), la requête correspond à tous les éléments de l'inventaire.
scopeName	Nom complet de la portée à laquelle la requête est limitée (facultatif)
dimensions	Liste des noms de dimensions à renvoyer dans le résultat de l'API de recherche d'inventaire. Il s'agit d'un paramètre facultatif. S'il n'est pas spécifié, les résultats renvoient toutes les dimensions disponibles. Cette option est utile pour spécifier un sous-ensemble des dimensions disponibles lorsque l'appelant ne se soucie pas du reste des dimensions.
limit	Limite du nombre d'éléments de réponse (facultatif)
offset	Objet décalage reçu de la réponse précédente (facultatif)

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Nom	Type	Description
offset	nombre entier	Décalage de réponse à transmettre pour la page de résultats suivante.
results	tableau d'objets	Liste des résultats.

La réponse peut contenir un champ `décalage` pour les réponses paginées. Les utilisateurs devront spécifier le même décalage dans la demande suivante pour obtenir la prochaine série de résultats.

Exemple de code Python

```
req_payload = {
  "scopeName": "Tetration", # optional
  "limit": 2,
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
  }
}

resp = restclient.post('/inventory/search', json_body=json.dumps(req_payload))
print resp.status_code
```

```
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Statistiques d'inventaire

Ce point terminal renvoie des statistiques sur les éléments d'inventaire.

```
GET /openapi/v1/inventory/{id}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Table 6:

Paramètres du chemin	Description
ID	ID de l'élément en d'inventaire comme `{ip}-{vrf_id}`, par exemple sur `1.1.1.1-123`

Paramètres de requête	Description
t0	Heure de début des statistiques en heure d'origine
t1	Heure de fin des statistiques en heure d'origine
td	Granularité pour les agrégations de statistiques. Un entier spécifie le nombre de secondes. Des chaînes peuvent être transmises, telles que « minute », « heure » et « jour ».

Exemple de code Python

```
resp = restclient.get('/inventory/1.1.1.1-123/stats?t0=1483228800&t1=1485907200&td=day')
```

Inventaire

Ce point terminal renvoie le nombre d'éléments de l'inventaire correspondant aux critères spécifiés.

```
POST /openapi/v1/inventory/count
```

La liste des colonnes qui peuvent être spécifiées dans les critères de filtre peut être obtenue à l'aide de l'API `/openapi/v1/inventory/ search/dimensions`.

Paramètres :

Nom	Type	Description
filter	JSON	Une requête de filtre.
scopeName	chaîne	(Facultatif) Nom de la portée par laquelle les résultats doivent être limités.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
{
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "hostname",
                "value": "prod"
            },
            {
                "type": "subnet",
                "field": "ip"
                "value": "6.6.6.0/24"
            }
        ]
    },
    "scopeName": "Default:Production:Web",  # optional
}
```

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Table 7:

Clés	Valeurs
Nombre	Nombre d'éléments de l'inventaire correspondant aux critères du filtre

Exemple de code Python

```
req_payload = {
  "scopeName": "Tetration", # optional
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
  }
}

resp = restclient.post('/inventory/count', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Vulnérabilité de l'inventaire

Ce point terminal renvoie les CVE correspondant aux adresses IP associées aux charges de travail vulnérables.

Cette API est uniquement disponible pour les utilisateurs disposant au minimum d'un accès en lecture à la portée racine.

```
POST /openapi/v1/inventory/cves/{rootScopeID}
```

Paramètres :

Nom	Type	Description
ips	liste de chaînes	Liste des adresses IP pour récupérer les informations CVE.

Le corps de la demande doit être une requête au format JSON. Un exemple de corps de requête est présenté ci-dessous.

```
    {
        "ips": [
            "10.18.187.72",
            "10.18.187.73"
        ]
    }
```

Réponse

La réponse est un tableau d'objets JSON contenu dans le corps du message, avec les propriétés suivantes.

Nom	Type	Description
ip	chaîne	Adresse IP
cve_ids	liste de chaînes	Liste des ID CVE dans l'inventaire avec l'adresse IP.

Exemple de code Python

```
  root_scope_id = "5fa0d242497d4f7d968c669b"
  req_payload = {
    "ips":["10.18.187.72", "10.18.187.73"]
  }

  resp = restclient.post('/inventory/cves/' + root_scope_id,
json_body=json.dumps(req_payload))
  print resp.status_code
  if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

Charge de travail

L'API de la charge de travail fournit un accès programmatique au contenu de la page Workload Profile *(Profil de la charge de travail)*. Cet ensemble d'API nécessite la fonctionnalité `sensor_management ou flow_inventory_query` associées à la clé API.

Détails de la charge de travail

Ce point terminal renvoie la charge de travail spécifique étant donné une UUID de l'agent.

```
GET /openapi/v1/workload/{uuid}
```

Paramètres du chemin	Description
UUID	UUID de l'agent

Réponse

La réponse est un objet de charge de travail associé à l'UUID spécifié. Le schéma des attributs de l'objet de charge de travail est décrit ci-dessous :

Table 8:

Attribut	Type	Description
agent_type	nombre entier	Type d'agent dans l'énumération
agent_type_str	chaîne	Type d'agent en texte brut
auto_upgrade_opt_out	booléen	Si la valeur est « vrai », les agents ne sont pas mis à niveau automatiquement lors de la mise à niveau de la grappe.
cpu_quota_mode	nombre entier	Contrôle de quota du processeur
cpu_quota_us	nombre entier	Utilisation du processeur
current_sw_version	chaîne	Version du logiciel agent exécutée sur la charge de travail
data_plane_disabled	booléen	Si la valeur est « vrai », les données de télémétrie de flux ne sont pas exportées de l'agent vers la grappe
desired_sw_version	chaîne	La version du logiciel agent destinée à être exécutée sur la charge de travail
enable_conversation_flows	booléen	Si la valeur est True (vraie), le mode conversation est activé.
enable_cache_sidechannel	booléen	Si la valeur est True (vraie), la détection des attaques par canal auxiliaire est activée
enable_forensics	booléen	Si la valeur est True (vraie), l'investigation criminalistique est activée
enable_meltdown	booléen	Si la valeur est True (vraie), la détection d'exploit de fusion est activée
enable_pid_lookup	booléen	Si la valeur est True (vraie), la recherche de processus est activée

Attribut	Type	Description
forensics_cpu_quota_mode	nombre entier	Contrôle du quota du processeur pour la criminalistique
forensics_cpu_quota_us	nombre entier	Utilisation des quotas de criminalistique
criminalistique_mem_quota_octets	nombre entier	Quota de mémoire destiné à la criminalistique en octets
host_name	chaîne	Nom d'hôte sur la charge de travail
interfaces	tableau	Tableau d'objets Interface
kernel_version	chaîne	Version du noyau
last_config_fetch_at	nombre entier	Dernière configuration récupérée le
last_software_update_at	nombre entier	Horodatage auquel l'agent a signalé sa version actuelle.
max_rss_limit	nombre entier	Limite de mémoire maximale
intégrée	chaîne	Plateforme de la charge de travail
UUID	chaîne	Identifiant unique de l'agent
windows_enforcement_mode	chaîne	Type de mode d'application Windows, WAF (Windows Advanced Firewall, pare-feu avancé Windows) ou WFP (Windows Filtering Platform, plateforme de filtrage Windows)

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s' % (agent_uuid))
```

Statistiques de la charge de travail

Ce point terminal renvoie des statistiques pour une charge de travail.

```
GET /openapi/v1/workload/{uuid}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Paramètres du chemin	Description
UUID	UUID de l'agent

L'URL de la requête contient les paramètres suivants :

Paramètres de requête	Description
t0	Heure de début des statistiques en heure d'origine
t1	Heure de fin des statistiques en heure d'origine L'heure de fin ne peut pas dépasser l'heure de début de plus d'un jour.
td	Granularité pour les agrégations de statistiques. Un entier spécifie le nombre de secondes. Des chaînes peuvent être transmises, telles que « minute », « heure » et « jour ».

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Nom	Type	Description
Horodatage	chaîne	Heure à laquelle les mesures ont été recueillies (heure d'origine ou ISO 8601)
results	objet	Indicateurs

Les mesures sont un objet JSON avec les propriétés suivantes :

Nom	Type	Description
flow_count	nombre entier	Nombre de flux.
rx_byte_count	nombre entier	Le nombre d'octets reçus.
rx_packet_count	nombre entier	Nombre de paquets reçus
tx_byte_count	nombre entier	Nombre d'octets transmis.
tx_packet_count	nombre entier	Nombre de paquets transmis.

Exemple de code Python

```
 agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
 td = 15 * 60 # 15 minutes
 resp = restclient.get('/workload/%s/stats?t0=1483228800&t1=1485907200&td=%d' % (agent_uuid,
td))

 # This code queries workload statistics for a week
 t0 = 1483228800
 for _ in range(7):
   t1 = t0 + 24 * 60 * 60
   resp = restclient.get('/workload/%s/stats?t0=%d&t1=%d&td=day' % (agent_uuid, t0, t1))
   t0 = t1
```

Paquets logiciels installés

Ce point terminal renvoie la liste des paquets logiciels installés sur les charges de travail.

```
GET /openapi/v1/workload/{uuid}/packages
```

Paramètres du chemin	Description
UUID	UUID de l'agent

Réponse

La réponse est un tableau d'objets JSON paquets logiciels. Le schéma de l'objet paquet est décrit ci-dessous :

Attribut	Type	Description
architecture	chaîne	Architecture du paquet
name	chaîne	Nom du paquet
publisher	chaîne	Serveur de publication du paquet
version	chaîne	Version du paquet

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/packages' % (agent_uuid))
```

Vulnérabilités de la charge de travail

Ce point terminal renvoie la liste des vulnérabilités observées sur la charge de travail.

```
GET /openapi/v1/workload/{uuid}/vulnerabilities
```

L'objet de vulnérabilités se compose d'un corps JSON avec les clés suivantes.

Paramètres du chemin	Description
UUID	UUID de l'agent

Réponse

La réponse est un tableau d'objets JSON de vulnérabilité. Le schéma de l'objet de vulnérabilité est décrit ci-dessous :

Attribut	Type	Description
cve_id	chaîne	ID d'exposition à la vulnérabilité commune
package_infos	tableau	Tableau d'objets Renseignements sur le paquet
v2_score	flottant	Note CVSS V2

Attribut	Type	Description
v2_access_complexity	chaîne	Complexité d'accès CVSS V2
v2_access_vector	chaîne	vecteur d’accès CVSS V2
v2_authentication	chaîne	Authentification CVSS V2
v2_availability_impact	chaîne	Incidence sur la disponibilité de CVSS V2
v2_confidentiality_impact	chaîne	Incidence sur la confidentialité de CVSS V2
v2_integrity_impact	chaîne	Incidence sur l’intégrité de CVSS V2
v2_severity	chaîne	Gravité de CVSS V2
v3_score	flottant	Note CVSS V3
v3_attack_complexity	chaîne	Complexité des attaques CVSS V3
v3_attack_vector	chaîne	Vecteur d’attaque CVSS V3
v3_availability_impact	chaîne	Incidence sur la disponibilité de CVSS V3
v3_base_severity	chaîne	Gravité de base CVSS V3
v3_confidentiality_impact	chaîne	Incidence sur la confidentialité de CVSS V2
v3_integrity_impact	chaîne	Incidence sur l’intégrité de CVSS V3
v3_privileges_required	chaîne	Privilèges CVSS V3 requis
v3_scope	chaîne	Portée de CVSS V3
v3_user_interaction	chaîne	Interaction avec l’utilisateur CVSS V3

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/vulnerabilities' % (agent_uuid))
```

Processus de longue durée de la charge de travail

Ce point terminal renvoie la liste des processus de longue durée sur la charge de travail. Les processus de longue durée sont définis comme des processus qui ont un temps de disponibilité d’au moins 5 minutes.

```
GET /openapi/v1/workload/{uuid}/process/list
```

Paramètres du chemin	Description
UUID	UUID de l'agent

Réponse

La réponse est une liste de processus d'objets JSON.

Attribut	Type	Description
CMD	chaîne	Chaîne de commande du processus
binary_hash	chaîne	SHA256 du binaire de processus en hexadécimal
ctime	long	ctime du processus binaire en nous
mtime	long	mtime du processus binaire en nous
exec_path	chaîne	Chemin d'accès de l'exécutable du processus
exit_usec	long	Heure de sortie du processus
num_libs	nombre entier	Nombre de bibliothèques chargées par le processus
pid	nombre entier	Process ID
ppid	nombre entier	ID du processus parent
pkg_info_name	chaîne	Nom du paquet associé au processus
pkg_info_version	chaîne	Version du paquet associé au processus
proc_state	chaîne	État du processus
disponibilité	long	Disponibilité du processus en nous
username	chaîne	Nom d'utilisateur du processus
resource_usage	tableau objet	Tableau d' Utilisation des ressources

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/openapi/v1/workload/%s/process/list' % (agent_uuid))
```

Résumé de l'instantané du processus de charge de travail

Ce point terminal renvoie un résumé d'instantané de processus pour cette charge de travail. Un instantané de processus contient tous les processus capturés par la charge de travail à un moment donné. Actuellement, une copie du dernier instantané de processus est conservée. Le point terminal prend en charge la méthode POST avec une charge utile vide pour faciliter une extension future.

```
POST /openapi/v1/workload/{uuid}/process/tree/ids
```

Paramètres du chemin	Description
UUID	UUID de l'agent

Réponse

La réponse est une liste d'objets JSON récapitulatifs d'instantané de processus.

Attribut	Type	Description
sensor_uuid	chaîne	UUID de l'agent
handle	chaîne	Descripteur de l'instantané de processus à récupérer
process_count	nombre entier	Nombre de processus dans l'instantané
ts_usec	nombre entier	Horodatage de capture de l'instantané

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
                       agent_uuid, json_body=json.dumps(payload))
```

Instantané du processus de charge de travail

Ce point terminal renvoie un instantané de processus sur cette charge de travail. Un instantané de processus contient tous les processus capturés par la charge de travail à un moment donné. Actuellement, une copie du dernier instantané de processus est conservée. Ce point terminal doit être utilisé avec le point de terminaison de résumé d'instantané du processus de charge de travail.

```
POST /openapi/v1/workload/{uuid}/process/tree/details
```

Paramètres du chemin	Description
UUID	UUID de l'agent

Champ de charge utile	Type	Description
handle	chaîne	Descripteur de l'instantané de processus à récupérer

Réponse

La réponse est une liste de processus appartenant à l'instantané en JSON.

Attribut	Type	Description
command_string	chaîne	Chaîne de commande marquée d'un jeton
command_string_raw	chaîne	Chaîne de commande brute
binary_hash	chaîne	SHA256 du binaire de processus en hexadécimal
ctime	long	ctime du processus binaire en nous
mtime	long	mtime du processus binaire en nous
exec_path	chaîne	Chemin d'accès de l'exécutable du processus
Process-id	nombre entier	Process ID
parent_process_id	nombre entier	ID du processus parent
process_key	nombre entier	Clé unique du processus
parent_process_key	nombre entier	Clé unique pour le processus parent
pkg_info_name	chaîne	Nom du paquet associé au processus
pkg_info_version	chaîne	Version du paquet associé au processus
proc_state	chaîne	État du processus
disponibilité	long	Disponibilité du processus en nous
username	chaîne	Nom d'utilisateur du processus
cve_ids	tableau	Tableau d'objets CVEID

Exemple de code Python

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
            agent_uuid, json_body=json.dumps(payload))
handle = json.loads(resp.text)['process_summary'][0]['summary'][0]['handle']
```

```
payload = {
  "handle": handle,
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/details' %
            agent_uuid, json_body=json.dumps(payload))
```

Définitions d'objets JSON

Interface

Attribut	Type	Description
ip	chaîne	L'adresse IP de l'interface.
MAC	chaîne	L'adresse MAC de l'interface.
name	chaîne	Le nom de l'interface:
masque réseau	chaîne	Masque réseau de l'interface
pcap_opened	booléen	Si la valeur est False, les captures de paquets ne sont pas activées pour l'interface
tags_scope_id	tableau	ID de la portée associés à l'interface
VRF	chaîne	Nom VRF
vrf_id	nombre entier	ID VRF

Renseignements sur le paquet

Attribut	Type	Description
name	chaîne	Nom du paquet
version	chaîne	Version du paquet

Utilisation des ressources

Attribut	Type	Description
cpu_usage	flottant	Utilisation du processeur
memory_usage_kb	nombre entier	Utilisation de la mémoire
ts_usec	long	Horodatage de l'heure à laquelle l'utilisation de la ressource a été capturée.

ID CVE

Attribut	Type	Description
cve_id	chaîne	ID CVE
impact__cvss_v2__access_complexity	chaîne	complexité de l'accès CVE
impact__cvss_v2__access_vector	chaîne	vecteur d'accès CVE

Configuration de génération de politiques par défaut

Cet ensemble d'API est utilisé pour lire et mettre à jour la configuration de génération de politiques par défaut pour une portée racine.

Les API nécessitent la capacité `app_policy_management` associée à la clé API.

Remarque Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

- Objet configuration de génération de politiques, à la page 105
- Obtenir la configuration de génération de politiques par défaut, à la page 107
- Définir la configuration de génération de politiques par défaut, à la page 107

Objet configuration de génération de politiques

Attribut	Type	Description
carry_over_policies	booléen	Toute politique marquée comme approuvée sera maintenue, si possible
deep_policy_generation	booléen	crée des politiques pour l'ensemble de l'arborescence de la portée sous la portée donnée, y compris tous les membres de la portée donnée
skip_clustering	booléen	défini à vrai pour ignorer la mise en grappe, génère des politiques avec des grappes et des portées approuvées existantes
auto_accept_policy_connectors	booléen	accepte automatiquement tous les connecteurs de politique sortants
enable_exclusion_filter	booléen	appliquer des filtres d'exclusion aux données de flux d'entrée

Attribut	Type	Description
enable_default_exclusion_filter	booléen	appliquer des filtres d'exclusion par défaut aux données de flux d'entrée
remove_redundant_policies	booléen	supprimer les politiques redondantes lors de la génération approfondie des politiques
enable_service_discovery	booléen	le paramètre faux permet d'ignorer la génération de politiques basées sur la plage de ports éphémères dans le pipeline adm rapporté par le capteur, actuellement utilisé pour générer des politiques pour Windows Active Directory.
externals	tableau	liste ordonnée des objets de dépendance externes
clustering_granularity	chaîne	un choix parmi VERY_COARSE (TRÈS GROSSIÈRE), COARSE (GROSSIÈRE), MEDUM (MOYENNE), FINE (FINE), VERY_FINE (TRÈS FINE)
policy_compression	chaîne	un choix parmi : DISABLED (DÉSACTIVÉE), CONSERVATIVE (CONSERVATRICE), MODERATE (MODÉRÉE), AGRESSIVE (AGRESSIVE), VERY_AGGRESSIVE (TRÈS AGRESSIVE)
port_generalization	chaîne	un choix parmi : DISABLED (DÉSACTIVÉE), CONSERVATIVE (CONSERVATRICE), MODERATE (MODÉRÉE), AGRESSIVE (AGRESSIVE), VERY_AGGRESSIVE (TRÈS AGRESSIVE)
sim_policy	nombre entier	1 => flux, 2 => processus, 5 => les deux

L'objet de dépendance externe

Nom	Type	Description
ID	chaîne	ID du filtre
filter_type	chaîne	AppScope ou UserInventoryFilter
inclure	tableau	objet avec user_filters booléen pour activer et user_filter_list pour la liste ordonnée des filtres d'inventaire de service fournis

Obtenir la configuration de génération de politiques par défaut

Ce point terminal renvoie la configuration par défaut actuelle de la génération de politiques. Peut renvoyer un objet vide si aucun objet n'a été créé.

```
GET /openapi/v1/app_scopes/default_adm_run_config
```

Paramètres :

L'URL de la demande contient les paramètres suivants

Nom	Type	Description
root_app_scope_id	chaîne	Identificateur unique de la portée racine à laquelle cette configuration par défaut s'applique

Objet de réponse : renvoie la configuration de génération de politique par défaut actuelle ou un objet vide si aucune configuration n'a été créée.

Définir la configuration de génération de politiques par défaut

Ce point terminal définit la configuration de génération de politiques par défaut

```
PUT /openapi/v1/app_scopes/default_adm_run_config
```

Paramètres en plus des valeurs de la liste d'objets de configuration de génération de politique ci-dessus.

Nom	Type	Description
root_app_scope_id	chaîne	Identificateur unique de la portée racine à laquelle cette configuration par défaut s'applique

Objet de la réponse : renvoie la configuration de génération de politiques par défaut.

Intent criminalistique

Les API des agents logiciels sont associées à la gestion des intents criminalistiques.

Les intents criminalistiques lient un profil criminalistique au groupe d'agents auquel il s'applique. Le groupe d'agents est défini à l'aide d'un filtre d'inventaire.

Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API.

Remarque Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

Objet intent criminalistique

Attribut	Type	Description
ID	chaîne	identifiant unique de l'intent
name	chaîne	nom de l'intent
inventory_filter_id	tableau	ID du filtre d'inventaire associé à l'intent
forensic_config_profile_id	nombre entier	ID du profil associé à cet intent
created_at	nombre entier	Horodatage Unix de la création de l'intent
updated_at	nombre entier	Horodatage Unix de la dernière mise à jour de l'intent

Liste des intents criminalistiques

Ce point terminal répertorie tous les profils criminalistiques existants

```
GET /openapi/v1/inventory_config/forensic_intents
```

Paramètres : Aucun

Ce point terminal renvoie un tableau de résumés d'objets intents criminalistiques

Récupération d'un intent criminalistique unique

```
GET /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Paramètres :

Nom	Type	Description
intent_id	chaîne	ID de l'intent

Renvoie une représentation détaillée de l'objet intent criminalistique.

Création d'un intent criminalistique

```
POST /openapi/v1/inventory_config/forensic_intents
```

Paramètres :

Nom	Type	Description
name	chaîne	nom de l'intent
inventory_filter_id	chaîne	ID du filtre d'inventaire associé à l'intent

Nom	Type	Description
forensic_config_profile_id	chaîne	ID du profil criminalistique associé à l'intent

Renvoie un objet intent criminalistique.

Mettre à jour un intent criminalistique

```
PUT /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Paramètres :

Nom	Type	Description
intent_id	chaîne	ID de l'intent
name	chaîne	nom de l'intent
inventory_filter_id	chaîne	ID du filtre d'inventaire associé à l'intent
forensic_config_profile_id	chaîne	ID du profil criminalistique associé à l'intent

Renvoie un objet intent criminalistique.

Supprimer un intent criminalistique

```
DELETE /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Paramètres :

Nom	Type	Description
intent_id	chaîne	ID de l'intent

Retourne 200 en cas de réussite.

Ordres des intents criminalistiques

Les API des agents logiciels sont associées à la gestion de l'ordre des intents criminalistiques.

Les profils criminalistiques sont appliqués aux agents à l'aide d'intents. Les intents utilisent des filtres d'inventaire pour définir les groupes d'agents. Si les filtres se chevauchent, nous devons savoir lesquels appliquer. Nous utilisons un ordre pour définir la priorité des intents.

Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API.

Remarque Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

- Objet ordre d'intent criminalistique, à la page 110
- Récupérer l'ordre actuel des intents criminalistiques, à la page 110
- Création d'un intent criminalistique, à la page 108

Objet ordre d'intent criminalistique

Attribut	Type	Description
version	chaîne	version de l'ordre actuel
intents	tableau	nom des objets d'intent dans l'ordre
intent_ids	tableau	tableau d'ID d'intents criminalistiques

Récupérer l'ordre actuel des intents criminalistiques

Ce point terminal renvoie l'ordre actuel des intents criminalistiques.

