# Table of Contents

**About this Guide** .............................................................................................................. 5  
About Qualys ......................................................................................................................... 5  
Qualys Support ....................................................................................................................... 5  
About Container Security Documentation ............................................................................... 5  

**Get Started** ......................................................................................................................... 6  
Accessing the APIs .................................................................................................................. 6  
Permissions required to use APIs ............................................................................................ 6  
Qualys API URLs ..................................................................................................................... 6  
Authentication for gateway URLs ............................................................................................ 7  
Where’s the Swagger UI? ......................................................................................................... 7  
Do I need to Authenticate to use Swagger? ........................................................................... 8  
API Rate Limiting .................................................................................................................... 8  
API output shows dates in Unix time (Epoch time) ............................................................... 10  
Kubernetes cluster attributes in API output .......................................................................... 10  

**Containers** ......................................................................................................................... 11  
Fetch a list of containers in your account ................................................................................ 12  
Fetch container details ............................................................................................................ 14  
Fetch a detailed containers list ............................................................................................... 16  
Fetch a list of software installed on a container ..................................................................... 21  
Fetch vulnerability details for a container ............................................................................. 22  
Fetch vulnerability count for a container ............................................................................... 25  
Delete containers in your account ......................................................................................... 26  

**Images** ................................................................................................................................. 29  
Fetch a list of images in your account .................................................................................... 30  
Fetch image details ................................................................................................................ 32  
Fetch a detailed images list .................................................................................................... 34  
Fetch associations for an image .............................................................................................. 39  
Fetch a list of repositories that contain this image ............................................................... 40  
Fetch a list of software installed on an image ........................................................................ 41  
Fetch vulnerability details for an image ................................................................................ 42  
Fetch vulnerability count for an image .................................................................................. 44  
Delete images in your account ............................................................................................. 45  

**Registries** ............................................................................................................................. 48  
Fetch a list of registries in your account .................................................................................. 50  
Fetch registry details .............................................................................................................. 51  
Fetch a list of ACR connectors in your account ..................................................................... 52
Fetch ACR connector details ................................................................. 53
Create a new ACR connector ................................................................. 53
Fetch AWS account ID and external ID ............................................... 54
Fetch AWS US GovCloud account ID and external ID ................. 54
Fetch a list of AWS connectors in your account ........................................ 55
Fetch a list of AWS connectors for a certain account ID ............ 55
Create a new AWS connector ................................................................. 56
Fetch a list of GCR connectors in your account ................................. 56
Fetch GCR connector details ................................................................. 57
Create a new GCR connector ................................................................. 58
Validate registry parameters ................................................................. 58
Create registry ....................................................................................... 60
Update registry ....................................................................................... 62
Fetch a list of repositories in a registry ............................................. 64
Fetch a list of schedules created for a registry ..................................... 65
Create registry schedule ................................................................. 67
Update registry schedule ................................................................. 68
Cancel registry schedule ................................................................. 69
Delete registries in your account ....................................................... 69
Delete a registry schedule ................................................................. 71
Delete multiple registry schedules (bulk delete) ................................. 71

Sensors ............................................................................................... 73
Fetch a list of sensors in your account ................................................ 73
Fetch sensor details ............................................................................. 76
Delete sensors in your account ......................................................... 78

Activation Key .................................................................................. 80
Get activation key ................................................................................ 80

Vulnerability Reports ....................................................................... 81
Create a report request ....................................................................... 81
Fetch a list of reports ........................................................................... 83
Download a report ............................................................................. 84
Delete reports ....................................................................................... 85

Compliance ...................................................................................... 86
Fetch compliance posture for an image ............................................. 86
Fetch compliance posture for a container ......................................... 87
Fetch control details ........................................................................... 88
Fetch a list of controls ......................................................................... 90
About this Guide

This user guide is intended for application developers who will use the Qualys Container Security API.

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of cloud-based security and compliance solutions. The Qualys Cloud Platform and its integrated apps help businesses simplify security operations and lower the cost of compliance by delivering critical security intelligence on demand and automating the full spectrum of auditing, compliance and protection for IT systems and web applications.