```
GET /openapi/v1/inventory_config/forensic_orders
```

Paramètres : Aucun

Ce point terminal renvoie l'objet ordre actuel des intents criminalistiques.

Création d'un ordre d'intent criminalistique

```
POST /openapi/v1/inventory_config/forensic_orders
```

Paramètres :

Nom	Type	Description
version	chaîne	doit correspondre à la commande en cours
intent_ids	tableau	tableau d'ID d'intent

Renvoie un objet ordre d'intent criminalistique.

Profils criminalistiques

Les API des agents logiciels sont associées à la gestion des profils criminalistiques.

Les profils criminalistiques sont des ensembles de règles qui peuvent être appliquées aux groupes d'agents utilisant les intents criminalistiques.

Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API.

Remarque Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

Objet profil criminalistique

Attribut	Type	Description
ID	chaîne	identifiant unique du profil
name	chaîne	nom du profil
forensic_rules	tableau	tableau des règles associées au profil
created_at	nombre entier	Horodatage Unix de la création du profil
updated_at	nombre entier	Horodatage Unix de la dernière mise à jour du profil
is_readonly	booléen	indique si le profil est en lecture seule
root_app_scope_id	chaîne	ID de la portée racine à laquelle le profil appartient

Répertorier les profils criminalistiques

Ce point terminal répertorie tous les profils criminalistiques existants

```
GET /openapi/v1/inventory_config/forensic_profiles
```

Paramètres : Aucun

Ce point terminal renvoie un tableau de résumés d'objets profils criminalistiques.

Récupération d'un seul profil criminalistique

```
GET /openapi/v1/inventory_config/forensic_profiles/{profile_id}
```

Paramètres :

Nom	Type	Description
profile_id	chaîne	ID du profil

Renvoie une représentation détaillée de l'objet profil criminalistique.

Création d'un profil criminalistique

```
POST /openapi/v1/inventory_config/forensic_profiles
```

Paramètres :

Nom	Type	Description
name	chaîne	nom du profil
root_app_scope_id	chaîne	ID de la portée racine à laquelle le profil appartient
forensic_rule_ids	tableau	Tableau d'ID de règles criminalistiques à associer à ce profil

Renvoie un objet de profil criminalistique.

Mettre à jour un profil criminalistique

```
PUT /openapi/v1/inventory_config/forensic_profiles/{id}
```

Paramètres :

Nom	Type	Description
profile_id	chaîne	ID du profil
name	chaîne	nom du profil
forensic_rule_ids	tableau	Tableau d'ID de règles criminalistiques à associer à ce profil

Renvoie un objet de profil criminalistique.

Supprimer un profil criminalistique

```
DELETE /openapi/v1/inventory_config/forensic_profiles/{profile_id}
```

Paramètres :

Nom	Type	Description
profile_id	chaîne	ID du profil

Retourne 200 en cas de réussite.

Règles criminalistiques

Les API des agents logiciels sont associées à la gestion des règles criminalistiques.

Les règles criminalistiques sont utilisées dans les profils criminalistiques qui sont ensuite appliqués à des groupes d'agents.

Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API.

Remarque Ces API ne sont disponibles que pour les administrateurs de site et les propriétaires de portées racine.

Objet règle criminalistique

Attribut	Type	Description
ID	chaîne	identifiant unique de la règle
name	chaîne	nom de la règle
description	chaîne	description de la règle
type	chaîne	PREDEFINED (PRÉDÉFINI) ou USER_DEFINED (UTILISATEUR DÉFINI)
eval_group_type	chaîne	AS_INDIVIDUAL (COMME_INDIVIDU) ou AS_GROUP (COMME_GROUPE)

Attribut	Type	Description
gravité	chaîne	l'un des éléments suivants : IMMEDIATE_ACTION, CRITICAL, HIGH, MEDIUM, LOW (ACTION_IMMÉDIATE, CRITIQUE, ÉLEVÉE, MOYENNE ou FAIBLE)
Actions	tableau	Tableau ou chaînes ALERT (ALERTE) ou REPORT (RAPPORT)
created_at	nombre entier	Horodatage Unix de la création de la règle
updated_at	nombre entier	Horodatage Unix de la dernière mise à jour de la règle

Liste des règles criminalistiques

Ce point terminal répertorie toutes les règles criminalistiques existantes

```
GET /openapi/v1/inventory_config/forensic_rules
```

Paramètres : Aucun

Ce point terminal renvoie un tableau de résumés d'objets de règles criminalistiques.

Récupération d'une seule règle criminalistique

```
GET /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Paramètres :

Nom	Type	Description
rule_id	chaîne	ID de la règle

Renvoie une représentation détaillée de l'objet règle criminalistique.

Création d'une règle criminalistique

```
POST /openapi/v1/inventory_config/forensic_rules
```

Paramètres :

Nom	Type	Description
root_app_scope_id	chaîne	ID de la portée racine à laquelle cette règle appartient
name	chaîne	nom de la règle

Nom	Type	Description
description	chaîne	description de la règle
eval_group_type	chaîne	type de règle
gravité	chaîne	gravité de la règle
Actions	tableau	Tableau ou chaînes ALERT (ALERTE) ou REPORT (RAPPORT)
clause	chaîne	la clause de la requête de la règle.

Renvoie un objet de règle criminalistique.

Mettre à jour une règle criminalistique

```
PUT /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Paramètres :

Nom	Type	Description
rule_id	chaîne	ID de la règle
name	chaîne	nom de la règle
description	chaîne	description de la règle
eval_group_type	chaîne	type de règle
gravité	chaîne	gravité de la règle
Actions	tableau	Tableau ou chaînes ALERT (ALERTE) ou REPORT (RAPPORT)
clause	chaîne	la clause de la requête de la règle.

Renvoie un objet de règle criminalistique.

Supprimer une règle criminalistique

```
DELETE /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Paramètres :

Nom	Type	Description
rule_id	chaîne	ID de la règle

Retourne 200 en cas de réussite.

Paramètres de la plateforme

Cet ensemble d'API peut être utilisé pour ajouter, modifier ou supprimer des paramètres de plateforme et nécessite la fonctionnalité appliance_management associée à la clé API.

Note Ces API sont uniquement disponibles pour le service d'assistance à la clientèle et les administrateurs de site.

Obtenir des certificats

Ce point terminal est utilisé pour récupérer les certificats SSL/TLS.

```
GET /openapi/v1/platform_settings/outbound_http
```

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
nom_paire_clés	chaîne	Précisez le nom de la paire de clés.
cert_sha1	chaîne	L'identifiant unique du certificat.
created_at	chaîne	Récupérer les certificats créés à partir d'une date et d'une heure précises.

Obtenir les paramètres d'analyse de l'utilisation

Ce point terminal est utilisé pour récupérer les paramètres liés à l'analyse de l'utilisation ou à la collecte de données de télémétrie dans la plateforme.

```
GET /openapi/v1/platform_settings/ usage_analytics
```

Paramètres :

Nom	Type	Description
query	objet	Récupérer les paramètres liés à l'analyse de l'utilisation ou à la collecte de données de télémétrie.

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
usage_analytics_enabled	booléen	Indiquer si l'analyse de l'utilisation ou la collecte de données de télémétrie est activée ou désactivée.

Obtenir les message de connexion

Ce point terminal est utilisé pour récupérer le message de connexion personnalisé qui s'affiche pour les utilisateurs sur la plateforme.

```
GET /openapi/v1/platfor_settings/ login_message
```

Il n'y a aucun paramètre pour le point terminal get log message (récupération du message de connexion).

Obtenir les paramètres HTTP sortants

Ce point terminal est utilisé pour récupérer les paramètres HTTP sortants actuels configurés pour la plateforme.

```
GET /openapi/v1/platform_settings/outbound_http
```

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
outbound_http_enabled	chaîne	Indiquer si les connexions HTTP sortantes sont activées.

Mettre à jour les paramètres HTTP sortants

Ce point terminal est utilisé pour récupérer les paramètres HTTP sortants actuels configurés pour la plateforme.

```
POST /openapi/v1/ platform_settings/ outbound_http
```

Paramètres :

Nom	Type	Description
query	objet	Mettez à jour les paramètres HTTP sortants configurés pour la plateforme.

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
`outbound_http_enabled`	chaîne	Indiquer si les connexions HTTP sortantes sont activées.

Tester les paramètres HTTP sortants

Ce point terminal est utilisé pour tester les paramètres HTTP sortants actuellement configurés pour la plateforme.

```
POST openapi/v1/platform_settings/outbound_http_test
```

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
`opération réussie`	booléen	Vérifiez si la connexion HTTP sortante est réussie.

Obtenir les paramètres de serveur mandataire HTTP sortants

Ce point terminal est utilisé pour récupérer les paramètres du serveur mandataire HTTP sortants configurés pour la plateforme.

```
GET /openapi/v1/platform_settings/outbound_http_proxy
```

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
outbound_http_enabled	booléen	Indiquez si la connexion HTTP sortante est activée.
http_proxy_enabled	booléen	Indiquer si un serveur mandataire HTTP est activé.
http_proxy_server	chaîne	Récupérer le serveur mandataire HTTP sortant configuré.
http_proxy_port	nombre entier	Précisez le numéro de port utilisé pour le serveur mandataire HTTP.
http_proxy_username	chaîne	Précisez le nom d'utilisateur utilisé pour l'authentification auprès du serveur mandataire HTTP.
http_proxy_password_present	booléen	Déterminez si la plateforme est actuellement configurée pour utiliser un mot de passe pour l'authentification auprès du serveur mandataire HTTP.

Mettre à jour les paramètres de serveur mandataire HTTP sortant

Ce point terminal est utilisé pour récupérer les paramètres du serveur mandataire HTTP sortants configurés pour la plateforme.

```
POST /openapi/v1/ platform_settings/ outbound_http_proxy
```

La réponse est un objet JSON comportant le schéma suivant :

Clé	Type	Valeur
outbound_http_enabled	booléen	Indiquez si la connexion HTTP sortante est activée.

Clé	Type	Valeur
http_proxy_enabled	booléen	Indiquer si un serveur mandataire HTTP est activé.
http_proxy_server	chaîne	Récupérer le serveur mandataire HTTP sortant configuré.
http_proxy_port	nombre entier	Précisez le numéro de port utilisé pour le serveur mandataire HTTP.
http_proxy_username	chaîne	Précisez le nom d'utilisateur utilisé pour l'authentification auprès du serveur mandataire HTTP.
http_proxy_password_present	booléen	Déterminez si le mot de passe est disponible.

Exécution

L'application des politiques est la fonctionnalité par laquelle les politiques générées sont envoyées vers les ressources de la portée associée à un espace de travail et de nouvelles règles de pare-feu sont écrites. Cet ensemble d'API nécessite la capacité `app_policy_management` associée à la clé API.

Pour en savoir plus, consultez la section Appliquer les politiques.

Configuration de politique de réseau de l'agent

Ce point terminal renvoie un objet Agent en fonction de l'ID d'agent. Ceci est utile pour récupérer la politique de réseau, la configuration de l'agent, sa version, etc.

```
GET /openapi/v1/enforcement/agents/{aid}/network_policy_config
```

Paramètres :

L'URL de la demande contient les paramètres suivants

Nom	Type	Description
aid	chaîne	UUID de l'agent pour la configuration de la politique de réseau

Le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
include_filter_names	booléen	Inclut les noms et les ID de filtres dans les politiques de réseau.

Nom	Type	Description
inject_versions	booléen	Inclut les versions des espaces de travail ADM dans les politiques de réseau.

Réponse

La réponse de ce point terminal est un objet Agent .

Statistiques sur la politique concrète

Ce point terminal renvoie les statistiques pour les politiques concrètes en fonction de l'ID d'agent et de l'ID de politique concrète. Le point terminal renvoie un tableau d'objets Série chronologique du résultat de la politique concrète.

```
GET /openapi/v1/enforcement/agents/{aid}/concrete_policies/{cid}/stats?t0=<t0>&t1=<t1>
↪&td=<td>
```

Paramètres :

L'URL de la demande contient les paramètres suivants

Nom	Type	Description
aid	chaîne	UUID de l'agent pour les statistiques.
cid	chaîne	Identifiant unique UUID de politique concrète pour les statistiques.

Le corps de la requête JSON contient les clés suivantes :

Table 9:

Nom	Type	Description
t0	nombre entier	Heure de début des statistiques en heure d'origine
t1	nombre entier	Heure de fin des statistiques en heure d'origine
td	nombre entier ou chaîne	Granularité pour les agrégations de statistiques. Un entier spécifie le nombre de secondes. Des chaînes peuvent être transmises, telles que « minute », « heure » et « jour ».

Définitions d'objets JSON

Agent

Attribut	Type	Description
agent_uuid	chaîne	UUID de l'agent.
agent_config	objet	Configuration de l'agent
agent_config_status	objet	État de configuration de l'agent
desired_network_policy_config	objet	Configuration de la politique de réseau
provisioned_network_policy_config	objet	Configuration de politique de réseau mise en service
provisioned_state_update_timestamp	nombre entier	horodatage d'origine en secondes de la prise en compte par l'agent de la politique mise en œuvre ci-dessus.
desired_policy_update_timestamp	nombre entier	horodatage d'origine en secondes de la génération de desired_network_policy_config.
agent_info	objet	Renseignements sur l'agent
skipped	booléen	Vrai (vrai) lorsque la génération de règles concrètes est ignorée.
message	chaîne	Raison pour laquelle la génération de politiques concrètes est ignorée

Configuration de l'agent

Attribut	Type	Description
agent_uuid	chaîne	UUID de l'agent.
enforcement_enabled	booléen	La configuration indiquant que l'application est activée sur l'agent.
fail_mode	chaîne	Mode échec.
version	nombre	Numéro de version de la configuration de l'agent.
control_tet_rules_only	booléen	Configuration des règles de contrôle seulement

Attribut	Type	Description
allow_broadcast	booléen	Autoriser la configuration de diffusion
allow_multicast	booléen	Autoriser la configuration de multidiffusion
allow_link_local	booléen	Autoriser le lien Configuration locale.
forcement_cpu_quota_mode	chaîne	Mode de quota de CPU de l'agent d'application.
enforcement_cpu_quota_us	chaîne	Quota de CPU de l'agent d'application en micro-secondes.
forcement_max_rss_limit	nombre	Limite RSS max. d'agents d'application

Configuration de la politique de réseau

Attribut	Type	Description
version	chaîne	Numéro de version
network_policy	tableau	Tableau d'objets de la Politique réseau
address_sets	tableau	Tableau d'objets Ensemble d'adresses pour la fonction d'ensemble IP.
container_network_policy	tableau	Tableau d'objets ContainerNetworkPolicy

Politique réseau

Attribut	Type	Description
priority	chaîne	Priorité d'une politique concrète
enforcement_intent_id	chaîne	ID d'intent d'application.
concrete_policy_id	chaîne	Identifiant de politique concrète
match	objet	Mettre en correspondance des critères de la politique Ce champ est obsolète.
action	objet	Action pour la correspondance de politique.

Attribut	Type	Description
workspace_id	chaîne	ID de l'espace de travail.
adm_data_set_id	chaîne	ID de l'ensemble de données de découverte automatique des politiques de l'espace de travail
adm_data_set_version	chaîne	la version de l'ensemble de données de découverte automatique des politiques de l'espace de travail. Défini uniquement lorsque inject_versions=vrai est transmis dans les paramètres
cluster_edge_id	chaîne	ID de périphérie de la grappe.
policy_intent_group_id	chaîne	ID de groupe d'intents de politique
match_set	objet	Objet d'Ensemble de correspondances pour la prise en charge d'ensembles IP. Un nombre exact de match ou match_set.
src_filter_id	chaîne	ID du filtre d'inventaire source Celui-ci sera défini lorsque la commande include_filter_names=vrai sera transmise en tant que paramètre.
src_filter_name	chaîne	Nom du filtre d'inventaire source. Celui-ci sera défini lorsque la commande include_filter_names=vrai sera transmise en tant que paramètre.
dst_filter_id	chaîne	ID du filtre d'inventaire de destination. Celui-ci sera défini lorsque la commande include_filter_names=vrai sera transmise en tant que paramètre.
dst_filter_name	chaîne	Nom du filtre d'inventaire de destination. Celui-ci sera défini lorsque la commande include_filter_names=vrai sera transmise en tant que paramètre.

ContainerNetworkPolicy

Attribut	Type	Description
pod_id	chaîne	ID de POD.

Attribut	Type	Description
network_policy	tableau	Tableau d'objets de la Politique réseau
déploiement	chaîne	Nom du déploiement
service_endpoint	tableau	Liste des noms de points terminaux de service.

Mettre en correspondance

Attribut	Type	Description
src_addr	objet	Objet de sous-Sous-réseau pour l'adresse source
dst_addr	objet	Objet de sous-Sous-réseau pour l'adresse de destination
src_port_range_start	int	Début de la plage du port source
src_port_range_end	int	Fin de la plage du port source
dst_port_range_start	int	Début de la plage du port de destination
dst_port_range_end	int	Fin de la plage du port de destination
ip_protocol	chaîne	Protocole IP
address_family	chaîne	Famille d'adresses IPv4 ou IPv6
direction	chaîne	Direction de la correspondance, INGRESS (ENTRÉE) ou EGRESS (SORTIE).
src_addr_range	objet	ObjetPlage d'adresses pour l'adresse source.
dst_add_range	objet	ObjetPlage d'adresses pour l'adresse de destination.

Action

Attribut	Type	Description
type	chaîne	Type d'action

Ensemble de correspondances

Attribut	Type	Description
src_set_id	chaîne	ID d'ensemble source de l'objet de Ensemble d'adresses dans Configuration de la politique de réseau `Ensembles_d'adresses` tableau
dst_set_id	chaîne	Tableau des ID d'ensemble de destination de l'objet de Ensemble d'adresses dans Configuration de la politique de réseau `Ensembles_d'adresses`
src_ports	tab-leau	Tableau d'objets de Plage de ports pour les ports sources
dst_ports	tab-leau	Tableau d'objets de Plage de ports pour les ports de destination.
ip_protocol	chaîne	Protocole IP
address_family	chaîne	Famille d'adresses IPv4 ou IPv6
direction	chaîne	Direction de la correspondance, INGRESS (ENTRÉE) ou EGRESS (SORTIE).

Ensemble d'adresses

Attribut	Type	Description
id_ensemble	chaîne	ID d'ensemble d'adresses
addr_ranges	tableau	Tableau d'objets de Plage d'adresses
subnets	tableau	Tableau d'objets deSous-réseau.
addr_family	chaîne	Famille d'adresses IPv4 ou IPv6

Sous-réseau

Attribut	Type	Description
ip_addr	chaîne	Adresse IP.
prefix_length	int	Longueur de préfixe pour le sous-réseau.

Plage d'adresses

Attribut	Type	Description
start_ip_addr	chaîne	Adresse IP de début de la plage
end_ip_addr	chaîne	Adresse IP de fin de la plage

Plage de ports

Attribut	Type	Description
start_port	int	Port de début de la plage.
end_port	int	Port de fin de la plage.

État de configuration de l'agent

Attribut	Type	Description
désactivé	booléen	La configuration indiquant que l'application est désactivée pour l'agent.
current_version	nombre	Version de configuration actuelle de l'agent appliquée à l'agent.
highest_seen_version	nombre	Version la plus récente de la configuration de l'agent reçue par l'agent.

Configuration de politique de réseau mise en service

Attribut	Type	Description
version	chaîne	Version de configuration de politique de réseau mise ne œuvre par l'agent.
error_reason	chaîne	CONFIG_SUCCESS (CONFIGURATION_SUCCÈS) lorsque l'agent a appliqué les politiques avec succès, sinon motif de l'erreur.
désactivé	booléen	La configuration indiquant que l'application est désactivée pour l'agent.
current_version	nombre	Version actuelle du NPC appliquée à l'agent.

Attribut	Type	Description
highest_seen_version	nombre	Version la plus récente du NPC reçue par l'agent.
policy_status	objet	Chaque état de politique de réseau

Renseignements sur l'agent

Attribut	Type	Description
agent_info_supported	booléen	La capacité de l'agent si agent_info est pris en charge.
ipset_supported	booléen	La capacité d'agent si les ensembles d'adresses IP sont pris en charge.

Résultat de la politique concrète

Attribut	Type	Description
byte_count	int	Nombre d'octets pour les résultats de politiques concrètes.
pkt_count	int	Nombre de paquets pour les résultats des politiques concrètes.

Série chronologique du résultat de la politique concrète

Attribut	Type	Description
Horodatage	chaîne	Chaîne d'horodatage pour l'agrégation des résultats.
résultat	objet	Résultat de la politique concrète

Configuration client-serveur

La détection des relations client-serveur est au cœur de diverses fonctions de Cisco Secure Workload. C'est pourquoi nous vous recommandons d'utiliser l'agent logiciel chaque fois que cela est possible, car il peut communiquer la situation réelle. Aucun point de surveillance de la télémétrie du réseau ne peut garantir l'observation de chaque paquet pour un flux donné - en raison d'un large éventail de circonstances, par exemple : deux moitiés unidirectionnelles d'un flux TCP peuvent prendre des chemins uniques dans le réseau et seront donc toujours inévitablement affectées par un niveau d'erreur.

Cisco Secure Workload tente de détecter et de minimiser ces erreurs sans aucune interaction avec l'utilisateur en appliquant des algorithmes d'apprentissage automatique à chaque flux, en créant un modèle statistique qui fournit un jugement lorsqu'une télémétrie incohérente est signalée. Dans la majorité des cas, les utilisateurs n'ont pas à se préoccuper de cet ensemble d'API. Cependant, dans certains cas, l'algorithme de détection

client-serveur n'obtient pas la bonne direction du flux. Les fonctionnalités qui dépendent de la direction du flux, par exemple la découverte automatique des politiques, peuvent présenter des comportements indésirables comme l'ouverture de ports inutiles.

Un ensemble d'API est fourni qui peut être utilisé pour fournir des conseils sur les ports de serveurs connus aux algorithmes de Cisco Secure Workload. Cet ensemble d'API est disponible pour les utilisateurs ayant le rôle de propriété de portée racine et nécessite la capacité `app_policy_management` associée à la clé API pour ces utilisateurs.

Il existe deux possibilités pour la configuration client serveur :

Configuration de l'hôte

Configuration de ports de serveur connus qui sont applicables à un sous-ensemble spécifique d'adresses IP dans une portée racine

Ajouter une configuration de port de serveur

Cette API peut être utilisée pour fournir des conseils aux algorithmes Cisco Secure Workload au sujet des ports de serveur connus pour une portée racine donnée. Vous pouvez fournir une liste de ports de serveur TCP/UDP connus pour un ensemble d'adresses IP appartenant à une portée racine afin d'aider les algorithmes Cisco Secure Workload à déterminer correctement la direction client-serveur dans les flux.

```
POST /openapi/v1/adm/{root_scope_id}/server_ports
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Identifiant unique de la portée du rôle

En outre, un fichier texte fourni comme entrée de cette API contient la configuration de port du serveur de point terminal au format suivant :

Configuration du port du serveur de point terminal

Attribut	Type	Description
ip_address	chaîne	L'adresse IP (peut être une adresse IPv4 ou IPv6). Les sous-réseaux ne sont pas autorisés.
tcp_server_ports	Liste des int	Liste des ports de serveur TCP connus correspondant à ip_address.
udp_server_ports	Liste des int	Liste des ports de serveur UDP connus correspondant à ip_address.

Configuration en bloc de ports de serveur

Attribut	Type	Description
host_config	Liste des objets de Configuration du port du serveur de point terminal .	Liste des adresses IP avec serveur connu associé ports.

Exemple de code Python

```
# contents of below file:
# {"host_config": [
#    {"ip_address": "1.1.1.1",
#     "tcp_server_ports": [100, 101, 102],
#     "udp_server_ports": [103]
#    },
#    {"ip_address": "1.1.1.2",
#     "tcp_server_ports": [200, 201, 202]
#    }
#  ]
# }

file_path = '/<path_to_file>/server_ports.txt'
root_scope_id = '<root-scope-id>'
restclient.upload(file_path,
                  '/adm/%s/server_ports' % root_scope_id,
                  timeout=200) # seconds
```

Note L'API ci-dessus écrase l'état complet de la configuration des ports de serveur connue dans le serveur principal. Si vous devez modifier quelque chose, ils doivent retélécharger la configuration complète après les modifications.

Obtenir une configuration de port de serveur

Cette API renvoie la liste des ports de serveur téléversés connus pour les points terminaux d'une portée racine.

```
GET /openapi/v1/adm/{root_scope_id}/server_ports
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Identifiant unique de la portée du rôle

Objet de réponse : une liste des objets de référence :*ServerPortConfig* .