Founded in 1999, Qualys has established strategic partnerships with leading managed service providers and consulting organizations including Accenture, BT, Cognizant Technology Solutions, Deutsche Telekom, Fujitsu, HCL, HP Enterprise, IBM, Infosys, NTT, Optiv, SecureWorks, Tata Communications, Verizon and Wipro. The company is also founding member of the Cloud Security Alliance (CSA). For more information, please visit www.qualys.com

Qualys Support

Qualys is committed to providing you with the most thorough support. Through online documentation, telephone help, and direct email support, Qualys ensures that your questions will be answered in the fastest time possible. We support you 7 days a week, 24 hours a day. Access online support information at www.qualys.com/support/

About Container Security Documentation

This document provides information about using the Qualys Container Security API.

For information on using the Container Security UI to monitor vulnerabilities in Images, Containers, and Registries, refer to the Qualys Container Security User Guide.

For information on deploying the sensor on MAC, CoreOS, and various orchestrators and cloud environments, refer to the Qualys Container Sensor Deployment Guide.

For information on deploying the sensor in CI/CD environments refer to:

Qualys Container Scanning Connector for Jenkins
Qualys Container Scanning Connector for Bamboo
Get Started

Accessing the APIs

All features of Container Security are available through REST APIs. Equivalent Rest API request for each tab is provided on the UI.

In the API response,

- `associatedContainersCount` shows count of containers in RUNNING or STOPPED state.
- `associatedHostsCount` shows count of hosts where Qualys sensor AND the image is installed.

Permissions required to use APIs

- User must have the Container module enabled
- User must have API ACCESS permission

Qualys API URLs

Container Security supports both API server URLs and API gateway URLs for API requests. The Qualys API server or gateway URL you should use for API requests depends on the Qualys platform where your account is located.

Click here to identify your Qualys platform and get the API URL
Authentication for gateway URLs

You must authenticate to the Qualys Cloud Platform using Qualys account credentials (user name and password) and get the JSON Web Token (JWT) before you can start using the Gateway URLs. Use the Qualys Authentication API to get the JWT.

For example,

```
curl -X POST https://gateway.qg1.apps.qualys.com/auth -d
"username=value1&password=passwordValue&token=true" -H "Content-Type: application/x-www-form-urlencoded"
```

where gateway.qg1.apps.qualys.com is the base URL to the Qualys API server where your account is located.

- **username** and **password** are the credentials of the user account for Container Security
- **token** should be true
- **Content-Type** should be "application/x-www-form-urlencoded"

The Authentication API returns a JSON Web Token (JWT) which you can use for authentication during Container Security API calls. The token expires in 4 hours. You must regenerate the token to continue using the Container Security API.

Container Security API documentation for Gateway URLs is available at:

```
https://<Qualys_Gateway_URL>/apidocs/csapi/v1.2
```

For example, if your account is on US Platform 1

```
https://gateway.qg1.apps.qualys.com/apidocs/csapi/v1.2
```

Where’s the Swagger UI?

Click **Rest Reference** in the “Equivalent REST request” dialog to launch the Swagger UI, where you can try out the Rest APIs.
You can directly access the Swagger UI from the following URL

https://<QualysURL>/csapi/swagger-ui.html

For example, if your account is on US Platform 2


Do I need to Authenticate to use Swagger?

Authentication to the Qualys Cloud Platform is necessary before you try out the APIs. To get started, click Authorize and provide your user name and password.

API Rate Limiting

The Qualys Container Security API is enforcing limits on the number of API calls a customer can make based on the API endpoint being called and the customer's Qualys platform. API rate limits are currently enforced for Gateway API calls made by customers on US Cloud Platform 2 (https://gateway.qg2.apps.qualys.com) and will be enforced on other Qualys platforms soon. The API rate limits are enforced uniformly across all subscriptions on a particular platform. There currently is no ability to enforce custom rate limits for a given subscription.

How it works

When an API call is received, Qualys checks the rate limit defined for the API endpoint. If the rate limit has been exceeded the API call is blocked and an error is returned.

For each API we've defined the following settings:

Rate Limit Size (per API): The maximum number of API calls allowed within the subscription during the rate limit period. Provided in the response header ‘X-RateLimit-Limit’.

Rate Limit Period (in seconds, per API): The period of time that defines a window when API calls are counted within the subscription for each API. The window starts from the moment each API call is received by the service. Provided in the response header ‘X-RateLimit-Window-Sec’.