Exemple de code Python

```
root_scope_id = '<root-scope-id>'
restclient.get('/adm/%s/server_ports' % root_scope_id)
```

Supprimer une configuration de port de serveur

Cette API supprime la configuration de port du serveur pour la portée racine spécifiée.

```
DELETE /openapi/v1/adm/{root_scope_id}/server_ports
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Identifiant unique de la portée du rôle

Objet de réponse : aucun.

Exemple de code Python

```
root_scope_id = '<root-scope-id>'
restclient.delete('/adm/%s/server_ports' % root_scope_id)
```

Configuration de port

La configuration des ports de serveur connus qui sont applicables à toutes les adresses IP qui appartiennent à une portée racine

Envoyer une configuration de port de serveur

Cette API peut être utilisée pour fournir des conseils aux algorithmes Cisco Secure Workload au sujet des ports de serveur connus pour une portée racine donnée. Les utilisateurs peuvent fournir une liste de ports de serveur TCP/UDP connus pour une portée racine donnée afin d'aider les algorithmes de Cisco Secure Workload à déterminer la direction client-serveur correcte dans les flux. Les utilisateurs ont également la possibilité d'associer un nom de service à chaque port de serveur.

Il existe également une liste par défaut des services connus qui sont applicables à toutes les portées racine (ci-après appelés services globaux). Cette liste peut être remplacée à tout moment par l'utilisateur.

Configuration du service

Un service est défini comme une paire (port, nom).

Attribut	Type	Description
port	int	TCP/UDP server port number
name	chaîne	Nom du service associé à ce port (facultatif)
override_in_conflicts	booléen	Forcer l'hôte à être le fournisseur en cas de conflit (facultatif)

Configuration des services en masse

Attribut	Sous-attribut	Type	Description
server_ports_config	tcp_service_list	Liste des objets de *configuration de service*.	Listes des services TCP connus
	udp_service_list	Liste des objets de *configuration de service*.	Listes des services UDP connus

```
Services Push par portée racine :

POST /openapi/v1/adm/{root_scope_id}/server_ports_config
```

Exemple de code Python

```
# contents of below file:
#{"server_ports_config":
#    {
#        "tcp_service_list": [
#                {
#                        "port": 80,
#                        "name" : "http"
#
#                },
#                {
#                        "port": 53,
#                        "name" : "dns"
#                },
#                {
#                        "port": 514,
#                        "name" : "syslog",
#                        "override_in_conflicts": true
#                }
#        ],
#        "udp_service_list": [
#                {
#                        "port": 161
#                },
#                {
#                        "port": 53,
#                        "name": "dns"
#                }
#        ]
#    }
#}

file_path = '/<path_to_file>/server_ports.json'

# Updating service list for a given root scope
#restclient.upload(file_path,
#                  '/openapi/v1/adm/{root_scope_id}/server_ports_config',
#                  timeout=200) # seconds
```

Note L'API ci-dessus écrase l'état complet de la configuration des ports de serveur connue dans le serveur principal. Si l'utilisateur doit modifier quelque chose, il doit télécharger à nouveau la configuration complète après les modifications.

Récupérer la configuration de port du serveur

Cette API renvoie la liste des ports de serveur connus d'une portée racine téléversée par l'utilisateur. La réponse est *Configuration de service en masse.*

```
Retrieve configured services per root scope:
    GET /openapi/v1/adm/{root_scope_id}/server_ports_config

Retrieve configured global services:
    GET /openapi/v1/adm/server_ports_config
```

Supprimer la configuration de port du serveur

Cette API supprime la configuration de port du serveur pour la portée racine spécifiée.

```
Remove configured services per root scope:
    DELETE /openapi/v1/adm/{root_scope_id}/server_ports_config
```

Agents logiciels

API des agents

Les API des agents logiciels sont associées à la gestion des agents logiciels Cisco Secure Workload. Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API. Les API *GET* ci-dessous sont également disponibles avec la capacité `flow_inventory_query` associée à la clé API.

Obtenir des agents logiciels

Ce point terminal renvoie une liste des agents logiciels. Chaque agent logiciel comporte deux champs pour décrire son type d'agent, `agent_type_str` est en texte brut tandis que `agent_type` est une énumération.

```
GET /openapi/v1/sensors
```

Paramètres :

Nom	Type	Description
limit	nombre entier	Limite le nombre de résultats renvoyés (facultatif)
offset	chaîne	Le décalage est utilisé pour les demandes paginées. Si la réponse renvoie un décalage, la requête suivante doit utiliser le même décalage pour obtenir plus de résultats sur la page suivante. (facultatif)

Obtenir un agent logiciel spécifique

Ce point terminal renvoie les attributs pour l'agent logiciel dont l'UUID fait partie de l'URI. Chaque agent logiciel comporte deux champs pour décrire son type d'agent.<agent_type_str> est en texte brut alors que<agent_type> est une énumération.

```
GET /openapi/v1/sensors/{uuid}
```

Suppression de l'agent logiciel

Ce point terminal est utilisé pour désactiver un agent logiciel en fonction de son UUID.

Utiliser l'API avec prudence; une fois qu'un agent est supprimé, il n'est plus disponible dans le tableau de bord Cisco Secure Workload et si l'agent est actif, les exportations de flux de l'agent ne sont pas autorisées dans Cisco Secure Workload.

```
DELETE /openapi/v1/sensors/{uuid}
```

Configuration de l'agent logiciel à l'aide des intents

Ce flux de travail d'API utilise quelques points terminaux REST définis ci-dessous.

Créer un filtre d'inventaire

Ce point terminal est utilisé pour préciser les critères qui correspondent aux hôtes d'agents sur lesquels l'utilisateur souhaite configurer des agents logiciels.

```
POST /openapi/v1/filters/inventories
```

Paramètres :

Nom	Type	Description
app_scope_id	chaîne	L'ID de la portée à affecter au filtre d'inventaire.
name	chaîne	Un nom pour le filtre d'inventaire
query	json	Filtre ou critères de correspondance pour l'hôte de l'agent.

Exemple de code Python

```
# app_scope_id can be retrieved by /app_scopes API
req_payload = {
   "app_scope_id": <app_scope_id>,
   "name": "sensor_config_inventory_filter",
   "query": {
      "type": "eq",
      "field": "ip",
      "value": <sensor_interface_ip>
   }
}
resp = restclient.post('/filters/inventories',
               json_body=json.dumps(req_payload)))
print resp.status_code
# returned response will contain the created filter and it's ID.
```

Création d'un profil de configuration d'agent logiciel

Ce point terminal est utilisé pour préciser l'ensemble d'options de configuration à appliquer à l'ensemble cible des agents logiciels.

```
POST /openapi/v1/inventory_config/profiles
```

Les options de configuration suivantes peuvent être spécifiées dans le profil de configuration de l'agent :

- allow_broadcast : option pour autoriser/interdire le trafic de diffusion (la valeur par défaut de cette option est True).
- allow_multicast : option pour autoriser/interdire le trafic en multidiffusion (la valeur par défaut de cette option est True).
- allow_link_local : option pour autoriser/interdire le trafic local (la valeur par défaut de cette option est True).
- auto_upgrade_opt_out : si la valeur est « vrai », les agents ne sont pas mis à niveau automatiquement lors de la mise à niveau de la grappe Cisco Secure Workload.
- cpu_quota_mode et cpu_quota_us : ces options sont utilisées pour contrôler la quantité de quota de processeur à octroyer à l'agent sur l'hôte final.
- data_plane_disabled : si la valeur est « vrai », l'agent arrête de signaler les flux à Cisco Secure Workload.
- enable_conversation_flows : option pour activer le mode conversation sur tous les agents.
- enable_forensics : option permettant d'activer la collecte des événements criminalistiques sur la charge de travail (par conséquent, l'agent utilise plus de CPU).
- enable_meltdown : active la détection d'exploit de fusion sur les charges de travail (l'agent utilise plus de CPU).
- allow_pid_lookup : si la valeur est True (vraie), l'agent tente de joindre des informations de processus aux flux. Notez que cette option de configuration utilise plus de CPU sur l'hôte final.
- enforcement_disabled : peut être utilisé pour désactiver l'application sur les hôtes exécutant des agents d'application.
- preserve_existing_rules : option pour préciser s'il faut conserver les règles iptable existantes.
- windows_enforcement_mode : option pour utiliser WAF (pare-feu avancé Windows) ou WFP (plateforme de filtrage Windows) (l'option par défaut est WAF).

Pour en savoir plus sur les options de configuration, consultez la section Configuration de l'agent logiciel .

Exemple de code Python

```
# Define profile to disable data_plane on agent
req_payload = {
   "root_app_scope_id": <root_app_scope_id>,
   "data_plane_disabled": True,
   "name": "sensor_config_profile_1",
   "enable_pid_lookup": True,
   "enforcement_disabled": False
}
resp = restclient.post('/inventory_config/profiles',
               json_body=json.dumps(req_payload))
print resp.status_code
```

```
# returned response will contain the created profile and it's ID.
parsed_resp = json.loads(resp.content)
```

Obtenir les profils de configuration d'agent logiciel

Ce point terminal renvoie une liste des profils de configuration d'agents logiciels visibles par l'utilisateur.

```
GET /openapi/v1/inventory_config/profiles
```

Paramètres : Aucun

Obtenir un profil de configuration d'agent logiciel précis

Ce point terminal renvoie une instance du profil de configuration d'agent logiciel.

```
GET /openapi/v1/inventory_config/profiles/{profile_id}
```

Renvoie l'objet de profil de configuration de l'agent logiciel associé à l'ID précisé.

Mettre à jour un profil de configuration d'agent logiciel

Ce point terminal met à jour un profil de configuration d'agent logiciel.

```
PUT /openapi/v1/inventory_config/profiles/{profile_id}
```

Les options de configuration suivantes peuvent être spécifiées dans le profil de configuration de l'agent :

- allow_broadcast : option pour autoriser/interdire le trafic de diffusion (la valeur par défaut de cette option est True).
- allow_multicast : option pour autoriser/interdire le trafic en multidiffusion (la valeur par défaut de cette option est True).
- allow_link_local : option pour autoriser/interdire le trafic local (la valeur par défaut de cette option est True).
- auto_upgrade_opt_out : si la valeur est « vrai », les agents ne sont pas mis à niveau automatiquement lors de la mise à niveau de la grappe Cisco Secure Workload.
- cpu_quota_mode et cpu_quota_us : ces options sont utilisées pour contrôler la quantité de quota de processeur à octroyer à l'agent sur l'hôte final.
- data_plane_disabled : si la valeur est « vrai », l'agent arrête de signaler les flux à Cisco Secure Workload.
- enable_conversation_flows : option pour activer le mode conversation sur tous les agents.
- enable_forensics : option permettant d'activer la collecte des événements criminalistiques sur la charge de travail (par conséquent, l'agent utilise plus de CPU).
- enable_meltdown : active la détection d'exploit de fusion sur les charges de travail (l'agent utilise plus de CPU).
- allow_pid_lookup : si la valeur est True (vraie), l'agent tente de joindre des informations de processus aux flux. Notez que cette option de configuration utilise plus de CPU sur l'hôte final.
- enforcement_disabled : peut être utilisé pour désactiver l'application sur les hôtes exécutant des agents d'application.
- preserve_existing_rules : option pour préciser s'il faut conserver les règles iptable existantes.

- windows_enforcement_mode : option pour utiliser WAF (pare-feu avancé Windows) ou WFP (plateforme de filtrage Windows) (l'option par défaut est WAF).

Pour en savoir plus sur les options de configuration, consultez la section Configuration de l'agent logiciel .

Renvoie l'objet de profil de configuration de l'agent logiciel modifié associé à l'ID précisé.

Supprimer un profil de configuration d'agent logiciel

Ce point terminal supprime le profil de configuration d'agent logiciel précisé.

```
DELETE /openapi/v1/inventory_config/profiles/{profile_id}
```

Créer un intent de configuration d'agent logiciel

Ce point terminal est utilisé pour préciser l'intention d'appliquer un ensemble d'options de configuration à un ensemble précisé d'agents logiciels. Cela créera l'intent et mettra à jour l'ordre de l'ensemble des intents en ajoutant l'intent nouvellement créé à cet ordre.

```
POST /openapi/v1/inventory_config/intents
```

Exemple de code Python

```
req_payload = {
  "inventory_config_profile_id": <>,
  "inventory_filter_id": <>
}
resp = restclient.post('/inventory_config/intents',
               json_body=json.dumps(req_payload))
print resp.status_code
# returned response will contain the created intent object and it's ID.
```

Préciser l'ordre des intents

Ce point terminal est utilisé pour préciser l'ordre des divers intents de configuration des agents logiciels. Par exemple, il peut y avoir deux intents : le premier pour activer la recherche d'ID de processus sur les ordinateurs de développement et l'autre pour désactiver la recherche d'ID de processus sur les ordinateurs Windows. Si le premier intent a une priorité plus élevée, la recherche d'ID de processus sera activée sur les ordinateurs Windows de développement. REMARQUE : Par défaut, lors de sa création, l'intent est ajouté au début de la liste de l'ordre des intents. Ce point terminal ne doit être utilisé que si l'utilisateur final doit modifier l'ordre existant des intents.

```
POST /openapi/v1/inventory_config/orders
```

Exemple de code Python

```
# Read the agent config intents ordered list
resp = restclient.get('/inventory_config/orders')
order_result_json = json.loads(resp.content)

# Modify the list by prepending the new intent in the list
order_rslt_json['intent_ids'].insert(0,<intent_id>)

# Post the new ordering back to the server
resp = restclient.post('/inventory_config/orders',
               json_body=json.dumps(order_rslt_json))
```

Supprimer l'intent de configuration de l'agent

Ce point terminal est utilisé pour supprimer un intent de configuration d'agent spécifique.

```
DELETE /openapi/v1/inventory_config/intents/{intent_id}
```

Exemple de code Python

```
intent_id = '588a51dcb5b30d0ee6da084a'
resp = restclient.delete('/inventory_config/intents/%s' % intent_id)
```

Intents de configuration d'interface

La méthode recommandée pour affecter des VRF aux agents est d'utiliser les paramètres de configuration d'un VRF distant. Dans de rares cas, lorsque les hôtes d'agents peuvent avoir plusieurs interfaces auxquelles doivent être affectées des VRF différents, les utilisateurs peuvent choisir de leur affecter des VRF à l'aide des intents de configuration d'interface. Accédez à **Manage (Gestion)** > **Agents (Agents)** et cliquez sur l'onglet **Configure (Configurer)**.

Objet intent de configuration d'inventaire

Les méthodes GET et POST renvoient un tableau d'objets JSON d'intent de configuration d'inventaire. Les attributs de l'objet sont décrits ci-dessous :

Attribut	Type	Description
vrf_id	nombre entier	valeur entière de l'ID VRF
vrf_name	chaîne	Nom VRF
inventory_filter_id	chaîne	ID de filtre d'inventaire
inventory_filter	JSON	Filtre d'inventaire Consultez OpenAPI > Filtres d'inventaire pour plus de détails.

Obtenir les intents de configuration d'interface

Ce point terminal renvoie une liste des intents de configuration d'inventaire à l'utilisateur.

```
GET /openapi/v1/inventory_config/interface_intents
```

Paramètres : Aucun

Créer ou mettre à jour la liste des intents de configuration d'interface

Ce point terminal est utilisé pour créer ou modifier la liste des intents de configuration d'interface. L'API prend une liste ordonnée d'intents. Pour supprimer un intent de cette liste, les utilisateurs doivent lire la liste existante des intents, la modifier et réécrire la liste modifiée.

```
POST /openapi/v1/inventory_config/interface_intents
```

Paramètres :

Nom	Type	Description
inventory_filter_id	chaîne	ID du filtre d'inventaire correspondant à l'interface
vrf_id	nombre entier	ID VRF à attribuer à l'interface

Exemple de code Python

```
  req_payload = {
    "intents": [
      {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_1>},
      {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_2>}
    ]
  }
  resp = restclient.post('/inventory_config/interface_intents',
json_body=json.dumps(req_payload))
```

Configuration VRF pour les agents derrière la NAT

Les ensembles d'API suivants sont utiles pour préciser les politiques d'attribution des VRF aux agents derrière les boîtiers NAT. Ces ensembles d'API nécessitent la capacité `sensor_management` associée à la clé API et ne sont disponibles que pour les administrateurs de site.

Répertorier les règles de configuration VRF pour les agents derrière la NAT

Ce point terminal renvoie une liste des règles de configuration VRF applicables aux agents derrière la NAT.

```
GET /openapi/v1/agentnatconfig
```

Créer une nouvelle configuration VRF applicable aux agents derrière la NAT

Ce point terminal est utilisé pour préciser les critères d'étiquetage VRF pour les hôtes en fonction de leur adresse IP source et de leur port source, comme vus par l'appareil Cisco Secure Workload .

```
POST /openapi/v1/agentnatconfig
```

Paramètres :

Nom	Type	Description
src_subnet	chaîne	Sous-réseau auquel l'adresse IP source peut appartenir (notation CIDR).
src_port_range_start	nombre entier	Limite inférieure de la plage de ports source (0 à 65 535).
src_port_range_end	nombre entier	Limite supérieure de la plage du port source (0 à 65 535).
vrf_id	nombre entier	ID VRF à utiliser pour étiqueter les flux des agents dont l'adresse source et le port source se situent dans la plage précisée ci-dessus.

Exemple de code Python

```
req_payload = {
    src_subnet: 10.1.1.0/24,          # src IP range for sensors
    src_port_range_start: 0,
    src_port_range_end: 65535,
    vrf_id: 676767                    # VRF ID to assign
}

resp = rc.post('/agentnatconfig', json_body=json.dumps(req_payload))
print resp.status_code
```

Supprimer la configuration VRF existante

```
DELETE /openapi/v1/agentnatconfig/{nat_config_id}
```

Téléchargement du logiciel Cisco Secure Workload

La fonction de téléchargement de logiciel Cisco Secure Workload permet de télécharger des paquets logiciels pour les agents Cisco Secure Workload. Ces ensembles d'API nécessitent la capacité `software_download` associée à la clé API. Cette fonctionnalité est uniquement offerte aux utilisateurs administrateurs du site, aux propriétaires de la portée racine et aux utilisateurs ayant des rôles d'installation d'agent.

API pour obtenir les plateformes prises en charge

Ce point de terminaison renvoie la liste des plateformes prises en charge.

```
GET /openapi/v1/sw_assets/platforms
```

Paramètres : Aucun

Objet de réponse : renvoie la liste des plateformes prises en charge.

Exemple de code Python

L'exemple de code ci-dessous récupère toutes les plateformes prises en charge.

```
resp = restclient.get('/sw_assets/platforms')
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Exemple de réponse

```
{"results": [{"platform": "OracleServer-6.3", "agent_type": "enforcer", "arch": "x86_64"},
 {"platform": "MSWindows8Enterprise", "agent_type": "legacy_sensor", "arch": "x86_64"}]}
```

API pour obtenir la version logicielle prise en charge

Ce point terminal renvoie la liste des versions de logiciel prises en charge pour des « type_agent », « type_de_paquet », « plateforme » et « architecture » précisés.

```
GET
/openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&list_version=<list_version>&installation_id=<installer_id>
```

où <agent_type> , <platform> et <arch> peuvent être l'un des résultats extraits de l' **API pour obtenir les plateformes prises en charge**, et <pkg_type> peut être « capteur_w_cfg » ou « capteur_bin_pkg ». Les deux

paramètres <pkg_type> et<agent_type> sont facultatif, mais au moins l'un d'eux doit être spécifié. <list_version>doit être « True » pour activer cette API.

Paramètres : L'URL de la demande contient les paramètres suivants.

Nom	Type	Description
intégrée	chaîne	Précisez la plateforme.
agent_type	chaîne	(Facultatif) Précisez le type d'agent.
pkg_type	chaîne	(Facultatif) Précisez le type de paquet. La valeur peut être « sensor_w_cfg » ou « sensor_bin_pkg ».
arch	chaîne	Préciser l'architecture.
list_version	chaîne	Définissez la valeur « True » pour activer la recherche de version du logiciel.

Objet de réponse : renvoie une liste des versions de logiciels prises en charge.

Exemple de code Python

```
resp =
restclient.get('/sw_assets/download?platform=OracleServer-6.3&pkg_type=sensor_w_cfg&arch=x86_64&list_version=True')
if resp.status_code == 200:
   print resp.content
```

```
resp =
[illegible]
if resp.status_code == 200:
   print resp.content
```

Exemple de réponse

```
3.3.1.30.devel
3.3.1.31.devel
```

API pour créer l'ID du programme d'installation

Ce point terminal crée un ID d'installation pour permettre à l'API de télécharger le logiciel Cisco Secure Workload.

```
GET /openapi/v1/sw_assets/installation_id
```

Objet de réponse : renvoie un ID du programme d'installation qui peut être utilisé dans l'API pour télécharger le logiciel Cisco Secure Workload.

Exemple de code Python

```
resp = restclient.get('/sw_assets/installation_id')
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Exemple de réponse

```
"[illegible]"
```

API pour télécharger le logiciel Cisco Secure Workload

Ce point terminal permet aux clients de télécharger le logiciel pour des « type-agent », « type_de_paquet », « plateforme », « architecture » et « version_capteur » précisés.

```
GET
/openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&sensor_version=<sensor_version>&installation_id=<installer_id>
```

où <agent_type> , <platform> et <arch> peuvent être l'un des résultats extraits de l' **API pour obtenir les plateformes prises en charge**, et <pkg_type> peut être « capteur_w_cfg » ou « capteur_bin_pkg ». Les deux paramètres <pkg_type> et<agent_type> sont facultatif, mais au moins l'un d'eux doit être spécifié. <sensor_version>peut être n'importe lequel des résultats extraits de l' **API pour obtenir la version de logiciel prise en charge**. Si « sensor_version » n'est pas spécifié, la **dernière** version du logiciel sera téléchargée.

Paramètres : L'URL de la demande contient les paramètres suivants.

Nom	Type	Description
intégrée	chaîne	Précisez la plateforme.
agent_type	chaîne	(Facultatif) Précisez le type d'agent.
pkg_type	chaîne	(Facultatif) Précisez le type de paquet. La valeur peut être « sensor_w_cfg » ou « sensor_bin_pkg ».
arch	chaîne	Préciser l'architecture.
sensor_version	chaîne	(Facultatif) Spécifiez la version du logiciel, la valeur par défaut est une chaîne vide.

Objet de réponse : renvoie le logiciel Cisco Secure Workload correspondant aux paramètres fournis.

Exemple de code Python

```
resp =
[redacted]
if resp.status_code == 200:
   print 'file downloaded successfully'
```

Mise à niveau des agents Cisco Secure Workload

La fonction de mise à niveau des agents Cisco Secure Workload permet de mettre à niveau les agents Cisco Secure Workload installés vers une version donnée. Elle met uniquement à jour les métadonnées, la mise à niveau réelle aura lieu lors du prochain enregistrement. L'API nécessite la fonctionnalité `software_download` associée à la clé API. Cette fonctionnalité est uniquement offerte aux utilisateurs administrateurs du site, aux propriétaires de la portée racine ou aux utilisateurs ayant des rôles d'installation d'agent.

API pour mettre à niveau un agent vers une version spécifique

Ce point terminal déclenche la mise à niveau de l'agent, compte tenu de sa mise à niveau « UUID » vers une « sensor_version » (version de capteur) spécifique. La dernière version sera appliquée si « sensor_version » n'est pas fourni. Cette API ne traitera pas les demandes de rétrogradation de version.

```
POST /openapi/v1/sensors/{UUID}/upgrade?sensor_version=<sensor_version>
```

où <sensor_version> peut être n'importe lequel des résultats extraits de l' API pour obtenir la version logicielle prise en charge .

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
sensor_version	chaîne	(facultatif) Précisez la version souhaitée, la dernière version sera appliquée par défaut

Renvoie l'état de cette demande de mise à niveau.

Exemple de code Python

```
resp = restclient.post('/openapi/v1/sensors/{UUID}/upgrade?sensor_version=3.4.1.1.devel')

if resp.status_code == 200:
  print 'agent upgrade was triggered successfully and in progress'
elif resp.status_code == 304:
  print 'provided version is not newer than current version'
elif resp.status_code == 400:
  print 'provided version is invalid'
elif resp.status_code == 403:
  print 'user does not have required capability'
elif resp.status_code == 404:
  print 'agent with {UUID} does not exist'
```

Règles de collecte

Cet ensemble d'API peut être utilisé pour gérer les règles de collecte. Les règles de collecte dans l'appareil Cisco Secure Workload permettent à l'utilisateur de préciser les adresses IP ou les sous-réseaux intéressants pour son déploiement. Lors de la réception de ces règles de collecte, les agents extraient uniquement les signaux de trafic pour les adresses IP qui correspondent à ces ensembles de règles de collecte. Ces API nécessitent la capacité `flow_inventory_query` associée à la clé API.

Note Ces API ne sont disponibles que pour les administrateurs du site.

Objet règle de collecte

Les attributs de l'objet règle de collecte sont décrits ci-dessous :

Attribut	Type	Description
subnet	chaîne	Sous-réseau ou adresse IP au format CIDR.
action	chaîne	Les valeurs possibles sont « INCLUDE (INCLURE) » ou « EXCLUDE (EXCLURE) ».

Mettre à jour les règles de collecte pour un VRF

Ce point terminal peut être utilisé pour mettre à jour la liste ordonnée des règles de collecte pour le VRF spécifié. Remarque : la liste des règles de collecte dans la requête POST est traitée comme une liste ordonnée.

```
POST /openapi/v1/collection_rules/{vrf_name}
```

Paramètres :

Liste ordonnée des objets règle de collecte dans le corps de la requête POST. **Les deux dernières règles doivent être les règles collectrices pour IPv4 et IPv6.** Les règles peuvent préciser les sous-réseaux 0.0.0.0/0 et::/0, respectivement, comme dans l'exemple ci-dessous.

Objet de réponse : liste ordonnée mise à jour des règles de collecte pour le VRF.

Exemple de code Python

```
req_payload = [
    {
       "subnet": "10.10.10.0/24",
       "action": "INCLUDE"
    },
    {
       "subnet": "11.11.11.0/24",
       "action": "INCLUDE"
    },
    {
       "subnet": "0.0.0.0/0",   # catch all rule for IPV4 addresses
       "action": "EXCLUDE"
    },
    {
       "subnet": "::/0",        # catch all rule for IPV6 addresses
       "action": "EXCLUDE"
    }
]
resp = restclient.post('/collection_rules/test_vrf', json_body=json.dumps(req_payload))
```

Obtenir les règles de collecte pour un VRF

Ce point terminal renvoie une liste ordonnée de règles de collecte pour un VRF spécifié.

```
GET /openapi/v1/collection_rules/{vrf_name}
```

Paramètres : Aucun

Objet de réponse : liste ordonnée des règles de collecte pour un VRF spécifié.

Exemple de code Python

```
resp = restclient.get('/collection_rules/test_vrf')
```

Incidence des règles de collecte

Il existe deux types d'éléments dans l'inventaire :

- Appris par le capteur (profil de charge de travail) : comprend toutes les adresses IP qui appartiennent aux charges de travail exécutant des capteurs Cisco Secure Workload
- Appris par le flux (Profil d'inventaire) : comprend toutes les adresses IP qui ont été vues dans les signaux de flux collectés par Cisco Secure Workload, mais qui ne sont associées à aucune charge de travail exécutant des agents Cisco Secure Workload.

Les règles de collecte EXCLUDE/INCLUDE contrôlent les éléments de l'inventaire qui font l'objet d'un suivi. Les éléments de l'inventaire appris par le capteur sont toujours suivis, quelles que soient les règles de collecte. Pour les éléments d'inventaire appris par le flux, s'ils sont exclus par les règles de collecte, l'élément d'inventaire n'existera pas. Par conséquent, la recherche d'inventaire ne renverra aucun résultat pour de tels inventaires.

La recherche de flux n'est pas affectée par les règles de collecte, sauf la colonne des étiquettes, qui ne sera pas remplie pour l'adresse IP exclue par les règles de collecte. Les règles de collecte n'ont aucune incidence sur la détermination du client-serveur pour un flux donné.

Les résultats de la découverte automatique des politiques peuvent être affectés, car nous ne suivons pas les étiquettes des adresses IP exclues par les règles de collecte.

Condensés de fichiers téléversés par l'utilisateur

Les utilisateurs peuvent télécharger une liste des condensés de fichiers dans Cisco Secure Workload et préciser si ces condensés sont inoffensifs ou marqués. Cisco Secure Workload signale en conséquence les processus avec les condensés binaires respectifs.

Cet ensemble d'API peut être utilisé pour charger ou supprimer une liste de condensés de fichiers dans Cisco Secure Workload. Pour appeler ces API, utilisez une clé API avec la capacité `user_data_upload`.

Note Vous pouvez avoir jusqu'à 1 million de condensés de fichiers par portée racine. 500 000 pour les condensés inoffensifs et marqués respectivement.

Les API suivantes sont à la disposition des propriétaires de la portée et des administrateurs de site et sont utilisées pour charger/télécharger/supprimer les condensés de fichiers dans une portée racine unique sur le |produit| appareil .

Téléversement du condensé de fichier par l'utilisateur

Ce point terminal est utilisé pour charger un fichier CSV avec le condensé de fichier pour une portée racine sur l'appareil Cisco Secure Workload. Les en-têtes de colonne `HashType` (type de condensé) et `FileHash` (condensé de fichier) doivent apparaître dans le fichier CSV. `HashType` doit être `SHA-1` ou `SHA-256`, FileHash ne doit pas être vide et doit être au format 40-hex SHA1 ou 64-hex SHA256.

Les en-têtes`FileName` (Nom de fichier) et `Notes` (Remarques) sont facultatifs. Le nom de fichier donné ne doit pas dépasser la longueur maximale de 150 caractères et les notes fournies ne doivent pas dépasser la longueur maximale de 1024 caractères.

```
POST /openapi/v1/assets/user_filehash/upload/{rootAppScopeNameOrID}/{benignOrflagged}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeNameOrID	chaîne	Nom ou ID de la portée racine.
benignOrflagged	chaîne	Peut être égal à `benign` (bénin) ou `flagged` (signalé).

Objet de réponse : aucun

Exemple de code Python

```
  # Sample CSV File
  # HashType,FileHash,FileName,Notes
  # SHA-1,1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29,application_1.exe,Sample Notes
  #
SHA-256,8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4,application_2.exe,Sample
 Notes

  file_path = '/<path_to_file>/user_filehash.csv'
  root_app_scope_name = 'Tetration'
  restclient.upload(file_path, '/assets/user_filehash/upload/%s/benign' % root_app_scope_name)
```

Suppression du condensé de fichier par l'utilisateur

Ce point terminal est utilisé pour charger un fichier CSV pour supprimer les condensés de fichiers de la portée racine sur l'appareil Cisco Secure Workload. Le fichier CSV doit avoir `FileHash` comme en-tête.

```
POST /openapi/v1/assets/user_filehash/delete/{rootAppScopeNameOrID}/{benignOrflagged}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeNameOrID	chaîne	Nom ou ID de la portée racine.
benignOrflagged	chaîne	Il peut s'agir de `benign (bénin)` ou `flagged` (marqué).

Objet de réponse : aucun

Exemple de code Python

```
  # Sample CSV File
  # FileHash
  # 1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29
  # 8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4

  file_path = '/<path_to_file>/user_filehash.csv'
  root_app_scope_name = 'Tetration'
```

```
  restclient.upload(file_path, '/assets/user_filehash/delete/' + root_app_scope_name +
'/benign')
```

Téléchargement du condensé de fichier par l'utilisateur

Ce point terminal renvoie le condensé du fichier utilisateur pour la portée racine donnée sur l'appareil Cisco Secure Workload sous forme de fichier CSV. Le fichier CSV comporte les en-têtes `HashType, FileHash, FileName` et `Notes` dans l'ordre respectif.

```
GET /openapi/v1/assets/user_filehash/download/{rootAppScopeNameOrID}/{benignOrflagged}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeNameOrID	chaîne	Nom ou ID de la portée racine.
benignOrflagged	chaîne	Peut être égal à benign (bénin) ou flagged (signalé).

Objet de réponse : aucun

Exemple de code Python

```
  file_path = '/<path_to_file>/output_user_filehash.csv'
  root_app_scope_name = 'Tetration'
  restclient.download(file_path, '/assets/user_filehash/download/%s/benign' %
root_app_scope_name)
```

Étiquettes définies par l'utilisateur

Ces API sont utilisées pour ajouter ou supprimer des étiquettes définies par l'utilisateur qui libellent les flux et les éléments d'inventaire sur l'appareil Cisco Secure Workload. Pour appeler ces API, utilisez une clé API avec la capacité `user_data_upload`. Consultez la section Schéma d'étiquette du guide de l'utilisateur de l'interface utilisateur pour connaître les directives régissant les clés et les valeurs utilisées pour l'étiquetage des flux et des éléments de l'inventaire.

Note Reportez-vous à la section Importation d'étiquettes personnalisées pour obtenir des instructions sur l'accès à cette fonctionnalité par l'interface utilisateur.

Note Reportez-vous aux limites des étiquettes pour connaître les limites du nombre d'adresses IPv4/IPv6 ou de sous-réseaux qui peuvent être téléversés.

API dépendantes de la portée

Les API suivantes sont utilisées pour obtenir, définir/supprimer les étiquettes dans une portée racine sur l'appareil Cisco Secure Workload. Elles sont mises à la disposition des **propriétaires de la portée** racine et

des **administrateurs de site**. En outre, les appels d'API GET sont disponibles pour les utilisateurs avec **un accès en lecture** à la portée racine.

Obtenir une étiquette d'inventaire

Ce point terminal renvoie des étiquettes pour une adresse IPv4/IPv6 ou un sous-réseau dans la portée racine sur l'appareil Cisco Secure Workload. L'adresse ou le sous-réseau utilisé pour interroger ce point terminal doivent correspondre exactement à la valeur utilisée pour le chargement des étiquettes.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}?ip={IPorSubnet}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.
IPorSubnet	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Objet de réponse :

Nom	Type	Description
attributes	JSON	Carte clé/valeur pour l'étiquetage des flux correspondants et des éléments de l'inventaire

Exemple de code Python

```
root_app_scope_name = 'Tetration'
restclient.get('/inventory/tags/%s' % root_app_scope_name, params={'ip': '10.1.1.1/24'})
```

Recherche d'étiquettes d'inventaire

Ce point terminal permet de rechercher des étiquettes pour une adresse IPv4/IPv6 ou un sous-réseau dans la portée racine sur l'appareil Cisco Secure Workload.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}/search?ip={IPorSubnet}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.
IPorSubnet	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Objet de réponse : cette API renvoie une liste d'objets au format suivant

Nom	Type	Description
key	chaîne	Adresse IPv4/IPv6 ou sous-réseau.
updatedAt	nombre entier	Horodatage Unix du moment où les étiquettes ont été mises à jour.

Nom	Type	Description
valeur	JSON	Carte clé/valeur des attributs de la clé.

Exemple de code Python

```
  root_app_scope_name = 'Tetration Scope'
  encoded_root_app_scope_name = urllib.quote(root_app_scope_name, safe='')
  restclient.get('/inventory/tags/%s/search' % encoded_root_app_scope_name, params={'ip':
'10.1.1.1/24'})
```

Définir une étiquette d'inventaire

Ce point terminal est utilisé pour définir des étiquettes pour les flux et les éléments de l'inventaire dans la portée racine sur l'appareil Cisco Secure Workload.

```
POST /openapi/v1/inventory/tags/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
ip	chaîne	Adresse IPv4/IPv6 ou sous-réseau.
attributes	JSON	Carte clé/valeur pour l'étiquetage des flux correspondants et des éléments de l'inventaire

Objet de réponse :

Nom	Type	Description
avertissements	JSON	Carte clé/valeur contenant les avertissements rencontrés lors de la définition des étiquettes.

Exemple de code Python

```
  root_app_scope_name = 'Tetration'
  req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}

  restclient.post('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

Supprimer les étiquettes d'inventaire

Ce point terminal supprime les étiquettes pour une adresse IPv4/IPv6 ou un sous-réseau dans une portée racine sur l'appareil Cisco Secure Workload.

```
DELETE /openapi/v1/inventory/tags/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
ip	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Exemple de code Python

```
  root_app_scope_name = 'Tetration'
  req_payload = {'ip': '10.1.1.1/24'}
  restclient.delete('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

Charger des étiquettes

Ce point terminal est utilisé pour charger un fichier CSV avec des étiquettes pour l'étiquetage des flux et des éléments d'inventaire dans un périmètre racine sur le l'appareil Cisco Secure Workload. Un en-tête de colonne avec le nom « `IP` » doit apparaître dans le fichier CSV. Sur les autres en-têtes de colonne, jusqu'à 32 de ces derniers peuvent être utilisés pour annoter les flux et les éléments de l'inventaire. Pour utiliser des caractères non latins dans les étiquettes, le fichier CSV téléversé doit être au format UTF-8.

```
POST /openapi/v1/assets/cmdb/upload/{rootAppScopeName}
```

Paramètres :

L'utilisateur doit fournir un type d'opération `(X-Tetration-Oper)` en tant que paramètre de cette API. `X-Tetration-Oper` peut être l'un des éléments suivants :

- add (ajouter) : ajoute des étiquettes aux adresses ou aux sous-réseaux nouveaux et existants. Résout les conflits en sélectionnant de nouvelles étiquettes par rapport à celles existantes. Par exemple, si les étiquettes d'une adresse dans la base de données sont *{« foo » : « 1 », « bar » : « 2 »}*et que le fichier CSV contient *{« z » : « 1 », « bar » : « 3 »}*, *Add (Ajouter) définit les* étiquettes pour cette adresse sur *{« foo » : « 1 », « z » : « 1 », « bar » : « 3 »}*.
- overwrite (remplacer) : insère des étiquettes pour les nouvelles adresses/sous-réseaux et remplace les étiquettes pour les adresses existantes. Par exemple, si les étiquettes d'une adresse dans la base de données sont *{« foo » : « 1 », « bar » : « 2 »}* et que le fichier CSV contient *{« z » : « 1 », « bar » : « 3 » }*, *overwrite (remplacer)* définit les étiquettes pour cette adresse sur *{« z » : « 1 », « bar » : « 3 »}*.
- merge (fusionner) : fusionne les étiquettes avec les adresses ou les sous-réseaux existants. Résout les conflits en sélectionnant des valeurs non vides prioritairement aux valeurs vides. Par exemple, si les étiquettes d'une adresse dans la base de données sont *{« foo »: « 1 », « bar » : « 2 », « qux » : « », « corge » : « 4 »}*et que le fichier CSV contient *{« z » : « 1 », « bar » : « , « qux » : « 3 », « corge » : « 4-updated »}*, *merge (fusionner)* définit des étiquettes pour cette adresse à *{« foo » : « 1 , « z » : « 1 », « bar » : « 2 », « qux » : « 3 », « corge » : « 4-updated »}* .

Note La valeur de « bar » dans n'est pas réinitialisée à « » (vide), au lieu de cela, la valeur existante de « bar » = « 2 » est conservée.

- delete (supprimer) : supprime les étiquettes pour une adresse ou un sous-réseau.

Objet de réponse :

Nom	Type	Description
avertissements	JSON	Carte clé/valeur contenant les avertissements rencontrés lors de la définition des étiquettes.

Exemple de code Python

```
file_path = '/<path_to_file>/user_annotations.csv'
root_app_scope_name = 'Tetration'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload/%s' % root_app_scope_name, req_payload)
```

Charger des étiquettes au format JSON

Ce point terminal est utilisé pour charger des étiquettes pour les flux d'étiquetage et les éléments de l'inventaire sur l'appareil Cisco Secure Workload au format JSON. Les attributs dans ip_tags doivent être un sous-ensemble d'en-têtes. Jusqu'à 32 en-têtes peuvent être utilisés pour annoter les flux et les éléments de l'inventaire. Pour utiliser des caractères non latins dans les étiquettes, la charge utile json doit être au format UTF-8.

```
POST /openapi/v1/multi_inventory/tags/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient le paramètre suivant

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine

Table 10: Paramètres de la charge utile

Nom	Type	Description
headers	tableau	Tableau de chaînes de caractères spécifiant les noms des étiquettes
operation	chaîne	Soit **add** (ajouter), soit **merge** (fusionner), si ce n'est pas spécifié, l'opération est considérée comme **add**.
ip_tags	tableau	Tableau d'objets ip_tags

Table 11: ip_tags

Nom	Type	Description
ip	chaîne	Adresse IPV4/IPV6 ou sous-réseau valide
attributes	JSON	JSON de paires d'étiquettes et de chaînes de valeurs; les étiquettes doivent être un sous-ensemble d'en-têtes.

Note

1. Si les étiquettes d'enregistrement ne sont pas un sous-ensemble d'en-têtes donné, un avertissement est généré pour alerter l'utilisateur d'une différence entre les en-têtes et les étiquettes de l'enregistrement.

2. Ce point terminal ne permettra à l'utilisateur de télécharger qu'un maximum de 95 000 entrées et 36 attributs.

- Pour les demandes d'ajout, le paramètre *operation* (opération) est facultatif. S'il n'est pas spécifié, il est considéré comme **add** (ajouter).
- Pour les demandes de fusion, *operation* doit être spécifiée sous la forme **merge**.

3. Vous pouvez télécharger un maximum de 95 000 entrées et 36 attributs.

Exemple de code Python pour la demande d'ajout

```
{
  "headers" : ["key1", "key2"],
  "operation": "add"
  "ip_tags" : [
      {
          "ip": "10.10.10.11",
          "attributes": {
              "key1": "val1",
              "key2": "val2"
          }
      },
      {
          "ip": "10.10.10.12",
          "attributes": {
              "key1": "val111",
              "key2": "val2"
          }
      }
  ]
}
```

Exemple de code Python pour la demande de fusion

```
{{
  "headers" : ["key1", "key2"],
  "operation": "merge"
  "ip_tags" : [
      {
          "ip": "10.10.10.11",
          "attributes": {
```

```
                "key1": "val1",
                "key2": "val2"
            }
        },
        {
            "ip": "10.10.10.12",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        }
    ]
}
resp = restclient.post('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))
```

Télécharger des étiquettes utilisateur

Ce point terminal renvoie les étiquettes téléversées par l'utilisateur pour une portée racine sur l'appareil Cisco Secure Workload sous forme de fichier CSV.

```
GET /openapi/v1/assets/cmdb/download/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Réponse :

Type de contenu : *text/csv*

Fichier CSV contenant les étiquettes téléversées par l'utilisateur pour la portée.

Exemple de code Python

```
file_path = '/<path_to_file>/output.csv'
root_app_scope_name = 'Tetration'
restclient.download(file_path, '/assets/cmdb/download/%s' % root_app_scope_name)
```

Obtenir les en-têtes de colonne

Ce point terminal renvoie une liste d'en-têtes de colonne pour une portée racine sur Cisco Secure Workload.

```
GET /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Objet de réponse : un tableau des aspects disponibles pour une étiquette.

Exemple de code Python

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/attributenames/%s' % root_app_scope_name)
```

Supprimer l'en-tête de colonne

Ce point terminal supprime un en-tête de colonne dans une portée racine sur l'appareil Cisco Secure Workload. La suppression d'un en-tête de colonne le supprime de la liste des aspects étiquetés et le supprime des étiquettes existantes.

```
DELETE /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}/{attributeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.
attributeName	chaîne	Attribut en cours de suppression

Objet de réponse : aucun

Exemple de code Python

```
   root_app_scope_name = 'Tetration'
   attribute_name = 'column1'
   resp = restclient.delete('/assets/cmdb/attributenames/%s/%s' % (root_app_scope_name,
attribute_name))
```

Supprimer des étiquettes au format JSON

Ce point terminal supprime les étiquettes pour plusieurs adresses IPv4/IPv6 ou plusieurs sous-réseaux à l'aide du format JSON.

```
DELETE /openapi/v1/multi_inventory/tags/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine

Table 12: Paramètres de la charge utile

Nom	Type	Description
headers	tableau	(Facultatif) tableau de chaînes de caractères spécifiant les noms des étiquettes
ip_tags	tableau	Tableau d'objets ip_tags

Table 13: ip_tags

Nom	Type	Description
ip	chaîne	Adresse IPV4/IPV6 ou sous-réseau valide

Nom	Type	Description
attributes	JSON	(Facultatif) JSON de paires d'étiquettes et de chaînes de valeurs; les étiquettes doivent être un sous-ensemble d'en-têtes.

Exemple de code Python

```
  {
    "ip_tags" : [
        {
            "ip": "10.10.10.11",
          },
        {
            "ip": "10.10.10.12",
            "attributes": {
                "key1": "val1",
                "key2": "val2"
            }
        }
    ]
}
resp = restclient.delete('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))
```

Obtenir la liste des aspects étiquetés

Ce point terminal renvoie une liste d'aspects étiquetés pour une portée racine sur l'appareil Cisco Secure Workload. Les aspects étiquetés sont un sous-ensemble d'en-têtes de colonne dans le fichier CSV téléversé qui est utilisé pour annoter les flux et les éléments de l'inventaire dans cette portée.

```
GET /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Objet de réponse : tableau d'aspects étiquetés pour la portée racine.

Exemple de code Python

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/annotations/%s' % root_app_scope_name)
```

Mettre à jour la liste des aspects étiquetés

Ce point terminal met à jour la liste des aspects utilisés pour l'annotation des flux et des éléments de l'inventaire dans une portée racine sur l'appareil Cisco Secure Workload.

```
PUT /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Objet de réponse : aucun

Exemple de code Python

```
# the following list is a subset of column headers in the
# uploaded CSV file
req_payload = ['location', 'region', 'detail']
root_app_scope_name = 'Tetration'
restclient.put('/assets/cmdb/annotations/%s' % root_app_scope_name,
               json_body=json.dumps(req_payload))
```

Purger les étiquettes téléversées par l'utilisateur

Ce point terminal purge les étiquettes des flux et des éléments d'inventaire dans la portée racine sur l'appareil Cisco Secure Workload. Les modifications affectent les nouvelles données; les anciennes données étiquetées demeurent inchangées.

```
POST /openapi/v1/assets/cmdb/flush/{rootAppScopeName}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
rootAppScopeName	chaîne	Nom de la portée racine.

Objet de réponse : aucun

Exemple de code Python

```
root_app_scope_name = 'Tetration'
restclient.post('/assets/cmdb/flush/%s' % root_app_scope_name)
```

API indépendantes de la portée

Les API suivantes peuvent s'étendre sur plusieurs portées sur l'appareil Cisco Secure Workload.

Note Le nombre de facettes annotées indépendantes et dépendantes de la portée ne doit pas dépasser 32 pour toute portée racine.

Charger des étiquettes

Ce point terminal est utilisé pour charger un fichier CSV avec des étiquettes pour les flux et les éléments d'inventaire sur l'appareil Cisco Secure Workload. Les en-têtes de colonne avec les noms `IP` et `VRF` doivent apparaître dans le fichier CSV, et `VRF` doit correspondre à la portée racine pour une étiquette. Sur les autres en-têtes de colonne, jusqu'à 32 de ces derniers peuvent être utilisés pour annoter les flux et les éléments de l'inventaire.

```
POST /openapi/v1/assets/cmdb/upload
```

Paramètres :

L'utilisateur doit fournir un type d'opération `(X-Tetration-Oper)` comme paramètre à cette API pour préciser l'opération à effectuer.

Objet de réponse :

Nom	Type	Description
avertissements	JSON	Matrice de clés ou de valeurs contenant les avertissements rencontrés lors de la définition des étiquettes.

Exemple de code Python

```
file_path = '/<path_to_file>/user_annotations.csv'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload', req_payload)
```

Télécharger des étiquettes utilisateur

Ce point terminal renvoie les étiquettes téléversées par l'utilisateur pour toutes les portées sur l'appareil Cisco Secure Workload dans un fichier CSV.

```
GET /openapi/v1/assets/cmdb/download
```

Paramètres : Aucun

Réponse :

Type de contenu : *text/csv*

Fichier CSV contenant les étiquettes téléversées par l'utilisateur pour la portée.

Exemple de code Python

```
file_path = '/<path_to_file>/output.csv'
restclient.download(file_path, '/assets/cmdb/download')
```

Étiquettes indépendantes de la portée

Ces étiquettes ne sont pas liées à une portée racine particulière et s'appliquent à toutes les portées sur l'appareil.

Obtenir une étiquette d'inventaire

Ces points terminaux renvoient des étiquettes indépendantes de la portée pour une adresse IPv4/IPv6 ou un sous-réseau sur l'appareil Cisco Secure Workload. L'adresse ou le sous-réseau utilisé pour interroger ce point terminal doit correspondre exactement à ceux utilisés pour le téléversement des étiquettes.

```
GET /openapi/v1/si_inventory/tags?ip={IPorSubnet}
```

Paramètres : L'URL de la demande contient les paramètres suivants.

Nom	Type	Description
IPorSubnet	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Objet de réponse :

Nom	Type	Description
attributes	JSON	Matrice de clés ou de valeurs pour l'étiquetage des flux et des éléments de l'inventaire correspondants

Exemple de code Python

```
restclient.get('/si_inventory/tags', params={'ip': '10.1.1.1/24'})
```

Recherche d'étiquettes d'inventaire

Ce point terminal permet de rechercher des étiquettes pour une adresse IPv4/IPv6 ou un sous-réseau sur l'appareil Cisco Secure Workload.

```
GET /openapi/v1/si_inventory/TAGs/search?ip= {IPorSubnet}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
IPorSubnet	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Objet de réponse : cette API renvoie une liste d'objets au format suivant

Nom	Type	Description
key	chaîne	Adresse IPv4/IPv6 ou sous-réseau.
updatedAt	nombre entier	Horodatage Unix de la mise à jour des étiquettes.
valeur	JSON	Carte de clé ou de valeur des attributs de la clé.

Exemple de code Python

```
restclient.get('/si_inventory/tags/search', params={'ip': '10.1.1.1/24'})
```

Définir une étiquette d'inventaire

Ce point terminal est utilisé pour définir des étiquettes pour les flux et les articles de l'inventaire sur l'appareil Cisco Secure Workload.

```
POST /openapi/v1/si_inventory/tags
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
ip	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Nom	Type	Description
attributes	JSON	Matrice de clés ou de valeurs pour l'étiquetage des flux et des éléments de l'inventaire correspondants.

Objet de réponse :

Nom	Type	Description
avertissements	JSON	Matrice de clés ou de valeurs contenant les avertissements rencontrés lors de la définition des étiquettes.

Exemple de code Python

```
req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}

restclient.post('/si_inventory/tags', json_body=json.dumps(req_payload))
```

Supprimer les étiquettes d'inventaire

Ce point terminal supprime les étiquettes pour une adresse IPv4/IPv6 ou un sous-réseau sur l'appareil Cisco Secure Workload.

```
DELETE /openapi/v1/si_inventory/tags
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
ip	chaîne	Adresse IPv4/IPv6 ou sous-réseau.

Exemple de code Python

```
req_payload = {'ip': '10.1.1.1/24'}
restclient.delete('/si_inventory/tags, json_body=json.dumps(req_payload))
```

Obtenir la liste des aspects étiquetés

Ce point terminal renvoie une liste d'attributs étiquetés indépendants de la portée sur l'appareil Cisco Secure Workload. Les aspects étiquetés sont un sous-ensemble d'en-têtes de colonnes utilisés pour annoter les flux et les éléments de l'inventaire dans tous les portées.

Note Excluez le nom de la portée de l'URL de la demande pour afficher et mettre à jour la liste des aspects annotés indépendamment de la portée.