Rate Limit Remaining (per API): The remaining number of calls within the rate limit time period. Provided in the response header ‘X-RateLimit-Remaining’.
Rate limits defined per API endpoint

See the table below to understand the rate limits defined for Container Security API endpoints. Rate limits do not currently apply to the Container Runtime Security API.

**represents any API endpoint that matches this path unless otherwise noted

<table>
<thead>
<tr>
<th>API Endpoint Path (currently vxx = v1.2 or v1.3)</th>
<th>Rate Limit Size (max number of API calls)</th>
<th>Rate Limit Period (in seconds)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/csapi/vxx/containers/list</td>
<td>120</td>
<td>60</td>
<td>Every 60 seconds, you can make 120 calls to the API</td>
</tr>
<tr>
<td>/csapi/vxx/containers/**</td>
<td>5000</td>
<td>60</td>
<td>Every 60 seconds, you can make 5000 calls to the API</td>
</tr>
<tr>
<td>/csapi/vxx/images/list</td>
<td>120</td>
<td>60</td>
<td>Every 60 seconds, you can make 120 calls to the API</td>
</tr>
<tr>
<td>/csapi/vxx/images/**</td>
<td>5000</td>
<td>60</td>
<td>Every 60 seconds, you can make 5000 calls to the API</td>
</tr>
<tr>
<td>/csapi/vxx/registry/**</td>
<td>1000</td>
<td>60</td>
<td>Every 60 seconds, you can make 1000 calls to the API</td>
</tr>
<tr>
<td>/csapi/vxx/sensors/**</td>
<td>1000</td>
<td>60</td>
<td>Every 60 seconds, you can make 1000 calls to the API</td>
</tr>
</tbody>
</table>

When the rate limit is reached

The API response “429 Too Many Requests” is returned anytime a user makes an API call and the rate limit for the API endpoint has already been reached. In other words, the rate limit size has already been reached for the rate limit period. Here’s an example:

**API request:**


**Response:**

Response header

HTTP/1.1 429 Too Many Requests
Server: nginx/1.19.1
Date: Thu, 17 Dec 2020 17:51:57 GMT
Content-Length: 0
Connection: keep-alive
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer
**API output shows dates in Unix time (Epoch time)**

You’ll notice dates/times that appear in several API responses appear in Unix time, also known as Epoch time, in milliseconds. You can use an online tool to convert the Unix timestamp into a human-readable date/time.

For example, 1536134457859 converts to September 5, 2018 8:00:57.859 AM GMT.

**Kubernetes cluster attributes in API output**

We added the collection of Kubernetes cluster attributes starting in Container Security version 1.10, and made this information searchable in the UI. Kubernetes cluster attributes include node details, pod details, controller details and more. Use Container Security APIs to see Kubernetes cluster attributes collected for containers and sensors.

**Important** - Kubernetes attributes will only be processed for containers discovered after the version 1.10 release. Kubernetes attributes are collected as part of container inspect processing when containers are discovered for the first time. To fetch Kubernetes cluster attributes for an existing deployment in Kubernetes, you will have to “rollout restart” the existing deployment, which will create new containers and this will start the container inspect processing. Kubernetes attributes will get collected for the newly created containers on Kubernetes clusters.

Use the following command for the "rollout restart":

```
kubectl rollout restart deployment <deployment-name> -n <namespace>
```

**API output**

Kubernetes cluster attributes appear in the API output for these Container Security APIs

- Fetch container details
- Fetch a detailed containers list
- Fetch sensor details

You’ll see these attributes in the API output, when available:

- Cluster type (Kubernetes)
- Cluster version
- Project name (collected for projects in Google Cloud Platform)
- Node name and flag indicating whether the node is the master node
- Pod name
- Pod UUID
- Pod namespace
- Pod labels (key and value pairs)
- Controller name
- Controller UUID
- Controller type (e.g. DaemonSet, Deployment, ReplicaSet, etc)
Containers

Here is the list of the APIs we currently support for containers:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show a list of containers in your account</td>
<td>GET</td>
<td>/csapi/v1.3/containers</td>
</tr>
<tr>
<td>Show details of a container</td>
<td>GET</td>
<td>/csapi/v1.3/containers/{containerSha}</td>
</tr>
<tr>
<td>Show a list of containers in your account with container details</td>
<td>GET</td>
<td>/csapi/v1.3/containers/list</td>
</tr>
<tr>
<td>Show software installed on a container</td>
<td>GET</td>
<td>/csapi/v1.3/containers/{containerId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for a container</td>
<td>GET</td>
<td>/csapi/v1.3/containers/{containerId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for a container</td>
<td>GET</td>
<td>/csapi/v1.3/containers/{containerId}/vuln/count</td>
</tr>
<tr>
<td>Delete containers in your account</td>
<td>DELETE</td>
<td>/csapi/v1.3/containers</td>
</tr>
</tbody>
</table>