```
GET /openapi/v1/assets/cmdb/annotations
```

Objet de réponse : un tableau de aspects étiquetés indépendants de la portée.

Exemple de code Python

```
resp = restclient.get('/assets/cmdb/annotations')
```

Mettre à jour la liste des aspects étiquetés

Ce point terminal met à jour la liste des aspects indépendants de la portée qui sont utilisés pour annoter les flux et les éléments de l'inventaire sur l'appareil Cisco Secure Workload.

```
PUT /openapi/v1/assets/cmdb/annotations
```

Objet de réponse : aucun

Exemple de code Python

```
# the following list is a subset of column headers in the
# uploaded CSV file
req_payload = ['location', 'region', 'detail']
restclient.put('/assets/cmdb/annotations',
               json_body=json.dumps(req_payload))
```

Routage et transfert virtuels

Cet ensemble d'API gère l'instance virtuelle de routage et de transmission (VRF)

Note Ces API ne sont disponibles que pour les administrateurs de site.

Objet VRF

Les attributs de l'objet VRF sont décrits ci-dessous :

Attribut	Type	Description
ID	int	Identifiant unique du VRF
name	chaîne	Nom spécifié par l'utilisateur du VRF.
tenant_id	int	ID du détenteur parent.
root_app_scope_id	chaîne	ID de la portée racine associée.
created_at	nombre entier	Horodatage Unix lors de la création du VRF.
updated_at	nombre entier	Horodatage Unix de la dernière mise à jour du VRF.

Obtenir des VRF

Ce point terminal renvoie une liste de VRF. Cette API est disponible pour les clés API avec la capacité `sensor_management ou flow_inventory_query`.

```
GET /openapi/v1/vrfs
```

Paramètres : Aucun

Objet de réponse : renvoie une liste des objets VRF.

Exemple de code Python

```
resp = restclient.get('/vrfs')
```

Créer un VRF

Ce point terminal est utilisé pour créer de nouveaux VRF. Une portée racine associée sera automatiquement créée avec une requête correspondant à l'ID du VRF. Cette API est disponible pour les clés API avec la capacité `sensor_management`.

```
POST /openapi/v1/vrfs
```

Paramètres :

Nom	Type	Description
ID	int	(Facultatif) Identifiant unique pour le VRF. Si elle n'est pas spécifiée, la grappe Cisco Secure Workload générera un ID unique pour le VRF nouvellement créé. La bonne pratique est de laisser Cisco Secure Workload générer ces ID au lieu que l'appelant spécifie explicitement des ID uniques.
tenant_id	int	(facultatif) ID du détenteur parent.
name	chaîne	Nom spécifié par l'utilisateur du VRF.
apply_monitoring_rules	booléen	(Facultatif) Indique si des règles de collecte doivent être appliquées pour le VRF. La valeur par défaut est « faux » (faux).

`tenant_id` est facultatif. S'il n'est pas fourni, le VRF sera ajouté au détenteur avec le même ID que le VRF, et il sera créé automatiquement si nécessaire. Si `tenant_id` est fourni, le locataire ne sera pas créé automatiquement et une erreur sera renvoyée s'il n'existe pas.

Objet de réponse : renvoie l'objet VRF nouvellement créé.

Exemple de code Python

```
req_payload = {
    "tenant_id": <tenant_id>,
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.post('/vrfs', json_body=json.dumps(req_payload))
```

Obtenir un VRF spécifique

Ce point terminal renvoie des informations concernant l'ID de VRF spécifié. Cette API est disponible pour les clés API avec la capacité `sensor_management ou flow_inventory_query`.

```
GET /openapi/v1/vrfs/{vrf_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
vrf_id	int	Identifiant unique du VRF

Objet de réponse : renvoie un objet VRF associé à l'ID spécifié.

Exemple de code Python

```
vrf_id = 676767
resp = restclient.get('/vrfs/%d'% vrf_id)
```

Mettre à jour un VRF

Ce point terminal met à jour un VRF. Cette API est disponible pour les clés API avec la capacité `sensor_management`.

```
PUT /openapi/v1/vrfs/{vrf_id}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
vrf_id	int	Identifiant unique du VRF

Le corps de la requête JSON contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Nom spécifié par l'utilisateur du VRF.
apply_monitoring_rules	booléen	(Facultatif) Indique si les règles de collecte doivent être appliquées au VRF.

Objet de réponse : renvoie l'objet VRF modifié associé à l'ID spécifié.

Exemple de code Python

```
vrf_id = 676767
req_payload = {
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.put('/vrfs/%d'% vrf_id,
                      json_body=json.dumps(req_payload))
```

Supprimer un VRF spécifique

Ce point terminal supprime un VRF. Il échoue si une portée racine est associée. Cette API est disponible pour les clés API avec la capacité `sensor_management`.

```
DELETE /openapi/v1/vrfs/{vrf_id}
```

Paramètres : le paramètre suivant fait partie de l'URL.

Nom	Type	Description
vrf_id	int	Identifiant unique du VRF

Exemple de code Python

```
vrf_id = 676767
resp = restclient.delete('/vrfs/%d'% vrf_id)
```

Orchestrateurs

Cet ensemble d'API peut être utilisé pour gérer l'apprentissage par inventaire de l'orchestrateur externe dans le déploiement de grappes Cisco Secure Workload. Elles nécessitent la capacité `external_integration` associée à la clé API.

Les types d'orchestrateurs actuellement pris en charge sont « vcenter » (vCenter 6.5 et versions ultérieures), « kubernetes », « dns », « f5 », « netscaler », « infoblox » et « Cisco FMC ». L'interface utilisateur prise en charge se trouve dans Orchestrateurs externes.

Objet orchestrateur

Les attributs de l'objet orchestrateur sont décrits ci-dessous : certains des champs ne s'appliquent qu'à des types d'orchestrateurs spécifiques; les restrictions sont mentionnées dans le tableau ci-dessous.

Attribut	Type	Description
ID	chaîne	Identifiant unique de l'orchestrateur.
name	chaîne	Nom de l'orchestrateur spécifié par l'utilisateur.

Attribut	Type	Description
type	chaîne	Type d'orchestrateur : valeurs prises en charge (*vcenter, KUbernetes, F5, netscaler, infoblox, DNS)*
description	chaîne	Description de l'orchestrateur précisée par l'utilisateur.
username	chaîne	Nom d'utilisateur pour le point terminal de l'orchestration. (inutile pour *DNS*)
password	chaîne	Mot de passe du point terminal de l'orchestration. (inutile pour *DNS*)
certificate	chaîne	Certificat client utilisé pour l'authentification (inutile pour *DNS*)
key	chaîne	Clé correspondant au certificat client (inutile pour *DNS*)
ca_certificate	chaîne	Certificat de l'autorité de certification pour valider le point terminal de l'orchestration (inutile pour *DNS*)
auth_token	chaîne	Jeton d'authentification opaque (jeton du porteur) (s'applique uniquement à *Kubernetes*)
insecure	booléen	Désactiver la vérification SSL stricte.
delta_interval	nombre entier	Délai d'interrogation de l'intervalle en secondes. Le gestionnaire d'inventaire Cisco Secure Workload effectuera une interrogation pour les modifications incrémentielles toutes les delta-interval secondes. Notez que ce paramètre ne s'applique pas à Infoblox et à Cisco Secure Firewall Management Center.
full_snapshot_interval	nombre entier	Intervalle de l'instantané complet en secondes. Le gestionnaire d'inventaire Cisco Secure Workload effectuera une interrogation d'actualisation complète à partir de l'orchestrateur

Attribut	Type	Description
verbose_tsdb_metrics	booléen	Mesures de la TSDB par point terminal
hosts_list	Tableau	Tableau de paires { « host_name », « port_number »,} (nom de l'hôte, numéro de port) qui précisent comment Cisco Secure Workload doit se connecter à l'orchestrateur
use_secureconnector_tunnel	booléen	Connexions de tunnel vers les hôtes de cet orchestrateur par l'intermédiaire du tunnel du connecteur sécurisé
route_domain	nombre entier	Numéro de domaine de routage à interroger sur les équilibreurs de charge F5 (s'applique uniquement à *F5*)
dns_zones	Tableau	Tableau de chaînes contenant les zones DNS à interroger à partir du serveur DNS (uniquement pour *DNS*). Chaque entrée de zone DNS DOIT se terminer ainsi.
enable_enforcement	booléen	Applicable uniquement aux orchestrateurs externes avec prise en charge de l'application de la politique, tels que les pare-feu et les équilibreurs de charge. Quelques exemples : *Cisco Secure Firewall Management Center*, *F5 BIGIP* et *Citrix Netscaler*. Cet indicateur prend la valeur « faux » (l'application des politiques est désactivée) par défaut. Si la valeur est « vrai », l'orchestrateur externe déploiera les politiques sur le dispositif d'équilibreur de charge donné lors de l'application des politiques pour l'espace de travail.
ingress_controllers	objet	Tableau des objets Contrôleur d'entrée

Attribut	Type	Description
fmc_enforcement_mode	chaîne	Applicable uniquement à *l'orchestrateur externe de Cisco Secure Firewall Management Center* et doit être soit *merge (fusion)* (par défaut) soit *override (remplacer).* La première instance indique au responsable de l'application des politiques de Cisco Secure Firewall Management Center de positionner toutes les règles de politique Cisco Secure Workload avant les règles de préfiltre existantes, tandis que la dernière instance supprimera toutes les règles de préfiltre créées par les utilisateurs.
infoblox_config	objet	Applicable uniquement à *l'orchestrateur externe Infoblox*. Sélecteurs de type d'enregistrement de Configuration Infoblox.

Contrôleur d'entrée

Attribut	Type	Description
pod_selector	objet	Sélecteur de Pod
controller_config	objet	Configuration du contrôleur

Sélecteur de Pod

Attribut	Type	Description
namespace	chaîne	L'espace de noms dans lequel le pod de contrôleur d'entrée est en cours d'exécution.
labels	Tableau	Tableau de paires {« clé », « valeur »} qui précisent les étiquettes des pods de contrôleur d'entrée.

Configuration du contrôleur

Attribut	Type	Description
ingress_class (classe_entrée)	chaîne	Nom de la classe d'entrée que le contrôleur d'entrée satisfait.
namespace	chaîne	Namespace (espace de noms) est le nom de l'espace de noms que le contrôleur d'entrée satisfait.
(http_ports) ports_http	Tableau	Tableau des ports http.
(https_ports) ports_https	Tableau	Tableau des ports https.

Configuration Infoblox

enable_network_record	booléen	La valeur par défaut est vrai. Si la valeur est Faux (False), les enregistrements du type *réseau* sont désactivés.
enable_host_record	booléen	La valeur par défaut est vrai. Si la valeur est Faux (False), les enregistrements du type *hôte* sont désactivés
enable_a_record	booléen	La valeur par défaut est vrai. Si la valeur est Faux (False), les enregistrements de type *a* sont désactivés.
enable_aaaa_record	booléen	La valeur par défaut est vrai. Si la valeur est Faux (False), les enregistrements de type *aaaa* sont désactivés.

** Champs d'état en lecture seule dans l'objet orchestrateur **

Attribut	Type	Description
authentication_failure	booléen	État de la connexion à l'orchestrateur Cisco Secure Workload – « *vrai* » indique une connexion réussie à l'orchestrateur. Si ce champ est *faux*, le champ *authentication_failure_error* fournira un message d'erreur détaillé expliquant la raison de l'échec d'authentification.

Attribut	Type	Description
authentication_failure_error	chaîne	Message d'erreur détaillé pour aider à déboguer les échecs de connectivité ou d'identifiants d'authentification avec les orchestrateurs
scope_id	chaîne	ID de la portée racine du détenteur où l'inventaire sera publié et visible

Obtenir des orchestrateurs

Ce point terminal renvoie une liste des orchestrateurs connus de l'appareil Cisco Secure Workload. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/orchestrator/{scope}
```

Paramètres : Aucun

Renvoie la liste des objets orchestrateur pour la portée racine fournie. La *portée* DOIT être un ID de portée racine.

Créer des orchestrateurs

Ce point terminal est utilisé pour créer des orchestrateurs.

```
POST /openapi/v1/orchestrator/{scope}
```

Exemple de code python pour les orchestrateurs vCenter

```
req_payload = {
    "name": "VCenter Orchestrator"
    "type": "vcenter",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 443}],
    "username":"admin",
    "password":"admin"
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Exemple de code python pour les orchestrateurs DNS

```
req_payload = {
    "name": "DNS Server"
    "type": "dns",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "dns_zones": [ "lab.corp.com.", "dev.corp.com." ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Exemple de code Python pour des orchestrateurs Kubernetes

```
req_payload = {
    "name": "k8s"
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
```

```
    "certificate": "",
    "key": "",
    "ca_certificate": "",
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Exemple de code python pour des orchestrateurs Kubernetes avec contrôleur d'entrée

Consultez les renseignements sur l'orchestrateur externe Kubernetes/OpenShift pour créer des renseignements détaillés d'authentification.

```
req_payload = {
    "name": "k8s",
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
    "ingress_controllers": [
        {
            "pod_selector": {
                "namespace": "ingress-nginx",
                "labels": [{ "key": "app", "value": "nginx-ingress"}],
            }
        }
    ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Exemple de code python pour des orchestrateurs Kubernetes avec plusieurs contrôleurs d'entrée

Consultez les renseignements sur l'orchestrateur externe Kubernetes/OpenShift pour créer des renseignements détaillés d'authentification.

```
req_payload = {
    "name": "k8s",
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
    "ingress_controllers": [
        {
            "pod_selector": {
                "namespace": "ingress-nginx",
                "labels": [{ "key": "app", "value": "nginx-ingress"}],
            },
            "controller_config": {
                "ingress_class": "nginx-class",
            }
        },
        {
            "pod_selector": {
                "namespace": "ingress-haproxy",
                "labels": [{ "key": "app", "value": "haproxy-ingress"}],
            },
            "controller_config": {
                "ingress_class": "haproxy-class",
                "http_ports": [8080],
                "https_ports": [8443],
                "namespace": "haproxy-watching-namespace"
            }
        }
```

```
    ],
  }
  resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))

  ** Type AWS and EKS are no longer supported in external orchestrators. They have been
ported to
    connectors.
```

Obtenir un orchestrateur spécifique

Ce point terminal renvoie une instance d'orchestrateur.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Renvoie l'objet orchestrateur associé à l'ID spécifié.

Mettre à jour un orchestrateur

Ce point terminal met à jour un orchestrateur.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Paramètres :

Identiques aux paramètres de POST

Supprimer un orchestrateur spécifique

Ce point terminal supprime l'orchestrateur spécifié.

```
DELETE /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Règles d'or de l'orchestrateur

Cet ensemble d'API peut être utilisé pour gérer les règles d'or pour les orchestrateurs Kubernetes externes. Des règles d'or sont nécessaires pour assurer la connectivité du plan de contrôle de Kubernetes en mode d'application de liste verte. Elles nécessitent la capacité `external_integration` associée à la clé API.

Le type d'orchestrateur actuellement pris en charge pour les règles d'or est uniquement « Kubernetes ». Les requêtes vers ce point terminal pour les orchestrateurs autres que Kubernetes échoueront.

Objet règles d'or de l'orchestrateur

Les attributs de l'objet Orchestrateur sont décrits ci-dessous :

Attribut	Type	Description
kubelet_port	nombre entier	Port d'API local au nœud de Kubelet
services	Tableau	Tableau d'objets des services Kubernetes

Obtenir les règles d'or de l'orchestrateur

Ce point terminal renvoie les règles d'or qui sont associées à un orchestrateur. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/orchestrator/{scope}/{id}/gr
```

Paramètres : Aucun

Renvoie un seul objet Règles d'or

Créer ou mettre à jour des règles d'or

Ce point terminal est utilisé pour créer ou mettre à jour les règles d'or pour un orchestrateur existant.

```
POST /openapi/v1/orchestrator/{scope}/{id}/gr
```

Paramètres :

Attribut	Type	Description
kubelet_port	nombre entier	Port d'API local au nœud de Kubelet
services	Tableau	Tableau d'objets des services Kubernetes

Exemple de code Python

```
  req_payload = {
              "kubelet_port":10255,
              "services": [
                          {
    "description": "kube-dns",
    "addresses":   [ "10.0.1.1:53/TCP", "10.0.1.1:53/UDP" ],
    "consumed_by":  [ "NODES", "PODS"],
   }
             ]
  }
  resp = restclient.post('/orchestrator/{scope_id}/{orchestrator_id}/gr',
json_body=json.dumps(req_payload))
```

Domaines FMC Orchestrator

Cet ensemble d'API peut être utilisé pour gérer les domaines pour les orchestrateurs FMC externes. Les domaines FMC sont nécessaires pour activer l'application sur un domaine FMC donné. Elles nécessitent la capacité `external_integration` associée à la clé API.

Le type d'orchestrateur actuellement pris en charge pour les domaines FMC est uniquement « fmc ». Les requêtes vers ce point terminal pour les orchestrateurs autres que FMC échoueront.

Objet domaines FMC de l'orchestrateur

Les attributs de l'objet Orchestrateur sont décrits ci-dessous :

Attribut	Type	Description
fmc_domains	Tableau	Tableau d'objets de domaine FMC

Les attributs de l'objet Domaine FMC sont décrits ci-dessous :

Attribut	Type	Description
name	chaîne	Nom du domaine FMC
enforcement_enabled	booléen	Cet indicateur est défini sur False par défaut. Si la valeur est « vrai », l'orchestrateur externe déploiera les politiques sur le domaine correspondant à « name » lorsque l'application des politiques est effectuée pour l'espace de travail.

Les attributs d'URL sont décrits ci-dessous :

Attribut	Type	Description
scope	chaîne	Nom ou ID de la portée racine du détenteur où l'inventaire sera publié et visible
orchestrator_id	chaîne	ID d'orchestrateur de l'orchestrateur FMC

Obtenir les domaines FMC

Ce point terminal renvoie les domaines FMC qui sont configurés sur le FMC associé à un orchestrateur FMC. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Paramètres : Aucun

Renvoie un objet JSON avec une liste d'attributs d'objet de domaine FMC.

Mettre à jour la configuration de domaine FMC pour l'orchestrateur externe FMC

Ce point terminal met à jour les attributs de domaine FMC pour un orchestrateur externe FMC existant.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Paramètres :

Attribut	Type	Description
fmc_domains	Tableau	Tableau d'objets de domaine FMC

Les attributs de l'objet Domaine FMC sont décrits ci-dessous :

Attribut	Type	Description
name	chaîne	Nom du domaine FMC
enforcement_enabled	booléen	Cet indicateur est défini sur False par défaut. Si la valeur est « vrai », l'orchestrateur externe déploie des politiques sur le domaine correspondant à « name » lorsque l'application des politiques est effectuée pour l'espace de travail.

Les attributs d'URL sont décrits ci-dessous :

Attribut	Type	Description
scope	chaîne	Nom ou ID de la portée racine du détenteur où l'inventaire sera publié et visible
orchestrator_id	chaîne	ID d'orchestrateur de l'orchestrateur FMC

Exemple de code Python

```
  req_payload = {
        "fmc_domains":    [
        {
            "enforcement_enabled": False,
            "name": "Global/Eng"
        },
        {
            "enforcement_enabled": True,
            "name": "Global/Prod"
        }
    ]
  }
  resp = restclient.put('/orchestrator/{scope}/{orchestrator_id}/fmcdomains',
json_body=json.dumps(req_payload))
```

Considérations relatives au contrôle d'accès en fonction des rôles (RBAC)

L'accès aux orchestrateurs sous une portée racine nécessite que la clé API utilisée pour la demande dispose des privilèges requis. Tous les appels d'API de l'orchestrateur sont déterminés et nécessitent toujours l'ID de portée racine dans l'URL. Les orchestrateurs résident toujours au niveau de la portée racine et ne peuvent pas être créés dans des sous-portées. Les orchestrateurs créés (et l'inventaire pris en charge par ces orchestrateurs) dans une portée racine de détenteur spécifique sont non vues pour les autres détenteurs.

Dans le cas des équilibreurs de charge F5 qui peuvent avoir plusieurs domaines de routage configurés (vrfs), la logique de filtrage du domaine de routage du F5 analyse toutes les entités sur F5 sur toutes les partitions,

mais élimine les entités (services, bassins de paquets, pools et dorsaux) qui ne sont pas évaluées par rapport au domaine de routage spécifié dans le champ *route_domain* (domaine de routage) de l'orchestrateur F5.

Facteurs à prendre en considération concernant la haute disponibilité et le basculement

Le paramètre hosts_list permet la configuration de plusieurs adresses de serveur pour un orchestrateur. La logique de sélection du serveur Cisco Secure Workload dans le cas de plusieurs adresses de serveur varie pour chaque type d'orchestrateur.

Pour *vCenter, Kubernetes, DNS, F5, Netscaler, Infoblox*, la sélection se fait sur la base du premier point terminal intègre. Les connexions ne sont pas persistantes (sauf pour *Kubernetes*) et donc, à chaque période d'interrogation, Secure Connector Orchestrator Manager analyse les hôtes et interroge le premier point terminal intègre trouvé dans hosts_list. Pour *Kubernetes*, un canal d'événement persistant est maintenu et, en cas d'échec de la connexion, une analyse de tous les hôtes et une interrogation complète ultérieure seront effectuées à l'aide du prochain point terminal intègre.

Considérations relatives aux ressources RBAC pour Kubernetes

Le client Kubernetes tente de RECEVOIR/RÉPERTORIER/SURVEILLER les ressources suivantes.

Les informations d'authentification Kubernetes fournies doivent avoir un ensemble minimal de privilèges sur les ressources suivantes :

Ressources	Verbes
daemonsets	[obtenir la liste de surveillance]
déploiements	[obtenir la liste de surveillance]
points terminaux	[obtenir la liste de surveillance]
espaces de noms	[obtenir la liste de surveillance]
nodes	[obtenir la liste de surveillance]
pods	[obtenir la liste de surveillance]
jeux de répliques	[obtenir la liste de surveillance]
contrôleurs de duplication	[obtenir la liste de surveillance]
services	[obtenir la liste de surveillance]
statefulsets	[obtenir la liste de surveillance]
daemonsets.apps	[obtenir la liste de surveillance]
deployments.apps	[obtenir la liste de surveillance]
endpoints.apps	[obtenir la liste de surveillance]

Ressources	Verbes
namespaces.apps	[obtenir la liste de surveillance]
nodes.apps	[obtenir la liste de surveillance]
pods.apps	[obtenir la liste de surveillance]
replicasets.apps	[obtenir la liste de surveillance]
replicationcontrollers.apps	[obtenir la liste de surveillance]
services.apps	[obtenir la liste de surveillance]
statefulsets.apps	[obtenir la liste de surveillance]
daemonsets.extensions	[obtenir la liste de surveillance]
deployments.extensions	[obtenir la liste de surveillance]
endpoints.extensions	[obtenir la liste de surveillance]
namespaces.extensions	[obtenir la liste de surveillance]
nœuds.extensions	[obtenir la liste de surveillance]
pods.extensions	[obtenir la liste de surveillance]
Replicasets.extensions	[obtenir la liste de surveillance]
replicationcontrollers.extensions	[obtenir la liste de surveillance]
services.extensions	[obtenir la liste de surveillance]
statefulsets.extensions	[obtenir la liste de surveillance]

Renseignements sur le site

Cette API peut être utilisée pour obtenir des informations sur la grappe telles que l'état de la grappe, le type de grappe, les adresses IP externes et les courriels.

Note Cette API est uniquement disponible pour les utilisateurs administrateurs du site.

Obtenir des renseignements sur le site

Ce point terminal renvoie un objet JSON avec des informations sur le site de la grappe.

```
GET /openapi/v1/site_infos
```

Paramètres : Aucun

Objet de réponse : objet JSON avec des renseignements sur le site de la grappe

Exemple de code Python

```
resp = restclient.get('/site_infos')
```

Exemple de réponse

```
{
  "cluster_state": "Enabled till 2020-12-31 23:59:59 UTC",
  "cluster_uuid": "00000000-0000-0000-0000-000000000000",
  "site_bosun_email": "customer-support@company.com",
  "site_cluster_type": "physical",
  "site_external_ips": [
      "1.1.1.1",
      "1.1.1.2",
      ...
      "1.1.1.7"
  ],
  "site_name": "cluster_name",
  "site_sensor_vip_ip": "2.1.1.1",
  "site_ui_admin_email": "site-admin@company.com",
  "site_ui_fqdn": "cluster.company.com",
  "site_ui_primary_customer_support_email": "customer-support@company.com"
}
```

État de la grappe

Cette API peut être utilisée pour obtenir l'état de tous les serveurs physiques dans Cisco Secure Workload.

Note Cette API est uniquement disponible pour les utilisateurs administrateurs du site.

Obtenir l'état d'intégrité de la grappe

Ce point terminal renvoie un objet JSON avec des informations sur l'intégrité de la grappe.

`GET /openapi/v1/cluster_nodes`

Paramètres : Aucun

Objet de réponse : objet JSON avec des informations sur l'intégrité de la grappe

Exemple de code Python

```
resp = restclient.get('/cluster_nodes')
```

État du service

Cette API peut être utilisée pour obtenir l'intégrité de tous les services utilisés dans la grappe Cisco Secure Workload ainsi que leurs dépendances.

Note Cette API est uniquement disponible pour les utilisateurs administrateurs du site.

Obtenir l'état d'intégrité du service

Ce point terminal renvoie un objet JSON avec des informations sur l'intégrité du service.

```
GET /openapi/v1/service_status
```

Paramètres : Aucun

Objet de réponse : objet JSON avec des renseignements sur l'intégrité du service

Exemple de code Python

```
resp = restclient.get('/service_status')
```

Connecteur sécurisé

OpenAPI affiche les points terminaux permettant de gérer les fonctions du connecteur sécurisé. Ces points terminaux nécessitent que la capacité `external_integration` soit associée à la clé API.

Note Les API du connecteur sécurisé ne peuvent pas être utilisées au niveau du site. Elles ne peuvent être utilisées qu'au niveau de la portée racine.

Obtenir l'état

Ce point terminal renvoie l'état actuel du tunnel du connecteur sécurisé pour la portée racine spécifiée.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/status
```

```
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/status
```

L'autorisation en lecture (READ) pour la portée racine précisée est requise.

L'état renvoyé est un objet json avec le schéma suivant :

Clé	Type	Valeur
actif	booléen	Un tunnel de connexion sécurisée (Secure Connector) est actuellement actif
peer	chaîne	<ip> :<port> de l'extrémité du tunnel client connecteur sécurisé
start_time	int	Horodatage de démarrage du tunnel (heure d'origine en secondes)

Clé	Type	Valeur
last_heartbeat	int	Horodatage de la dernière pulsation du client (heure d'origine en secondes)

Obtenir un jeton

Ce point terminal renvoie un nouveau jeton à usage unique pour une durée limitée à utiliser pour démarrer un client connecteur sécurisé pour la portée racine spécifiée.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/token
```

```
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/token
```

L'autorisation OWNER (PROPRIÉTAIRE) pour la portée racine précisée est requise.

Le jeton renvoyé est une chaîne qui contient un jeton signé de manière cryptographique et valide pendant une heure. Un jeton valide ne peut être utilisé qu'une seule fois pour démarrer un client connecteur sécurisé.

Alterner les certificats

Ce point terminal force la création d'un nouveau certificat pour la portée racine spécifiée. Le nouveau certificat sera utilisé par le serveur de connecteur sécurisé et sera utilisé pour signer les demandes de signature de certificat des clients pour cette portée racine.

```
POST /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/rotate_certs?invalidate_old=
‹→{vrai|faux}
```

```
POST /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/rotate_certs?invalidate_old= ‹→{vrai|faux}
```

L'autorisation OWNER (PROPRIÉTAIRE) pour la portée racine précisée est requise.

Une fois que ce point terminal est appelé, la communication entre le client et le serveur pour cette portée racine passera immédiatement à l'utilisation du nouveau certificat.

Si *invalidate_old* est défini à False, les clients existants créent automatiquement une nouvelle paire de clés publique/privée et utilisent leurs certificats existants pour signer un nouveau certificat pour la nouvelle clé publique.

Si *invalidate_old* est défini avec True, le certificat existant est immédiatement invalidé. Les clients existants ne pourront pas se connecter au serveur et devront être redémarrés à l'aide d'un nouveau jeton. Consultez la section Déploiement du connecteur sécurisé pour de plus amples renseignements.

Analyse des vulnérabilités Kubernetes

Obtenir les registres Kubernetes utilisés pour le balayage sur les vulnérabilités des pods

Ce point terminal renvoie une liste de tous les registres Kubernetes affichés dans la grappe Cisco Secure Workload pour un VRF donné.

```
GET /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
root_scope_name_or_id	chaîne	Nom ou ID de la portée racine

Objet de réponse : renvoie un tableau d'objets de registre avec les attributs suivants :

Attribut	Type	Description
ID	chaîne	ID du registre
clusters	tableau	Tableau d'objets de grappe Kubernetes utilisant le registre
connection_status	chaîne	Définit l'état de la connexion au registre
credential_status	chaîne	État indiquant si les renseignements d'authentification sont fournis
last_scanned	Int64	Dernier analyseur à l'heure d'origine
URL	chaîne	URL du registre

Objet Grappe Kubernetes

Attribut	Type	Description
ID	chaîne	ID de la grappe Kubernetes
connector_id	chaîne	ID du connecteur utilisé pour intégrer la grappe Kubernetes
connector_type	chaîne	Type de connecteur utilisé pour intégrer la grappe Kubernetes
name	chaîne	Nom de la grappe Kubernetes

Exemple de code Python

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/registry' % root_app_scope_name)
```

Ajouter des informations d'authentification au registre Kubernetes

Ce point terminal vous permet d'ajouter des renseignements d'authentification sur le registre Kubernetes. Les renseignements d'authentification acceptés sont basés sur le type de registre.

Par exemple :

Type de registre : AWS; Type d'informations d'authentification acceptées : objet informations d'authentification aws_auth

Type de registre : OTHER; Type d'informations d'authentification acceptés : Objet d'authentification basic_auth

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry/{registry_id}
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
root_scope_name_or_id	chaîne	Nom ou ID de la portée racine
registry_id	chaîne	ID de registre Kubernetes
registry_credential	objet	Objet identifiants

Objet identifiants

Attribut	Type	Description
basic_auth	objet	Objet authentification de base
aws_auth	objet	Objet authentification AWS
azure_auth	objet	Objet authentification Azure
gcp_auth	objet	Objet authentification GCP

basic_auth object

Attribut	Type	Description
username	chaîne	Username
password	chaîne	Mot de passe

aws_auth object

Attribut	Type	Description
aws_access_key_id	chaîne	Clé d'accès aux renseignements d'authentification AWS
aws_secret_access_key	chaîne	Clé secrète d'accès aux renseignements d'authentification AWS

Objet Azur_auth :

Attribut	Type	Description
azure_tenant_id	chaîne	ID de détenteur Azure
azure_client_id	chaîne	ID du client Azure
azure_client_secret	chaîne	Code secret du client Azure

gcp_auth object:

Attribut	Type	Description
gcp_service_account	objet	Compte der service GCP

Exemple de code Python

```
root_app_scope_name = 'Tetration'
registry_id = `64cdc7a7362f57192dcc1625’
pay_load = {
        "registry_credential": {
            "basic_auth": {
                "username": "username",
                "password": "password",
            }
        }
    }
restclient.put('/kubernetes/%s/vulnerability_scanning/registry/%s' % root_app_scope_name,
registry_id, json_body=json.dumps(pay_load))
```

Obtenir les analyseurs de pods Kubernetes

Ce point terminal renvoie une liste de tous les analyseurs de pods Kubernetes affichés dans la grappe Cisco Secure Workload pour un VRF donné.

```
GET /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/scanner
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
root_scope_name_or_id	chaîne	Nom ou ID de la portée racine

Objet de réponse : renvoie un tableau d’objets de registre avec les attributs suivants :

Attribut	Type	Description
ID	chaîne	ID du balayage
kubernetes_cluster	objet	Ajouter des informations d’authentification au registre Kubernetes
health_status	chaîne	Définit l’état d’intégrité de l’analyseur
health_object	chaîne	Objet d’état d’intégrité
scanner_action	chaîne	Avertit si le balayage est ACTIVÉ ou DÉSACTIVÉ
name	chaîne	Nom de l’analyseur

Objet d’intégrité

Attribut	Type	Description
last_checkin	chaîne	Dernier signalement en heure d'origine
scanner_sensor_name	chaîne	Nom du nœud Kubernetes sur lequel l'analyseur est exécuté
scanner_sensor_uuid	chaîne	ID de l'agent exécuté sur le nœud Kubernetes
status	chaîne	Avertit si l'intégrité est signalée par l'analyseur

Exemple de code Python

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/scanner % root_app_scope_name)
```

Modifier la requête et l'action du filtre de l'analyseur

Ce point terminal vous permet de modifier la requête et l'action du filtre d'analyseur Kubernetes.

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/scanner/{scanner_id}
```

Paramètres : le corps de la requête JSON contient les clés suivantes :

Nom	Type	Description
rootAppScopeName	chaîne	Nom ou ID de la portée racine
scanner_id	chaîne	ID de l'analyseur Kubernetes
scanner_action	chaîne	Pour activer ou désactiver l'analyseur. Les valeurs attendues sont ENABLED (ACTIVÉ) ou DISABLED (DÉSACTIVÉ).
filter_query	objet	Valider une requête de filtre d'inventaire pour filtrer les pods pour l'analyse des vulnérabilités.

Exemple de code Python

```
root_app_scope_name = 'Tetration'
scanner_id = '64cdc7a7362f57192dcc1625'
pay_load = {
        "scanner_action": "ENABLED"
        "filter_query": {
            "type": "contains",
            "field": "user_orchestrator_system/pod_name",
            "value": "pod"
        }
}
restclient.put('/kubernetes/%s/vulnerability_scanning/scanner/%s' % root_app_scope_name,
```

```
scanner_id, json_body=json.dumps(pay_load))
```

État d'application des politiques des orchestrateurs externes

Cet ensemble d'API est utilisé pour fournir l'état d'application des politiques pour les orchestrateurs externes de l'équilibreur de charge tels que *F5 BIG-IP* ou *Citrix Netscaler*.

Note Pour utiliser ces API, vous devez avoir accès à la portée associée au VRF.

Obtenir l'état d'application des politiques de tous les orchestrateurs externes

Ce point terminal renvoie l'état d'application de la politique pour tous les orchestrateurs externes appartenant au VRF donné. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/tnp_policy_status/{vrfID}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
vrfID	nombre entier	ID VRF pour la portée racine.

Objet de réponse : renvoie une liste des politiques de réseau avec l'état `ENFORCED (appliqué)` ou `FAILED (échec)` ou `IGNORED (ignoré)`.

Exemple de code Python

```
vrf_id = 676767
restclient.get('/tnp_policy_status/%d' % vrf_id)
```

Obtenir l'état d'application des politiques pour un orchestrateur externe

Ce point terminal renvoie l'état d'application de la politique pour un orchestrateur externe appartenant au VRF donné. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/tnp_policy_status/{vrfID}/{orchestratorID}
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
vrfID	nombre entier	ID VRF pour la portée racine.
orchestratorID	chaîne	ID d'orchestrateur externe

Objet de réponse : renvoie une liste des politiques de réseau avec l'état `ENFORCED (appliqué)` ou `FAILED (échec)` ou `IGNORED (ignoré)`.

Exemple de code Python

```
vrf_id = 676767
orchestrator_id = '5ee3c991497d4f3b00f1ee07'
restclient.get('/tnp_policy_status/%d/%s' % (vrf_id, orchestrator_id))
```

Télécharger les certificats pour les surveilleurs de données et les collecteurs de données gérés

Cet ensemble d'API est utilisé pour télécharger les certificats pour les dérivateurs et les récepteurs de données gérées.

Note Pour utiliser ces API, vous devez avoir accès à la portée associée au VRF.

Obtenir la liste des surveilleurs de données gérés pour un ID VRF donné.

Ce point terminal renvoie une liste des dérivations de données gérées dans un VRF donné. Cette API est disponible pour les clés API avec la capacité `external_integration`.

`GET /openapi/v1/mdt/{vrfID}`

Paramètres : Aucun

Renvoie une liste des surveilleurs de données gérés avec des attributs tels que l'ID du surveilleur de données géré.

Télécharger des certificats de surveilleurs de données gérés pour un ID MDT donné

Ce point terminal est utilisé pour télécharger les certificats pour un ID de surveilleur de données gérées donné. L'ID MDT peut être obtenu en utilisant le point terminal **/openapi/v1/mdt/{vrfID}**, comme expliqué dans la documentation ci-dessus. Cette API est disponible pour les clés API avec la capacité `external_integration`.

`GET /openapi/v1/mdt/{vrfID}/{mdtID}/certs`

Paramètres :

Nom	Type	Description
Format	chaîne	Les formats de magasin de clés et de magasin de confiance. Valeurs : `jks`(valeur par défaut) ou `cert`

Renvoie un fichier tar.gz qui contient les fichiers suivants :

Pour le format `jks` : **truststore.jks, topic.txt, passphrase.txt, keystone.jks, kafkaBrokerIps.txt, consumer_name.txt, consumer_group_id.txt.**

Pour le format `jks` : **KafkaConsumerCA.cert, KafkaConsumerPrivateKey.key, kafkaCA.cert, kafkaBrokerIps.txt, topic.txt**

KafkaConsumerCA.cert est le fichier de certificat public et le fichier **KafkaConsumerPrivateKey.key** contient la clé privée. **kafkaCA.cert** a le certificat d'autorité de certification et **kafkaBrokerIps.txt** a la liste des adresses IP et des ports des brokers Kafka. sujet.txt porte le nom du sujet qui doit être utilisé pour récupérer les données MDT. **truststore.jks** et **keystone.jks** sont des fichiers de magasin de clés Java.

Obtenir la liste des collecteurs de données pour un ID VRF donné

Ce point terminal renvoie une liste de récepteurs de données dans un VRF donné. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/datasinks/{vrfID}
```

Paramètres : Aucun

Renvoie une liste de récepteurs de données avec des attributs tels que l'ID de récepteur de données.

Télécharger des certificats de collecteurs de données pour un ID donné.

Ce point terminal est utilisé pour télécharger les certificats pour un ID de collecteur de données donné. L'ID de collecteur de données peut être obtenu en utilisant le point terminal **/openapi/v1/datasinks/{vrfID}**, comme expliqué dans la documentation ci-dessus. Cette API est disponible pour les clés API avec la capacité `external_integration`.

```
GET /openapi/v1/datasinks/{vrfID}/{dsID}/certs
```

Paramètres : Aucun

Renvoie un fichier tar.gz qui contient les fichiers suivants :- **userCA.cert, userPrivateKey.key, intermediateCA.cert, kafkaCA.cert, kafkaBrokerIps.txt, topic.txt**.

userCA.cert est le fichier de certificat public et le fichier **KafkaConsumerPrivateKey.key** contient la clé privée. **intermediateCA. cert** et **kafkaCA.cert** possède le certificat d'autorité de certification pour l'autorité de certification intermédiaire et racine respectivement. **kafkaBrokerIps.txt** contient la liste des adresses IP et des ports des intermédiaires Kafka. **topic.txt** porte le nom du sujet qui doit être utilisé pour récupérer les données du collecteur de données datasink.

Journaux des modifications

Cette API fournit un accès en lecture aux éléments du journal des modifications. Cette API nécessite la capacité `user_role_scope_management` associée à la clé API.

Note Cette API est uniquement disponible pour les administrateurs de site et les propriétaires de portées racine.

Objet journal des modifications

Les descriptions des attributs d'objets du journal des modifications sont les suivantes :

Attribut	Type	Description
ID	chaîne	Identifiant unique de l'élément du journal des modifications.
association_chain	tableau d'objets	Liste des noms et des ID associés à cette modification.
scope	chaîne	Portée du changement (différente d'une portée Cisco Secure Workload.
action	chaîne	Action de modification.
détails	chaîne	Plus de détails sur les actions, lorsqu'ils sont disponibles.
created_at	nombre entier	Horodatage Unix de la création de l'élément du journal des modifications.
modifier	objet	Utilisateur responsable du changement.
modifié	objet	Champs et valeurs modifiés.
initial	objet	Champs et valeurs avant modification.
version	nombre entier	Identifiant de version

Rechercher

Ce point terminal renvoie la liste des éléments du journal des modifications correspondant aux critères spécifiés.

```
GET /openapi/v1/change_logs
```

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_app_scope_id	chaîne	(Facultatif) Requis pour les propriétaires de portée racine. Filtrer les résultats par portée racine.
association_name	chaîne	(Facultatif) Requis pour les propriétaires de portée racine. Le type d'élément à retourner. Par exemple : « H4Users »
history_action	chaîne	(Facultatif) Action de modification. Par exemple : « mettre à jour »

Nom	Type	Description
détails	chaîne	(Facultatif) Détails de l'action. Par exemple : « suppression logicielle »
before_epoch	nombre entier	(Facultatif) Incluez les résultats créés avant cet horodatage Unix.
after_epoch	nombre entier	(Facultatif) Incluez les résultats créés depuis cet horodatage Unix.
offset	nombre entier	(facultatif) Nombre de résultats à ignorer.
limit	nombre entier	(Facultatif) Nombre limité de résultats, 1 000 par défaut.

Objet de réponse : renvoie une liste des objets du journal des modifications.

Réponse

La réponse est un objet JSON contenu dans le corps, avec les propriétés suivantes.

Nom	Type	Description
total_count	nombre entier	Nombre total d'éléments correspondants avant d'appliquer le décalage ou la limite.
Éléments	tableau d'objets	Liste des résultats.

Exemple de code Python

Récupérer les 100 dernières modifications d'objets de portée dans une portée racine donnée au cours de la dernière journée.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24*60*60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

Récupérer les deuxièmes mille modifications d'objet de portée.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'offset': 1000})
```

Affinez davantage ces résultats pour afficher uniquement les créations de nouvelles portées.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24 * 60 * 60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'history_action': 'create',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

L'administrateur d'un site peut utiliser la limite et le décalage pour récupérer de manière itérative toutes les modifications dans toutes les portées.

```
resp = restclient.get('/change_logs', params={'offset': 100, 'limit': 100})
```

Points terminaux non routables

Les API suivantes sont utilisées pour gérer les points terminaux non routables, pour marquer une adresse IP ou un sous-réseau comme non routables ou obtenir une liste des points terminaux non routables marqués par un utilisateur, ou pour désélectionner une adresse IP ou un sous-réseau comme point terminal non routable. La capacité `user_data_upload` associée à la clé API est requise.

Objet de point terminal non routable

Les descriptions des attributs de l'objet terminal non routable sont les suivantes :

Attribut	Type	Description
ID	chaîne	Identifiant unique pour le point terminal non routable.
name	chaîne	Nom spécifié par l'utilisateur du point terminal non routable.
subnet	chaîne	Sous-réseau IPv4/IPv6.
vrf_id	long	ID du VRF auquel le point terminal non routable appartient.
address_type	chaîne	IPV4/IPV6 en fonction du type d'adresse de sous-réseau
host_uuid	chaîne	Identifiant unique de l'agent
description.	chaîne	Description du point terminal non routable fournie par l'utilisateur.

GET Points terminaux non routables

Ce point terminal renvoie une liste des points terminaux non routables dans le détenteur donné.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Paramètres : Aucun

Créer un point terminal non routable

Ce point terminal est utilisé pour créer un point terminal non routable.

```
POST /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Paramètres :

Attribut	Type	Description
name	chaîne	Nom spécifié par l'utilisateur du point terminal non routable.
subnet	chaîne	Le sous-réseau IPv4 ou IPv6.
address_type (facultatif)	chaîne	IPv4 ou IPv6 selon le type d'adresse de sous-réseau
host_uuid (facultatif)	chaîne	Identifiant unique de l'agent
description (facultatif)	chaîne	Description du point terminal non routable fournie par l'utilisateur.

* si les champs facultatifs ne sont pas spécifiés, des valeurs nulles sont utilisées.

Exemple de code Python

```
  req_payload = {
    "name": "nre-1",
    "subnet": "1.1.1.1/30",
    "address_type": IPV4,
    "description": "sample parameters test"
  }
  resp = restclient.post('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

Obtenir des points terminaux non routables spécifiques avec nom

Ce point terminal renvoie un point terminal non routable pour le nom spécifié.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Paramètres : Aucun

Obtenir des points terminaux spécifiques non routables avec ID

Ce point terminal renvoie un point terminal non routable pour l'ID spécifié.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Paramètres : Aucun

Mettre à jour le nom d'un point d'accès spécifique non routable

Ce point terminal est utilisé pour mettre à jour un point terminal non routable. Il utilise un ID ou un nom du point terminal non routable existant pour mettre à jour son nom.

```
PUT /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Paramètres :

Attribut	Type	Description
ID	chaîne	Identifiant unique pour le point terminal non routable.
name	chaîne	Nom spécifié par l'utilisateur du point terminal non routable.
new_name	chaîne	Nouveau nom à mettre à jour

Exemple de code Python

```
  req_payload = {
    "name": "nre-1",
    "new_name": "nre-updated",
  }
  resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))


  req_payload = {
    "id": "5f706964a5b5f16ed4b0aacb",
    "new_name": "nre-updated",
  }
  resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

Supprimer le point terminal non routable spécifique avec le nom

Ce point terminal supprime le point terminal non routable spécifique.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Supprimer un point terminal non routable spécifique avec un ID

Ce point terminal supprime le point terminal non routable spécifique.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Schémas de configuration et de commande pour les appareils et les connecteurs externes

API des groupes de configuration

L'API des groupes de configuration fournit des schémas de configuration pour les API des appareils et des connecteurs. Ces API nécessitent la capacité `sensor_management` ou `external_integration` qui est associée à la clé API.

API pour obtenir le schéma de configuration

Ce point terminal renvoie un schéma de configuration statique pour le type ou les groupes de configurations sélectionnés.

```
GET /openapi/v1/config_groups/schema/<type>
```

où <type> est le type de configuration de l'appareil.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
type	chaîne	Préciser le type de configuration parmi « VM1 » « VM3 » « NTP » « LOG » « LDAP » « NETFLOW » « IPFIX » « NETSCALER » « F5 » « AWS » « ENDPOINT » « SLACK_NOTIFIER » « GCP_CONNECTOR » « PAGERDUTY_NOTIFIER » « SYSLOG_NOTIFIER » « KINESIS_NOTIFIER » « EMAIL_NOTIFIER » « ISE » « MERAKI » « SLACK_NOTIFIER_OVERRIDE » « PAGERDUTY_NOTIFIER_OVERRIDE » « SYSLOG_NOTIFER_OVERRIDE » « KINESIS_NOTIFER_OVERRIDE » « AZURE_CONNECTOR » « EMAIL_NOTIFIER_OVERRIDE » « SYSLOG_SEVERITY_MAPPING » « SERVICENOW » « SYNC_INTERVAL » « ALERT » « VM3_ERSPAN » « AWS_CONNECTOR » « VM0 »

Objet de réponse : renvoie le schéma de configuration pour le type de configuration sélectionné.

Exemple de réponse

```
resp = restclient.get('/config_groups/schema/LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "type": "LOG",
    "name": "Log",
    "mode": "TEST",
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20 }
    },
    "fill_ins": [
      {
        "field": "log-level",
        "label": "Logging Level",
        "placeholder": "info",
        "type": "user_fill_in",
        "input_type": "dropdown",
```

```
          "possible_values": [
            "trace",
            "debug",
            "info",
            "warn",
            "error"
          ]
      },
        {
          "field": "max-log-size",
          "label": "Max Log File Size (in MB)",
          "placeholder": 10,
          "type": "user_fill_in",
          "input_type": "number",
          "min": 1,
          "max": 10240
        },
        {"field": "max-log-age",
          "label": "Log Rotation (in days)",
          "placeholder": 30,
          "type": "user_fill_in",
          "input_type": "number",
          "min": 1,
          "max": 365
        },
        {
          "field": "max-log-backups",
          "label": "Log Rotation (in instances)",
          "placeholder": 20,
          "type": "user_fill_in",
          "input_type": "number",
          "min": 1,
          "max": 100
}
      ]
    }
```


API pour obtenir le schéma des commandes de dépannage

Ce point terminal renvoie un schéma de configuration de commande de dépannage statique pour le type sélectionné de commande de dépannage.