Samples for various operations on containers:

- Fetch a list of containers in your account
- Fetch container details
- Fetch a detailed containers list
- Fetch a list of software installed on a container
- Fetch vulnerability details for a container
- Fetch vulnerability count for a container
- Delete containers in your account
## Fetch a list of containers in your account

/v1.3/containers

### [GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Optional) The page to be returned. Page numbers start with 1. For API v1.1 and v1.2, pageNumber and pageNo are both supported. If both are specified in the same request, then pageNumber takes precedence. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example created:desc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

### API request:

```sh
```

### Response:

```json
{
  "data": [
    {
      "uuid": "2c8d6485-0c6d-3d7b-a2bd-86616ff78205",
      "sha": "27723ada671ea9f25624a9ffdadde273038e4a48373bada6133371cb3bd7a9d3",
      "imageId": "27723ada671e",
      "repo": [
        {
          "registry": "docker.io",
          "repository": "qualysdemo/checkoutapp",
          "tag": "demo"
        }
      ],
      "repoDigests": [
        {
          "registry": "docker.io",
          "repository": "qualysdemo/checkoutapp",
```
"digest": "6d0f0a22ba1768eb6eb30dabba5d856fc5536609ec12fe2237be7c2a7d47c9b",
"created": 1507592726000,
"updated": 1603767703217,
"associatedContainersCount": 0,
"associatedHostsCount": 1,
"lastScanned": null,
"size": 718071042,
"vulnerabilities": {
  "severity5Count": 0,
  "severity4Count": 0,
  "severity3Count": 0,
  "severity2Count": 0,
  "severity1Count": 0
},
"registryUuid": null,
"source": [
  "GENERAL"
],
"isDockerHubOfficial": false,
"isInstrumented": false,
"instrumentedFrom": null,
"instrumentationState": null,
"instrumentationErrors": null,
"scanType": null,
"scanErrorCode": null,
"scanStatus": null,
"lastFoundOnHost": {
  "sensorUuid": "cb9fa762-b161-43bb-9268-ebaa5f606af",
  "uuid": null,
  "hostname": "Test HostName",
  "ipAddress": "254.254.254.254",
  "lastUpdated": "2020-03-19T11:01:08.907Z"
},
"compliance": {
  "passCount": 3,
  "failCount": 1,
  "errorCount": 1
},
"lastComplianceScanned": 1507592707000
}
Fetch container details

/v1.3/containers/{containerSha}

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerSha</td>
<td>Specify the SHA value of a specific container in the user's scope.</td>
</tr>
</tbody>
</table>

API request:

```bash
curl -X GET 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/containers/fe23f264e3dd2b5ebca5a9aaccdf5e4ac0d3ba3144bf097f6c5ba69e00d5fed' --header 'Authorization: Bearer <token>'
```

Response:

```json
{
  "portMapping": [
    {
      "protocol": "tcp",
      "port": 5000,
      "hostIp": "0.0.0.0",
      "hostPort": 5000
    }
  ],
  "imageId": "5c4008a25e05",
  "created": "1615380901000",
  "updated": "161544071646",
  "label": null,
  "uuid": "2866748c-acf2-3d09-8384-8ab0ed522048",
  "sha": "fe23f264e3dd2b5ebca5a9aaccdf5e4ac0d3ba3144bf097f6c5ba69e00d5fed",
  "privileged": false,
  "path": "/entrypoint.sh",
  "imageSha": "5c438a25e0565faf88dd3f060a8d7688f8c41c539a3ee9c693b05f08a12ebf9",
  "macAddress": "02:42:ac:11:00:02",
  "customerUuid": "74d66479-27e6-73bf-8033-34037f109fc9",
  "ipv4": "10.10.10.10",
  "ipv6": null,
  "name": "registry",
  "host": {
    "sensorUuid": "07ef65f6-472e-4d4c-9383-591b551f87ab",
    "hostname": "cloudagent",
    "ipAddress": "10.11.11.11",
    "uuid": "1fc95c1-f881-4574-a452-febc3f55809a",
    "lastUpdated": "2021-03-10T12:56:01.213Z"
  }
}
```
"state": "STOPPED",
"imageUuid": "bfbd86f3-e99a-32da-b73d-5d1c0cb77eaa",
"containerId": "fe14f222e3dd",
"stateChanged": "1615440815994",
"services": null,
"users": null,
"operatingSystem": null,
"lastScanned": null,
"source": "GENERAL",
"isInstrumented": null,
"environment": {
  "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
},
"arguments": [
  "/etc/docker/registry/config.yml"
],
"command": "/etc/docker/registry/config.yml",
"drift": null,
"vulnerabilities": null,
"softwares": null,
"isDrift": false,
"isRoot": true,
"lastComplianceScanned": null,
"cluster": {
  "type": "kubernetes",
  "version": "v1.1.1",
  "project": "k8s-project",

  "node": {
    "name": "k8s-node",
    "isMaster": true
  },
  "pod": {
    "name": "pod-name",
    "uuid": "63225d4f-bf0c-3488-8144-89c7d03dfacf",
    "namespace": "cs",
    "label": [
      {
        "key": "com.docker.compose.container-number",
        "value": "1"
      },
      {
        "key": "com.docker.compose.service",
        "value": "lb"
      }
    ]
  },
  "controllers": [
    {
      "uuid": "a3145d4f-bf0c-3488-8144-89c7d03dfacf",
      "name": "deployment-name",
      "type": "DEPLOYMENT"
    },
    {
      "uuid": "b3145d4f-bf0c-3488-8144-89c7d03dfacf",
      "name": "service-name",
      "type": "SERVICE"
    }
  ]
},
"services": [
  {
    "name": "service-name",
    "uuid": "e3145d4f-bf0c-3488-8144-89c7d03dfacf",
    "type": "SERVICE"
  }
]}
Fetch a detailed containers list

/v1.3/containers/list

[GET]

When you make your API request, you’ll use the “limit” input parameter to specify the number of records to return. For example, you can choose to return details for 10 containers at a time by specifying limit=10. The Response Header will include a unique link that you can specify in the next API request to get the next set of results. i.e. the next 10 containers. Simply take the link as is for the new request, no additional parameters are needed. When there are no more records to return, the Response Header will show a value of “null” for the link.

The details shown for each container in the list response will be the same as if you were fetching a single container, except that the “vulnerabilities” section of the response will only show qid, result and software.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Specify the number of records to include in the response. Enter a value from 1 to 250.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

**Example: List containers with limit of 2 records**

**API request:**


**Response:**

```json
{
    "content-type": "application/json; charset=UTF-8",
    "date": "Wed, 26 Aug 2020 07:56:59 GMT",
    "link": "<http://qualysapi.qualys.com/csapi/v1.3/containers/list?limit=2&pageNumber=2&paginationQuery=updated%3E%3D1594876830694+AND+not+uuid%3A+%5B65eff01b-7775-3a0a-"
```
Containers
Fetch a detailed containers list

```
bd68-e6dale7a39f5%5D>;rel=next",
"referrer-policy": "same-origin",
"server": "Qualys",
"transfer-encoding": "chunked",
"x-content-type-options": "nosniff",
"x-frame-options": "sameorigin",
"x-permitted-cross-domain-policies": "none",
"x-powered-by": "Qualys",
"x-xss-protection": "1; mode=block"
}

Response JSON:
{
  "data": [
    {
      "portMapping": null,
      "imageId": "3aabd9e6d667",
      "created": "1593759719000",
      "updated": "1594876830316",
      "label": null,
      "uuid": "cdb22685-9fbb-3971-90aa-706086a64079",
      "sha": "a249ee950e639492563f2dd7caa231adc5fc3fe5714f1d82a23cc2f8bbcdca22f",
      "privileged": false,
      "path": "/bin/sh",
      "imageSha": "3abcd9e66670283b5e2df06bb77354b379b17ac7b940e8ea9edb202d423af13",
      "macAddress": "",
      "customerUuid": "a1d17d52-03fb-c803-83a6-1048acccaca9",
      "ipv4": null,
      "ipv6": null,
      "name": "container-1",
      "host": {
        "sensorUuid": "213d6f22-f8cc-42ad-814a-49b4b5c38d64",
        "hostname": "cloudagent",
        "ipAddress": "10.11.12.13",
        "uuid": "52d08bfe-2799-4f5c-81fb-8c0950d80f5e",
        "lastUpdated": null
      },
      "state": "DELETED",
      "imageUuid": "e2c22565-d1d9-3f9c-ac83-ad9b09f110ef",
      "containerId": "a978ee930e63",
      "stateChanged": "1593759720784",
      "services": null,
      "users": null,
      "operatingSystem": null,
      "lastScanned": null,
      "source": "GENERAL",
      "isInstrumented": null,
      "environment": {
        "VAULT_VERSION=0.9.3",
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      }
    }
  ]
}
```
Containers