```
GET /openapi/v1/config_groups/command_schema/<type>
```

Dans la charge utile de la demande,<type> est le type de configuration de la commande de dépannage.

Paramètres :

L'URL de la demande contient les paramètres suivants

Nom	Type	Description
type	chaîne	Spécifiez le type de commande parmi "SHOW LOG" "SHOW_SERVICE_LOG" "SHOW_RUNNING_CONF" "SHOW_SERVICE_RUNNING_CONF" "SHOW_SYS_COMMANDS" "SHOW_DOCKER_COMMANDS" "SHOW_DOCKER_INSTANCE_COMMANDS" "OPER_DOCKER_INSTANCE_COMMANDS" "SHOW_SUPERVISOR_COMMANDS" "SHOW_SUPERVISOR_SERVICE_COMMANDS" "OPER_SUPERVISOR_SERVICE_COMMANDS" "NETWORK_CONNECTIVITY_COMMANDS" "LIST_FILES" "LIST_SERVICE_FILES" "PACKET_CAPTURE " "SHOW_DATA_EXPORT_LGO" "SHOW_DATAEXPORT_RUNNING_CONF" "SHOW_DATA_EXPORT_SYS_COMMANDS" "UPDATE_LISTENING_PORT" "UPDATE_TAN_LOG_CONF" "SNAPSHOT_APPLIANCE" "SNAPSHOT_CONNECTOR" "SHOW_AWS_DOWNLOADER_LOG" "CONTROLLER_PROFILING" "SERVICE_PROFILING" "RESTART_CONNECTOR_CONTAINER" "RESTART_CONNECTOR_SERVICE" "CONNECTOR_ALERT_INTERVAL_APPLIANCE" "CONNECTOR_ALERT_INTERVAL_CONNECTOR" "EXEC_SCRIPT" "SHOW_SEGMENTATION_POLICIES"

Objet de réponse : renvoie le schéma de configuration pour le type de configuration sélectionné.

Exemple de réponse

```
resp = restclient.get('/config_groups/command_schema/SHOW_LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "name": "Show logs",
    "desc": "Show the contents of a log file",
   "long_desc": "Show the contents of a log file and optionally grep the file for a specified
 pattern. The output is tailed for the last 5000 lines.",
    "valid_appliances": [
      "TETRATION_DATA_INGEST",
      "TETRATION_EDGE",
      "TETRATION_EXPORT"
    ],
    "valid_connectors": [
      "netflow",
      "netscaler",
```

```
      "f5",
      "aws",
      "anyconnect",
      "slack",
      "kinesis",
      "syslog",
      "email",
      "pagerduty",
      "ise",
      "asa",
      "meraki",
      "servicenow",
      "wad"
    ],
    "arg_fillins": [
      {
        "field": "pattern",
        "label": "Grep Pattern",
        "input_type": "text",
        "optional": true
      }
    ],
    "output_type": "FILE",
    "output_ext": "LOG"
  }
```

Appareils externes

API des appareils externes

Les API des appareils externes sont associées à la gestion des appareils externes SecureWorkload. Cet ensemble d'API nécessite la capacité `sensor_management` ou `external_integration` associée à la clé API.

API pour obtenir la liste des appareils

Ce point terminal renvoie la liste des appareils.

```
GET /openapi/v1/ext_appliances?root_scope_id=<root_scope_id>&type=<type>
```

où <root_scope_id> est le root_scope_id qui peut être obtenu à partir de l'API Obtenir les portées, <type> est une chaîne pour décider du type d'appareil.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Préciser la portée racine
type	chaîne	Préciser le type d'appareil. la valeur peut être « TETRATION_EDGE », « TETRATION_DATA_INGEST », « TETRATION_EXPORT », « TETRATION_ERSPAN » ou « TETRATION_INTERNAL »

Objet de réponse : renvoie la liste des appareils.

Exemple de réponse

```
resp =
restclient.get('/ext_appliances?root_scope_id=63bf8d2f497d4f7287dbd335&type=TETRATION_INTERNAL')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[
    {
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "type": "tetration_internal",
      "status": {
        "state": "active",
        "controller_state": "up",
        "message": "",
        "display_state": "active"
      },
      "auto_upgrade": true,
      "created_at": 1673498141,
      "updated_at": 1673498141,
      "registered_at": 1673498141,
      "last_checkin_at": 0,
      "last_rpm_sent_at": 0,
      "upgrade_attempts": 0,
      "delete_attempts": 0,
      "last_delete_msg_sent_at": 0,
      "taas": false,
      "deleted": false,
      "deleted_at": 0,
      "connector_limit": 5000,
      "available_slots": 5000,
      "internal": true,
      "id": "63bf8e1d6419d06bef39bc85",
      "ha_peer_appliance_id": "",
      "display_type": "Tetration Internal"
    }
  ]
```


API pour créer un appareil

Ce point terminal crée un appareil.

```
POST /openapi/v1/ext_appliances
```

Dans la charge utile de la demande, pour obtenir<config>, sélectionnez l'une des réponses « valid_config » dans API pour obtenir le schéma de l'appareil, à la page 203, appliquez la commande « valid_config » à API pour obtenir le schéma de configuration, à la page 190.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Précisez le nom
root_scope_id	chaîne	Préciser la portée racine

Nom	Type	Description
type	chaîne	Préciser le type d'appareil. la valeur peut être « TETRATION_EDGE », « TETRATION_DATA_INGEST », « TETRATION_EXPORT », « TETRATION_ERSPAN » ou « TETRATION_INTERNAL »
config (configurer)	set	Fournir le schéma de configuration rempli au format JSON
taas	booléen	Indiquez si l'appareil est conçu pour l'environnement TAAS
version	chaîne	Précisez la version

Objet de réponse : renvoie l'appareil créé.

Exemple de réponse

```
req_payload = {
    "name":"Data Ingest Appliance",
    "type":"tetration_data_ingest",
    "root_scope_id":"63c41ff2497d4f5f5be73662",
    "config":{
      "VM3":{
        "secured":{},
        "unsecured":{
          "cidr":[
            "172.26.231.141/23",
            "172.26.231.142/23",
            "172.26.231.143/23"
          ],
          "gateway":[
            "172.26.231.140",
            "172.26.231.140",
            "172.26.231.140"
          ],
          "cidr_v6":[],
          "gateway_v6":[],
          "dns":[
            "testserver"
          ],
          "search_domains":[],
          "hostname":"",
          "use_proxy_for_tetration":false,
          "https_proxy":"",
          "no_proxy":[],
          "docker_subnet":""
        }
      }
    }
  }
  resp = restclient.post('/ext_appliances', json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
   "root_scope_id":"63c41ff2497d4f5f5be73662",
   "type":"tetration_data_ingest",
   "status":{
     "state":"pending_dio",
     "controller_state":"down",
     "message":"Setting up appliance",
     "display_state":"preparing"
   },
   "auto_upgrade":true,
   "created_at":1674183549,
   "updated_at":1674183549,
   "registered_at":0,
   "last_checkin_at":0,
   "last_rpm_sent_at":0,
   "upgrade_attempts":0,
   "delete_attempts":0,
   "last_delete_msg_sent_at":0,
   "name":"Data Ingest Appliance",
   "taas":false,
   "deleted":false,
   "deleted_at":0,
   "connector_limit":3,
   "available_slots":0,
   "internal":false,
   "id":"63ca037dbca44e263daeb5d0",
   "ha_peer_appliance_id":"",
   "display_type":"Tetration Data Ingest"
 }
```


API pour supprimer un appareil

Ce point terminal supprime l'appareil indiqué.

```
DELETE /openapi/v1/ext_appliances/<id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil

Objet de réponse : renvoie l'état de l'appareil supprimé.

Exemple de réponse

```
resp = restclient.delete('/ext_appliances/63be3b1ade36423c12bff6e1')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
 {
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```


API pour obtenir un appareil par ID

Ce point terminal obtient l'appareil à l'aide de l'ID donné.

```
GET /openapi/v1/ext_appliances/<id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil

Objet de réponse : renvoie l'appareil à l'aide de l'ID donné.

Exemple de réponse

```
resp = restclient.get('/ext_appliances/63bf8e1d6419d06bef39bc85')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
{
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "type": "tetration_internal",
    "status": {
      "state": "active",
      "controller_state": "up",
      "message": "",
      "display_state": "active"
    },
    "auto_upgrade": true,
    "created_at": 1673498141,
    "updated_at": 1673498141,
    "registered_at": 1673498141,
    "last_checkin_at": 0,
    "last_rpm_sent_at": 0,
    "upgrade_attempts": 0,
    "delete_attempts": 0,
    "last_delete_msg_sent_at": 0,
    "taas": false,
    "deleted": false,
    "deleted_at": 0,
    "connector_limit": 5000,
    "available_slots": 5000,
    "internal": true,
    "id": "63bf8e1d6419d06bef39bc85",
    "ha_peer_appliance_id": "",
    "display_type": "Tetration Internal"
  }
```


API pour renommer un appareil

Ce point terminal renomme l'appareil à l'aide de l'ID donné.

```
PUT /openapi/v1/ext_appliances/<id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil

Nom	Type	Description
name	chaîne	Préciser le nouveau nom de l'appareil

Objet de réponse : renvoie l'appareil avec l'ID et le nouveau nom donnés.

Exemple de réponse

```
req_payload = {
    "name": "new_name",
  }
  resp = restclient.put('/ext_appliances/63bf8e1d6419d06bef39bc85',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
{
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "type": "tetration_internal",
    "status": {
      "state": "active",
      "controller_state": "up",
      "message": "",
      "display_state": "active"
    },
    "auto_upgrade": true,
    "created_at": 1673498141,
    "updated_at": 1673498141,
    "registered_at": 1673498141,
    "last_checkin_at": 0,
    "last_rpm_sent_at": 0,
    "upgrade_attempts": 0,
     "delete_attempts": 0,
    "last_delete_msg_sent_at": 0,
    "name": "new_name",
    "taas": false,
    "deleted": false,
    "deleted_at": 0,
    "connector_limit": 5000,
    "available_slots": 5000,
    "internal": true,
    "id": "63bf8e1d6419d06bef39bc85",
    "ha_peer_appliance_id": "",
    "display_type": "Tetration Internal"
  }
```


API pour obtenir les configurations par type de configuration

Ce point terminal obtient les configurations de l'appareil avec l'ID et le type de configuration donnés.

```
GET /openapi/v1/ext_appliances/<id>/config?config_type=<config_type>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <config_type> est la « valid_config » qui peut être obtenue à partir de API pour obtenir le schéma de l'appareil, à la page 203.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
config_type	chaîne	Précisez le type de configuration. Reportez-vous à API pour obtenir le schéma de configuration, à la page 190 pour toutes les valeurs possibles répertoriées dans la description

Objet de réponse : renvoie les configurations avec l'ID d'appareil et le type de configuration donnés

Exemple de réponse

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config?config_type=VM3_ERSPAN')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[
    {
      "mode": "TEST_AND_APPLY",
      "name": "VM3_ERSPAN",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "deleted": false,
      "type": "VM3_ERSPAN",
      "state": "COMMIT",
      "attempts": 0,
      "config": {
        "secured": {},
        "unsecured": {
          "cidr": [
            "172.26.231.141/23",
            "172.26.231.142/23",
            "172.26.231.143/23"
          ],
          "gateway": [
            "172.26.231.140",
            "172.26.231.140",
            "172.26.231.140"
          ],
          "cidr_v6": [],
          "gateway_v6": [],
          "dns": [
            "test"
          ],
          "search_domains": [],
          "hostname": "hjtest",
          "https_proxy": "",
          "no_proxy": []
        }
      },
      "push_to_dio_at": 0,
      "created_at": 1673602848,
      "updated_at": 1673602848,
      "discovery_completed_at": 0,
      "committed_at": 0,
```

```
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63c1272039042a1c0ddd3e21"
  }
]
```

API pour ajouter une nouvelle configuration à un appareil externe

Ce point terminal ajoute une nouvelle configuration à l'appareil à l'aide de l'ID donné

```
POST /openapi/v1/ext_appliances/<id>/config
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193. Dans la charge utile de la demande,<type> est la « valid_config » qui peut être obtenue à partir de API pour obtenir le schéma de l'appareil, à la page 203. Pour obtenir le schéma <config>, sélectionnez une des réponses « valid_config » dans la réponse API pour obtenir le schéma de l'appareil, à la page 203, appliquez « valid_config » à API pour obtenir le schéma de configuration, à la page 190.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Préciser le nom de la configuration
type	chaîne	Précisez le type de configuration. Reportez-vous à API pour obtenir le schéma de configuration, à la page 190 pour toutes les valeurs possibles répertoriées dans la description
config (configurer)	set	Fournir le schéma de configuration rempli au format JSON

Objet de réponse : renvoie la configuration mise à jour.

Exemple de réponse

```
req_payload = {
    "name": "new_config",
    "type": "VM3_ERSPAN",
    "config": {}
  }
  resp = restclient.post('/ext_appliances/63c1272039042a1c0ddd3e20/config',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "prev_id": "63c1272039042a1c0ddd3e21",
    "mode": "TEST_AND_APPLY",
    "name": "new_config",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "appliance_id": "63c1272039042a1c0ddd3e20",
```

```
    "deleted": false,
    "type": "VM3_ERSPAN",
    "state": "COMMIT",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": null
    },
    "push_to_dio_at": 0,
    "created_at": 1673661042,
    "updated_at": 1673661042,
    "discovery_completed_at": 0,
    "committed_at": 0,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63c20a7239042a0991b871b7"
}
```


API pour supprimer une configuration

Ce point terminal supprime une configuration d'un appareil donné.

```
DELETE /openapi/v1/ext_appliances/<id>/config/<config_id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <config_id> est l'ID qui peut être obtenu à partir de API pour obtenir les configurations par type de configuration, à la page 198.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
config_id	chaîne	Préciser l'ID de configuration

Objet de réponse : renvoie l'état de la configuration supprimée pour l'appareil donné.

Exemple de réponse

```
resp =
restclient.delete('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
{
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```


API pour obtenir la configuration

Ce point terminal obtient la configuration à l'aide de l'ID donné

```
GET /openapi/v1/ext_appliances/<id>/config/<config_id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <config_id> est l'ID qui peut être obtenu à partir de API pour obtenir les configurations par type de configuration, à la page 198.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
config_id	chaîne	Préciser l'ID de configuration

Objet de réponse : renvoie la configuration à l'aide de l'ID donné.

Exemple de réponse

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
b
```

Exemple de réponse


```
{
    "mode": "TEST_AND_APPLY",
    "name": "VM3_ERSPAN",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "deleted": false,
    "type": "VM3_ERSPAN",
    "state": "COMMIT",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": {
        "cidr": [
          "172.26.231.141/23",
          "172.26.231.142/23",
          "172.26.231.143/23"
        ],
        "gateway": [
          "172.26.231.140",
          "172.26.231.140",
          "172.26.231.140"
        ],
        "cidr_v6": [],
        "gateway_v6": [],
        "dns": [
          "test"
        ],
        "search_domains": [],
        "hostname": "hjtest",
        "https_proxy": "",
        "no_proxy": []
      }
    },
    "push_to_dio_at": 0,
    "created_at": 1673602848,
    "updated_at": 1673602848,
    "discovery_completed_at": 0,
```

```
    "committed_at": 0,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63c1272039042a1c0ddd3e21"
  }
```

API pour obtenir le schéma de l'appareil

Ce point terminal obtient le schéma de l'appareil d'un type donné

```
GET /openapi/v1/ext_appliances/schema/<type>
```

où <type> est une chaîne pour décider du type d'appareil.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
type	chaîne	Préciser le type d'appareil (la valeur peut être « TETRATION_EDGE », « TETRATION_DATA_INGEST », « TETRATION_EXPORT », « TETRATION_ERSPAN » ou « TETRATION_INTERNAL »

Objet de réponse : renvoie le schéma de configuration.

Exemple de réponse

```
resp = restclient.get('/ext_appliances/schema/TETRATION_ERSPAN')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
 {
    "name": "Data Ingest for ERSPAN",
    "type": "tetration_erspan",
    "desc": "Data Ingest Appliance for ERSPAN",
    "long_desc": "Data Ingest appliance for ERSPAN is a software appliance that can export
 flow data from ERSPAN appliance.",
    "valid_config": [
      "VM3_ERSPAN"
    ],
    "deploy_config": [
      "VM3_ERSPAN"
    ],
    "connectors": [
      "ERSPAN"
    ],
    "limit_connectors_per_appliance": 0,
    "limit_per_rootscope": 8,
    "limit_per_rootscope_taas": 4,
    "limit_per_cluster": 150,
    "cco_url":
"https://software.cisco.com/download/home/286309796/type/286309874/release/3.7.1.26.devel"
  }
```

-

API pour répertorier les commandes de dépannage disponibles pour un appareil

Ce point terminal renvoie la liste des commandes de dépannage disponibles pour un appareil donné.

```
GET /openapi/v1/ext_appliances/<id>/commands/available
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil

Objet de réponse : renvoie la liste des commandes de dépannage disponibles pour un appareil donné.

Exemple de réponse

```
resp = restclient.get('/ext_appliances/63c6ef42bca44e2b5e729191/commands/available')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
[
    {
      "type": "UPDATE_LISTENING_PORT",
      "name": "Update the listening port on a connector"
    },
    {
      "type": "SNAPSHOT_APPLIANCE",
      "name": "Collect Snapshot from Appliance"
    },
    {
      "type": "LIST_FILES",
      "name": "List a directory"
    },
    {
      "type": "CONTROLLER_PROFILING",
      "name": "Collect controller profile"
    },
    {
      "type": "SHOW_LOG",
      "name": "Show logs"
    },
    {
      "type": "SHOW_SUPERVISOR_COMMANDS",
      "name": "Execute supervisorctl command"
    },
    {
      "type": "PACKET_CAPTURE",
      "name": "Packet capture"
    },
    {
      "type": "NETWORK_CONNECTIVITY_COMMANDS",
      "name": "Test network connectivity"
    },
    {
      "type": "SHOW_DOCKER_COMMANDS",
      "name": "Execute docker command"
    },
```

```
    {
      "type": "CONNECTOR_ALERT_INTERVAL_APPLIANCE",
      "name": "Override connector alert interval for Appliance"
    },
    {
      "type": "SHOW_RUNNING_CONF",
      "name": "Show running configuration"
    },
    {
      "type": "SHOW_SYS_COMMANDS",
      "name": "Execute system command"
    }
  ]
```


API pour lister toutes les commandes de dépannage

Ce point terminal renvoie la liste des commandes de dépannage activées pour l'appareil donné.

```
GET /openapi/v1/ext_appliances/<id>/commands
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil

Objet de réponse : renvoie la liste des commandes de dépannage activées pour l'appareil donné.

Exemple de réponse

```
resp = restclient.get('/ext_appliances/63be3b1ade36423c12bff6e1/commands')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[
    {
      "appliance_id": "63be3b1ade36423c12bff6e1",
      "state": "pending",
      "level": "APPLIANCE",
      "command": "SHOW_LOG",
      "arg_string": "",
      "args": {},
      "tailed": false,
      "rc": 0,
      "push_to_dio_at": 0,
      "attempts": 0,
      "deleted": false,
      "deleted_at": 0,
      "created_at": 1673595392,
      "total_chunk": 0,
      "id": "63c10a0039042a6aee1b008c"
    }
  ]
```


API pour créer une commande de dépannage

Ce point terminal crée une commande de dépannage disponible pour un appareil donné.

```
POST /openapi/v1/ext_appliances/<id>/commands
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193. Dans la charge utile de la demande,<command> est un type de commande de dépannage qui peut être obtenu à partir du champ « valid_appliances » de la réponse API pour obtenir le schéma des commandes de dépannage, à la page 191. <arguments>est un objet JSON contenant le schéma de commande, qui peut être obtenu à partir de API pour obtenir le schéma des commandes de dépannage, à la page 191.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
commande	chaîne	Préciser le type de commande
arguments	set	Fournir le schéma de commande rempli au format JSON

Objet de réponse : renvoie la commande de dépannage créée pour l'appareil donné.

Exemple de réponse

```
req_payload = {
    "command": "SHOW_LOG",
    "arguments": {}
  }
  resp = restclient.post('/ext_appliances/63be3b1ade36423c12bff6e1/commands',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "appliance_id": "63be3b1ade36423c12bff6e1",
    "state": "pending",
    "level": "APPLIANCE",
    "command": "SHOW_LOG",
    "args": {},
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 0,
    "attempts": 0,
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1673595392,
    "total_chunk": 0,
    "id": "63c10a0039042a6aee1b008c"
  }
```

-

API pour supprimer une commande de dépannage

Ce point terminal supprime une commande de dépannage activée pour l'appareil donné.

```
DELETE /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <command_id> est l'ID qui peut être obtenu à partir de API pour lister toutes les commandes de dépannage, à la page 205.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : renvoie l'état de la commande de dépannage supprimée pour l'appareil donné.

Exemple de réponse

```
resp =
restclient.delete('/ext_appliances/63be3b1ade36423c12bff6e1/commands/63c10a0039042a6aee1b008c')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
 {
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API pour renvoyer une commande de dépannage

Ce point terminal renvoie la commande de dépannage sélectionnée pour un appareil donné.

```
GET /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

où <id> est l'appareil_id obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <command_id> est l'ID obtenu de API pour lister toutes les commandes de dépannage, à la page 205.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : renvoie la commande de dépannage sélectionnée pour un appareil donné.

Exemple de réponse

```
resp =
restclient.get('/ext_appliances/63be3b1ade36423c12bff6e1/commands/63c10fd139042a1c0ddd3e1f')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "appliance_id": "63be3b1ade36423c12bff6e1",
    "state": "pending",
    "level": "APPLIANCE",
    "command": "SHOW_LOG",
    "arg_string": "",
```

```
    "args": {},
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 0,
    "attempts": 0,
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1673596881,
    "total_chunk": 0,
    "id": "63c10fd139042a1c0ddd3e1f"
}
```


API pour télécharger la sortie de la commande de l'appareil sous forme de fichier

Ce point terminal télécharge la sortie de la commande en tant que fichier.

```
GET /openapi/v1/appliances/<id>/commands/{command_id}/download
```

où <id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <command_id> est l'ID qui peut être obtenu à partir de API pour lister toutes les commandes de dépannage, à la page 205. Toutes les commandes n'ont pas une sortie téléchargeable, vérifiez le schéma de commande fourni par API pour obtenir le schéma des commandes de dépannage, à la page 191, où « output_type » : « FILE » indique que du contenu est téléchargeable et « output_ext » indique le type de fichier.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : téléchargez le résultat de la commande sous forme de fichier.

Exemple de réponse

```
resp = restclient.download('downloadFile',
'/appliances/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')
```

Connecteurs

API de connecteurs

Les API des connecteurs sont associées à la gestion des connecteurs de Cisco Secure Workload. Cet ensemble d'API nécessite la capacité `sensor_management` ou `external_integration` associée à la clé API.

API pour obtenir tous les types de connecteurs

Ce point terminal obtient tous les types de connecteurs d'une portée racine donnée.

```
GET /openapi/v1/connectors/cards?root_scope_id=<root_scope_id>
```

où <root_scope_id> est le root_scope_id qui peut être obtenu à partir de l'API Obtenir les portées, à la page 63.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Préciser la portée racine

Objet de réponse : renvoie tous les types de connecteurs avec un ID de portée racine donné.

Exemple de réponse

```
resp = restclient.get('/connectors/cards?root_scope_id=63bf8d2f497d4f7287dbd335')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[{
    "type": "NETFLOW",
    "name": "NetFlow",
    "desc": "Collect telemetry from network devices",
    "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as
routers and switches.",
    "group": "Flow Ingest",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "NETSCALER",
    "name": "NetScaler",
    "desc": "Collect telemetry from Citrix ADC",
   "long_desc": "Collect telemetry from Citrix ADC, stitch client and server side flows.",

    "group": "Flow Ingest",
    "index": 2,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
```

```
  {
    "type": "F5",
    "name": "F5",
    "desc": "Collect telemetry from F5 BIG-IP",
    "long_desc": "Collect telemetry from F5 BIG-IP, stitch client and server side flows,
enrich client inventory with user attributes.",
    "group": "Flow Ingest",
    "index": 1,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "User Insights",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "ANYCONNECT",
    "name": "AnyConnect",
    "desc": "Collect telemetry from AnyConnect NVM",
    "long_desc": "Collect telemetry data from Cisco AnyConnect Network Visibility Module
(NVM) and enrich endpoint inventories with user attributes",
    "group": "Endpoints",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 50,
    "limit_per_cluster": 500,
    "config": [
      "ENDPOINT",
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Process Annotation",
      "User Insights",
      "Endpoint Insights",
      "Inventory Enrichment"
    ],
    "connector_count": 0,
    "group_order": 1
  },
  {
    "type": "ASA",
```

```
    "name": "Cisco Secure Firewall",
    "desc": "Collect telemetry from Cisco Secure Firewall",
    "long_desc": "Collect telemetry from Cisco Secure Firewall, stitch client and server
side flows. Recommended usage with ISE connector for flow visibility.",
    "group": "Flow Ingest",
    "index": 3,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility",
      "Flow Stitching",
      "ADM"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "SLACK",
    "name": "Slack",
    "desc": "Send alerts to Slack",
    "long_desc": "Send Tetration Alerts to Slack.",
    "group": "Alerts",
    "index": 2,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SLACK_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "KINESIS",
    "name": "Kinesis",
    "desc": "Send alerts to Kinesis",
    "long_desc": "Send Tetration Alerts to Kinesis.",
    "group": "Alerts",
    "index": 4,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
```

```
      "KINESIS_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "SYSLOG",
    "name": "Syslog",
    "desc": "Send alerts to Syslog server",
    "long_desc": "Send Tetration Alerts to Syslog server.",
    "group": "Alerts",
    "index": 0,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SYSLOG_NOTIFIER",
      "SYSLOG_SEVERITY_MAPPING",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "EMAIL",
    "name": "Email",
    "desc": "Send alerts to Email",
    "long_desc": "Send Tetration Alerts to Email.",
    "group": "Alerts",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "EMAIL_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
```

```
    "type": "PAGERDUTY",
    "name": "Pager Duty",
    "desc": "Send alerts to Pager Duty",
    "long_desc": "Send Tetration Alerts to Pager Duty",
    "group": "Alerts",
    "index": 3,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "PAGERDUTY_NOTIFIER",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Alert Destination"
    ],
    "connector_count": 0,
    "group_order": 3
  },
  {
    "type": "ISE",
    "name": "ISE",
    "desc": "Collect endpoints and inventories from Cisco ISE",
    "long_desc": "Collect information about endpoints and inventories managed by Cisco ISE
 appliances and enrich endpoint inventories with user attributes and secure group tags
(SGT). Recommended usage with Cisco Secure Firewall connector for flow visibility.",
    "group": "Endpoints",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "LDAP",
      "LOG",
      "ALERT"
    ],
    "instance_spec": "ISE",
    "max_instances": 20,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "User Insights",
      "Inventory Enrichment",
      "Endpoint Insights",
      "Software Compliance Posture",
      "MDM Insights"
    ],
    "connector_count": 0,
    "group_order": 1
  },
  {
    "type": "MERAKI",
    "name": "Meraki",
    "desc": "Collect telemetry from Meraki firewalls",
    "long_desc": "Collect telemetry data from Meraki firewalls.",
    "group": "Flow Ingest",
    "index": 5,
```

```
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 10,
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "group_order": 0
  },
  {
    "type": "SERVICENOW",
    "name": "ServiceNow",
    "desc": "Collect ServiceNow CMDB records for inventories",
   "long_desc": "Collect CMDB information and service records from ServiceNow instance and
enriches endpoint inventories with cmdb attributes.",
    "group": "Inventory Enrichment",
    "index": 1,
    "appliance_type": "tetration_edge",
    "state": "disabled",
    "limit_per_appliance": 1,
    "limit_per_rootscope": 1,
    "limit_per_cluster": 150,
    "config": [
      "SERVICENOW",
      "SYNC_INTERVAL",
      "LOG",
      "ALERT"
    ],
    "instance_spec": "SERVICENOW",
    "max_instances": 10,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "User Insights",
      "Inventory Enrichment",
      "Endpoint Insights",
      "Software Compliance Posture"
    ],
    "connector_count": 0,
    "group_order": 2
  },
  {
    "type": "ERSPAN",
    "name": "ERSPAN",
    "desc": "Collect ERSPAN traffic",
    "long_desc": "",
    "group": "Flow Ingest",
    "index": 7,
    "appliance_type": "tetration_erspan",
    "state": "enabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 24,
    "limit_per_cluster": 450,
    "config": [],
    "max_instances": 0,
```

```
  "noteworthy": false,
  "hidden": false,
  "capabilities": [],
  "connector_count": 3,
  "group_order": 0
},
{
  "type": "AWS_CONNECTOR",
  "name": "AWS",
  "desc": "AWS Connector",
  "long_desc": "",
  "group": "Cloud",
  "index": 0,
  "appliance_type": "tetration_internal",
  "state": "disabled",
  "limit_per_appliance": 5000,
  "limit_per_rootscope": 5000,
  "limit_per_cluster": 100000,
  "config": [
    "AWS_CONNECTOR"
  ],
  "max_instances": 0,
  "noteworthy": true,
  "pre_release_tag": "BETA",
  "hidden": false,
  "capabilities": [
    "Flow Visibility",
    "Segmentation",
    "Managed K8s",
    "Inventory Enrichment"
  ],
  "connector_count": 0,
  "group_order": 5
},
{
  "type": "AZURE_CONNECTOR",
  "name": "Azure",
  "desc": "Azure Connector",
  "long_desc": "",
  "group": "Cloud",
  "index": 1,
  "appliance_type": "tetration_internal",
  "state": "disabled",
  "limit_per_appliance": 5000,
  "limit_per_rootscope": 5000,
  "limit_per_cluster": 100000,
  "config": [
    "AZURE_CONNECTOR"
  ],
  "max_instances": 0,
  "noteworthy": true,
  "pre_release_tag": "BETA",
  "hidden": false,
  "capabilities": [
    "Flow Visibility",
    "Segmentation",
    "Managed K8s",
    "Inventory Enrichment"
  ],
  "connector_count": 0,
  "group_order": 5
},
{
  "type": "GCP_CONNECTOR",
```

```
    "name": "GCP",
    "desc": "GCP Connector",
    "long_desc": "",
    "group": "Cloud",
    "index": 2,
    "appliance_type": "tetration_internal",
    "state": "disabled",
    "limit_per_appliance": 5000,
    "limit_per_rootscope": 5000,
    "limit_per_cluster": 100000,
    "config": [
      "GCP_CONNECTOR"
    ],
    "max_instances": 0,
    "noteworthy": true,
    "pre_release_tag": "BETA",
    "hidden": false,
    "capabilities": [
      "Managed K8s"
    ],
    "connector_count": 0,
    "group_order": 5
  }]
```

API pour supprimer un connecteur

Ce point terminal supprime le connecteur donné.

```
DELETE /openapi/v1/connectors/<id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur

Objet de réponse : renvoie l'état du connecteur supprimé.

Exemple de réponse

```
resp = restclient.delete('/connectors/63c12e316419d0131767e21c')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
 {
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API pour obtenir un connecteur par ID

Ce point terminal obtient le connecteur à l'aide de l'ID donné.

```
GET /openapi/v1/connectors/<id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur

Objet de réponse : renvoie le connecteur à l'aide de l'ID donné.

Exemple de réponse

```
resp = restclient.get('/connectors/63c12e316419d0131767e21b')
 if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
   "name": "ERSPAN",
   "updated_at": 0,
   "created_at": 1673604657,
   "appliance_id": "63c1272039042a1c0ddd3e20",
   "root_scope_id": "63bf8d2f497d4f7287dbd335",
   "vrf_id": 1,
   "type": "ERSPAN",
   "ip_bindings": [],
   "internal": false,
   "id": "63c12e316419d0131767e21b"
 }
```

API pour renommer un connecteur

Ce point terminal renomme le connecteur à l'aide de l'ID donné.

```
PUT /openapi/v1/connectors/<id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
name	chaîne	Précisez le nouveau nom du connecteur

Objet de réponse : renvoie le connecteur avec l'ID et le nouveau nom donnés.

Exemple de réponse

```
req_payload = {
    "name": "ERSPAN2",
  }
  resp = restclient.put('/ext_appliances/63c12e316419d0131767e21b',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "name": "ERSPAN2",
    "updated_at": 0,
```

```
    "created_at": 1673604657,
    "appliance_id": "63c1272039042a1c0ddd3e20",
    "root_scope_id": "63bf8d2f497d4f7287dbd335",
    "vrf_id": 1,
    "type": "ERSPAN",
    "ip_bindings": [],
    "internal": false,
    "id": "63c12e316419d0131767e21b"
  }
```

API pour obtenir des renseignements sur le connecteur avec plus de détails

Ce point terminal obtient les informations sur le connecteur avec les détails.

```
GET /openapi/v1/connectors/cards/<type>?root_scope_id=<root_scope_id>
```

où <root_scope_id> est le root_scope_id qui peut être obtenu à partir de l'API Obtenir les portées, à la page 63. Dans la charge utile de la demande,<type> est une chaîne pour décider du type de connecteur.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Préciser la portée racine
type	chaîne	Précisez le type de connecteur. La valeur peut être « NETFLOW », « NETSCALER », « F5 » « AWS » « ANYCONNECT » « ASA » « SLACK » « KINESIS » « SYSLOG » « EMAIL » « MERAKI » « PAGERDUTY » « ISE » « SERVICENOW » « ERSPAN » « AWS_CONNECTOR » « AZURE_CONNECTOR » « GCP_CONNECTOR »

Objet de réponse : renvoie les connecteurs d'une portée donnée.

Exemple de réponse

```
resp = restclient.get('/connectors/cards/NETFLOW?root_scope_id=63bf8d2f497d4f7287dbd335')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "type": "NETFLOW",
    "name": "NetFlow",
    "desc": "Collect telemetry from network devices",
    "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as
routers and switches.",
    "group": "Flow Ingest",
    "index": 0,
    "appliance_type": "tetration_data_ingest",
    "state": "disabled",
    "limit_per_appliance": 3,
    "limit_per_rootscope": 10,
```

```
    "limit_per_cluster": 100,
    "config": [
      "LOG",
      "ALERT"
    ],
    "max_instances": 0,
    "noteworthy": false,
    "hidden": false,
    "capabilities": [
      "Flow Visibility"
    ],
    "connector_count": 0,
    "info": {
      "help": "NetFlow connector collects telemetry data from various network devices (such
 as routers, switches).<br> It supports ingest of telemetry data in IPFIX and NetFlow V9
protocols. This connector can be used to discover inventory as it provides a network context.
 The connector helps convert data from flow exports and send them securely as Tetration
Flow records into an instance of Tetration. <br><br><b> Connector Alerts: </b><br> When
Connector Alerts are enabled, you may receive the following alerts: <br>   1. NetFlow
Connector is down (due to missing heartbeats). <br>   2. Informational alert on high
CPU/Memory usage."
    },
    "group_order": 0
  }
```


API pour obtenir des connecteurs

Ce point terminal renvoie tous les connecteurs pour un appareil donné.

```
GET
/openapi/v1/connectors?root_scope_id=<root_scope_id>&appliance_id=<appliance_id>&type=<type>&state=<state>
```

où <root_scope_id> est le root_scope_id qui peut être obtenu à partir de l'API Obtenir les portées, à la page 63, <appliance_id> est l'appareil_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193, <type> est une chaîne pour décider du type de connecteur qui peut être obtenue à partir du champ API pour obtenir le schéma de l'appareil, à la page 203 « connecteurs », <state> est un filtre pour l'état des connecteurs.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
root_scope_id	chaîne	Préciser la portée racine
appliance_id	chaîne	Préciser l'ID de l'appareil
type	chaîne	Précisez le type de connecteur. La valeur peut être « NETFLOW », « NETSCALER » « F5 » « AWS » « ANYCONNECT » « ASA » « SLACK » « KINESIS » « SYSLOG » « EMAIL » « MERAKI » « PAGERDUTY » « ISE » « SERVICENOW » « ERSPAN » « AWS_CONNECTOR » « AZURE_CONNECTOR » « GCP_CONNECTOR »

Nom	Type	Description
state	chaîne	Filtrez l'état des connecteurs (la valeur peut être « ENABLED », « DISABLED » ou « UNAVAILABLE »)

Objet de réponse : renvoie tous les connecteurs pour un appareil donné.

Exemple de réponse

```
resp =
restclient.get('root_scope_id=63bf8d2f497d4f7287dbd335&appliance_id=63c1272039042a1c0ddd3e20&type=ERSPAN&state=ENABLED')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21b"
    },
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21c"
    },
    {
      "name": "ERSPAN",
      "updated_at": 0,
      "created_at": 1673604657,
      "appliance_id": "63c1272039042a1c0ddd3e20",
      "root_scope_id": "63bf8d2f497d4f7287dbd335",
      "vrf_id": 1,
      "type": "ERSPAN",
      "ip_bindings": [],
      "state": "enabled",
      "internal": false,
      "id": "63c12e316419d0131767e21d"
    }
  ]
```

API pour créer un connecteur

Ce point terminal crée un connecteur pour un appareil donné.

```
POST /openapi/v1/connectors
```

Dans la charge utile de la demande,<root_scope_id> est le root_scope_id qui peut être obtenu à partir de l'API Obtenir les portées, à la page 63,<appliance_id> est l'appliance_id qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193,<type> est une chaîne pour décider du type de connecteur qui peut être obtenue à partir du champ API pour obtenir le schéma de l'appareil, à la page 203 « connecteurs ».

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Précisez le nom
root_scope_id	chaîne	Préciser la portée racine
appliance_id	chaîne	Préciser l'ID de l'appareil
type	chaîne	Précisez le type de connecteur. La valeur peut être « NETFLOW », « NETSCALER » « F5 » « AWS » « ANYCONNECT » « ASA » « SLACK » « KINESIS » « SYSLOG » « EMAIL » « MERAKI » « PAGERDUTY » « ISE » « SERVICENOW » « ERSPAN » « AWS_CONNECTOR » « AZURE_CONNECTOR » « GCP_CONNECTOR »
version	chaîne	Précisez la version

Objet de réponse : renvoie le connecteur créé.

Exemple de réponse

```
req_payload = {
    "root_scope_id": "63c41ff2497d4f5f5be73662",
    "appliance_id": "63c6ef42bca44e2b5e729191",
    "type": "NETFLOW",
    "name": "netflowtest",
    "version": "1.1.1"
  }
  resp = restclient.post('/connectors', json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "name": "netflowtest",
    "updated_at": 0,
    "created_at": 1674187875,
    "appliance_id": "63c6ef42bca44e2b5e729191",
    "root_scope_id": "63c41ff2497d4f5f5be73662",
```

```
    "vrf_id": 1,
    "type": "NETFLOW",
    "ip_bindings": [],
    "sources": [],
    "internal": false,
    "id": "63ca14631a49bd4c0e0cefa2"
  }
```


API pour obtenir les configurations sur le type de configuration du connecteur

Ce point terminal obtient les configurations du connecteur à l'aide de l'ID donné.

```
GET /openapi/v1/connectors/<id>/config?config_type=<config_type>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir la liste des appareils, à la page 193. <config_type>est la « valid_config » qui peut être obtenue à partir de API pour obtenir le schéma de l'appareil, à la page 203.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
config_type	chaîne	Précisez le type de configuration. Reportez-vous à API pour obtenir le schéma de configuration, à la page 190 pour toutes les valeurs possibles répertoriées dans la description

Objet de réponse : renvoie les configurations avec l'ID de connecteur et le type de configuration donnés.

Exemple de réponse

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config?config_type=LOG')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
 [{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 22:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "deleted": false,
    "type": "LOG",
    "state": "COMMIT",
    "error_code": "NO_ERROR",
    "error_text": "",
    "attempts": 1,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
```

```
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 1675319360,
    "created_at": 1675319360,
    "updated_at": 1675319364,
    "discovery_completed_at": 0,
    "committed_at": 1675319364,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db5840f029813659f9fcf5"
  }]
```

API pour ajouter une nouvelle configuration au connecteur

Ce point terminal ajoute une nouvelle configuration au connecteur à l'aide de l'ID donné

```
POST /openapi/v1/connectors/<id>/config
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219. <config_type>est la « valid_config » qui peut être obtenue à partir de API pour obtenir le schéma de l'appareil, à la page 203. Pour obtenir<config> schéma, sélectionnez une des « config » de la réponse API pour obtenir tous les types de connecteurs, à la page 208, appliquez la « config » à API pour obtenir le schéma de configuration, à la page 190.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
name	chaîne	Préciser le nom de la configuration
type	chaîne	Précisez le type de configuration. Reportez-vous à API pour obtenir le schéma de configuration, à la page 190 pour toutes les valeurs possibles répertoriées dans la description
config (configurer)	set	Fournir le schéma de configuration rempli au format JSON

Objet de réponse : renvoie la configuration mise à jour.

Exemple de réponse

```
req_payload = {
    "name": "Log instance 2/1/23 22:29",
    "type": "LOG",
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    }
  }
```

```
  resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/config',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 11:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "deleted": false,
    "type": "LOG",
    "state": "PENDING",
    "attempts": 0,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 0,
    "created_at": 1675322272,
    "updated_at": 1675322272,
    "discovery_completed_at": 0,
    "committed_at": 0,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db63a0f029813659f9fcf7"
  }
```


API pour supprimer une configuration

Ce point terminal supprime une configuration d'un connecteur donné.

```
DELETE /openapi/v1/connectors/<id>/config/<config_id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219, <config_id> est l'ID qui peut être obtenu à partir de API pour obtenir les configurations sur le type de configuration du connecteur, à la page 222.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
config_id	chaîne	Préciser l'ID de configuration

Objet de réponse : renvoie l'état de la configuration supprimée pour un connecteur donné.

Exemple de réponse

```
resp =
restclient.delete('/connectors/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

-

API pour obtenir la configuration

Ce point terminal obtient la configuration à l'aide de l'ID donné

```
GET /openapi/v1/connectors/<id>/config/<config_id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219, <config_id> est l'ID qui peut être obtenu à partir de API pour obtenir les configurations sur le type de configuration du connecteur, à la page 222.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID de l'appareil
config_id	chaîne	Préciser l'ID de configuration

Objet de réponse : renvoie la configuration à l'aide de l'ID donné.

Exemple de réponse

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config/63db5840f029813659f9fcf5')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "mode": "TEST_AND_APPLY",
    "name": "Log instance 2/1/23 22:29",
    "root_scope_id": "63d98f45497d4f53005b24fa",
    "vrf_id": 1,
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "deleted": false,
    "type": "LOG",
    "state": "COMMIT",
    "error_code": "NO_ERROR",
    "error_text": "",
    "attempts": 1,
    "config": {
      "secured": {},
      "unsecured": {
        "log-level": "info",
```

```
        "max-log-size": 10,
        "max-log-age": 30,
        "max-log-backups": 20
      }
    },
    "push_to_dio_at": 1675319360,
    "created_at": 1675319360,
    "updated_at": 1675319364,
    "discovery_completed_at": 0,
    "committed_at": 1675319364,
    "delete_at": 0,
    "error_at": 0,
    "hidden": false,
    "discovery_phase": 0,
    "internal": false,
    "id": "63db5840f029813659f9fcf5"
  }
```


API pour répertorier les commandes de dépannage disponibles pour le connecteur

Ce point terminal renvoie la liste des commandes de dépannage disponibles pour un connecteur donné.

```
GET /openapi/v1/connectors/<id>/commands/available
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur

Objet de réponse : renvoie la liste des commandes de dépannage disponibles pour un connecteur donné.

Exemple de réponse

```
resp = restclient.get('/connectors/63c6f2701a49bd2bb0696cab/commands/available')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse


```
[
    {
      "type": "LIST_SERVICE_FILES",
      "name": "List a directory in a service"
    },
    {
      "type": "CONTROLLER_PROFILING",
      "name": "Collect controller profile"
    },
    {
      "type": "SHOW_LOG",
      "name": "Show logs"
    },
    {
      "type": "SHOW_SERVICE_LOG",
      "name": "Show service logs"
    },
    {
      "type": "RESTART_CONNECTOR_CONTAINER",
      "name": "Restart the connector docker container"
    },
```

```
    {
      "type": "SHOW_SUPERVISOR_COMMANDS",
      "name": "Execute supervisorctl command"
    },
    {
      "type": "CONNECTOR_ALERT_INTERVAL_CONNECTOR",
      "name": "Override connector alert interval for Connector"
    },
    {
      "type": "SERVICE_PROFILING",
      "name": "Collect connector profile"
    },
    {
      "type": "SNAPSHOT_CONNECTOR",
      "name": "Collect Snapshot from a connector"
    },
    {
      "type": "PACKET_CAPTURE",
      "name": "Packet capture"
    },
    {
      "type": "NETWORK_CONNECTIVITY_COMMANDS",
      "name": "Test network connectivity"
    },
    {
      "type": "SHOW_SERVICE_RUNNING_CONF",
      "name": "Show running configuration of a service"
    },
    {
      "type": "RESTART_CONNECTOR_SERVICE",
      "name": "Restart the connector service"
    },
    {
      "type": "SHOW_SYS_COMMANDS",
      "name": "Execute system command"
    }
  ]
```


API pour lister toutes les commandes de dépannage

Ce point terminal renvoie la liste des commandes de dépannage activées pour un connecteur donné.

```
GET /openapi/v1/connectors/<id>/commands
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur

Objet de réponse : renvoie la liste des commandes de dépannage activées pour un connecteur donné.

Exemple de réponse

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
[
    {
      "appliance_id": "63dad690e6ee1131f255e985",
      "connector_id": "63db5418e6ee1167a4c0986c",
      "service_id": "63db5418e6ee1167a4c0986d",
      "state": "success",
      "level": "SERVICE",
      "command": "SHOW_LOG",
      "arg_string": "",
      "args": {
        "pattern": "info"
      },
      "tailed": false,
      "rc": 0,
      "push_to_dio_at": 1675319615,
      "attempts": 1,
      "error_code": "NO_ERROR",
      "error_text": "",
      "deleted": false,
      "deleted_at": 0,
      "created_at": 1675319613,
      "total_chunk": 0,
      "id": "63db593df029813659f9fcf6"
    }
  ]
```


API pour créer une commande de dépannage

Ce point terminal crée une commande de dépannage disponible pour un connecteur donné.

```
POST /openapi/v1/connectors/<id>/commands
```

ici<id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219. Dans la charge utile de la demande,<command> est un type de commande de dépannage qui peut être obtenu à partir de API pour répertorier les commandes de dépannage disponibles pour le connecteur, à la page 226. <arguments>est un objet JSON contenant le schéma de commande, qui peut être obtenu à partir de API pour obtenir le schéma des commandes de dépannage, à la page 191.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
commande	chaîne	Préciser le type de commande
arguments	set	Fournir le schéma de commande rempli au format JSON

Objet de réponse : renvoie la commande de dépannage créée pour l'appareil donné.

Exemple de réponse

```
req_payload = {
    "command": "SHOW_LOG",
    "arguments": {
      "pattern": "info"
    }
  }
  resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/commands',
json_body=json.dumps(req_payload))
  if resp.status_code == 200:
```

```
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "state": "pending",
    "level": "SERVICE",
    "command": "SHOW_LOG",
    "args": {
      "pattern": "info"
    },
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 0,
    "attempts": 0,
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1675319613,
    "total_chunk": 0,
    "id": "63db593df029813659f9fcf6"
  }
```

API pour supprimer une commande de dépannage

Ce point terminal supprime une commande de dépannage disponible pour un connecteur donné.

```
DELETE /openapi/v1/connectors/<id>/commands/<command_id>
```

où <id> est un ID qui peut être obtenu à partir de la commande API pour obtenir des connecteurs, à la page 219, <command_id> est l'ID qui peut être obtenu à partir de API pour lister toutes les commandes de dépannage, à la page 205.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : renvoie l'état de la commande de dépannage supprimée pour le connecteur donné.

Exemple de réponse

```
resp =
restclient.delete('/connectors/63c12e316419d0131767e21c/commands/63c10a0039042a6aee1b008c')

  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "status": 200,
    "code": 1000,
    "message": "deleted"
  }
```

API pour renvoyer une commande de dépannage

Ce point terminal renvoie la commande de dépannage sélectionnée pour un connecteur donné.

```
GET /openapi/v1/connectors/<id>/commands/<command_id>
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219,<command_id> est l'ID qui peut être obtenu à partir de API pour lister toutes les commandes de dépannage, à la page 205.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : renvoie la commande de dépannage sélectionnée pour un connecteur donné.

Exemple de réponse

```
resp =
restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands/63db593df029813659f9fcf6')
  if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp)
```

Exemple de réponse

```
{
    "appliance_id": "63dad690e6ee1131f255e985",
    "connector_id": "63db5418e6ee1167a4c0986c",
    "service_id": "63db5418e6ee1167a4c0986d",
    "state": "success",
    "level": "SERVICE",
    "command": "SHOW_LOG",
    "arg_string": "",
    "args": {
      "pattern": "info"
    },
    "tailed": false,
    "rc": 0,
    "push_to_dio_at": 1675319615,
    "attempts": 1,
    "error_code": "NO_ERROR",
    "error_text": "",
    "deleted": false,
    "deleted_at": 0,
    "created_at": 1675319613,
    "total_chunk": 0,
    "id": "63db593df029813659f9fcf6"
  }
```

API pour télécharger la sortie de la commande du connecteur sous forme de fichier

Ce point terminal télécharge la sortie de la commande en tant que fichier.

```
GET /openapi/v1/connectors/<id>/commands/{command_id}/download
```

où <id> est l'ID qui peut être obtenu à partir de API pour obtenir des connecteurs, à la page 219,<command_id> est l'ID qui peut être obtenu à partir de API pour lister toutes les commandes de dépannage, à la page 205. Toutes les commandes n'ont pas une sortie téléchargeable, vérifiez le schéma de commande fourni par API

pour obtenir le schéma des commandes de dépannage, à la page 191, où « output_type » : « FILE » indique qu'il a du contenu téléchargeable et « output_ext » indique le type de fichier.

Paramètres : L'URL de la demande contient les paramètres suivants :

Nom	Type	Description
ID	chaîne	Préciser l'ID du connecteur
command_id	chaîne	Préciser l'ID de commande

Objet de réponse : renvoie la commande de dépannage sélectionnée pour un connecteur donné.

Exemple de réponse

```
resp = restclient.download('downloadFile',
'/connectors/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')
```

À propos de la traduction

Cisco peut fournir des traductions du présent contenu dans la langue locale pour certains endroits. Veuillez noter que des traductions sont fournies à titre informatif seulement et, en cas d’incohérence, la version anglaise du présent contenu prévaudra.