Fetch a detailed containers list

},
"arguments": [
  "-c",
  "sed -i 's/\r//g' /usr/local/bin/init.sh"
],
"command": "/bin/sh -c sed -i 's/\r//g' /usr/local/bin/init.sh",
"drift": null,
"vulnerabilities": null,
"softwares": null,
"isDrift": null,
"isRoot": true
],
{
  "portMapping": null,
  "imageId": "dd2f1712bc5a",
  "created": "1593759720000",
  "updated": "1594876830694",
  "label": null,
  "uuid": "62eff01b-7775-4a0a-bd68-e6db1e7a39f5",
  "sha": "c0db6c214c3e6c6d620a07e4e06da540c50c3c0c1372f84116dbe8aa03dd8d",
  "privileged": false,
  "path": "/bin/sh",
  "imageSha": "dd6d1713ac6a55dffe5faff50e7408f51f8f1108f7c4c3e8e84697becfa38b",
  "macAddress": ",",
  "customerId": "a1d17f51-05fb-c803-84a6-1048acccaca9",
  "ipv4": null,
  "ipv6": null,
  "name": "container-2",
  "host": {
    "sensorUuid": "213d6f22-f8cc-43bd-814a-49b4d5c38e64",
    "hostname": "cloudagent",
    "ipAddress": "10.11.12.13",
    "uuid": "53d08bfe-2899-4f5c-81fa-8c0950d80f5e",
    "lastUpdated": null
  },
  "state": "DELETED",
  "imageUuid": "ca5f92da-7785-344b-8681-8c64f30a043b",
  "containerId": "c0da6c314c2e",
  "stateChanged": "1593759721924",
  "services": null,
  "users": null,
  "operatingSystem": null,
  "lastScanned": null,
  "source": "GENERAL",
  "isInstrumented": null,
  "environment": {
    "VAULT_VERSION=0.9.3",
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  },
  "arguments": [
    "-c",
    ]
"sed -i 's/\r//g' /usr/local/bin/docker-entrypoint.sh",
  "command": "/bin/sh -c sed -i 's/\r//g' /usr/local/bin/docker-entrypoint.sh",
  "drift": null,
  "vulnerabilities": null,
  "softwares": null,
  "isDrift": null,
  "isRoot": true,
  "cluster": {
    "type": "kubernetes",
    "version": "v1.1.1",
    "project": "k8s-project",
    "node": {
      "name": "k8s-node",
      "isMaster": true
    },
    "pod": {
      "name": "pod-name",
      "uuid": "6325d4f-bf0c-3488-8144-89c7d03dfacf",
      "namespace": "cs",
      "label": [
        {
          "key": "com.docker.compose.container-number",
          "value": "1"
        },
        {
          "key": "com.docker.compose.service",
          "value": "lb"
        }
      ]
    }
  }
  "controllers": [
    {
      "uuid": "a3145d4f-bf0c-3488-8144-89c7d03dfacf",
      "name": "deployment-name",
      "type": "DEPLOYMENT"
    },
    {
      "uuid": "b3145d4f-bf0c-3488-8144-89c7d03dfacf",
      "name": "replicaset-name",
      "type": "REPLICASET"
    }
  ]
}
**Example 2: Get the next set of results for containers list**

In this example, the link from the Response Header from the previous API request is specified as part of the new request in order to get the next set of results. The response will be similar to the previous example.

**API request:**
```
curl -X GET 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/containers/list?limit=2&paginationQuery=updated%3E%3D1594876830694+AND+not+uuid%3A+%5B65eff01b-7775-3a0a-bd68-e6da1e7a39f5%5D' --header 'Authorization: Bearer <token>'
```

**Example 3: Response with no link**

Here's a sample response where there are no more records to return after this, so there is no link provided in the Response Header section. You’ll see a value of “null” for the link.

**Response:**
```
Response Headers:
{
    "content-type": "application/json;charset=UTF-8",
    "date": "Wed, 26 Aug 2020 07:56:59 GMT",
    "link": "null",
    "referrer-policy": "same-origin",
    "server": "Qualys",
    "transfer-encoding": "chunked",
    "x-content-type-options": "nosniff",
    "x-frame-options": "sameorigin",
    "x-permitted-cross-domain-policies": "none",
    "x-powered-by": "Qualys",
    "x-xss-protection": "1; mode=block"
}
...
```
# Fetch a list of software installed on a container

/v1.3/containers/{containerId}/software

[GET]

## Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerId</td>
<td>Specify the ID or SHA value of a specific container in the user’s scope.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. The default value is <code>name:asc</code>. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
<tr>
<td>isDrift</td>
<td>Specify <code>true</code> if you are looking for drift containers. Default is <code>false</code>.</td>
</tr>
</tbody>
</table>

## API Request:

```bash
curl -X GET
```

## Response:

```json
{
  "data": [
    {
      "name": "file",
      "version": "1:5.25-2ubuntu1.1",
      "fixVersion": "1:5.25-2ubuntu1.2",
      "vulnerabilities": {
        "severity5Count": 0,
        "severity4Count": 0,
        "severity3Count": 1,
        "severity2Count": 0,
        "severity1Count": 0
      }
    },
    {
      "name": "libgpg-error0:amd64",
      "version": "1.21-2ubuntu1",
      "fixVersion": null,
      "vulnerabilities": {
        "severity5Count": null,
        "severity4Count": null,
        "severity3Count": null,
        "severity2Count": null,
        "severity1Count": null
      }
    }
  ]
}
```
Fetch vulnerability details for a container

/v1.3/containers/{containerId}/vuln

[GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerId</td>
<td>Specify the ID or SHA value of a specific container in the user's scope.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>type</td>
<td>Specify the type of information to be fetched: Summary, Details, All.</td>
</tr>
<tr>
<td>isDrift</td>
<td>Specify true is you are looking for drift containers. Default is false.</td>
</tr>
</tbody>
</table>
API request:
```
curl -X GET
```

Response:
```
{
  "details": {
    "vulns": [
      {
        "vulnerability": null,
        "result": "libudev1 229-4ubuntu21.16 229-4ubuntu21.21
libsystemd0 229-4ubuntu21.16 229-4ubuntu21.21 \nsystemd 229-4ubuntu21.16 229-4ubuntu21.21 \nsystemd-sysv 229-4ubuntu21.16 229-4ubuntu21.21",
        "lastFound": "1557343689800",
        "firstFound": "1557318183463",
        "fixed": null,
        "severity": 3,
        "customerSeverity": 3,
        "port": null,
        "typeDetected": "CONFIRMED",
        "status": null,
        "nonRunningKernel": null,
        "nonExploitableConfig": null,
        "runningService": null,
        "risk": 30,
        "category": "Ubuntu",
        "os": null,
        "discoveryType": [
          "AUTHENTICATED"
        ],
        "authType": [
          "UNIX_AUTH"
        ],
        "supportedBy": [
          "VM",
          "CA-Linux Agent"
        ],
        "product": [
          "None"
        ],
        "vendor": [
          "ubuntu"
        ],
        "cveids": [
          "CVE-2019-3842"
        ],
        "threatIntel": {
          "activeAttacks": null,
          "zeroDay": null,
          "publicExploit": true,
```
"highLateralMovement": null,
"easyExploit": null,
"highDataLoss": null,
"noPatch": null,
"denialOfService": null,
"malware": null,
"exploitKit": null,
"publicExploitNames": null,
"malwareNames": null,
"exploitKitNames": null
},
"qid": 197424,
"title": "Ubuntu Security Notification for systemd vulnerability (USN-3938-1)",
"cvssInfo": {
  "baseScore": "4.4",
  "temporalScore": "3.4",
  "accessVector": "Local"
},
"cvss3Info": {
  "baseScore": "7.0",
  "temporalScore": "6.3"
},
"patchAvailable": true,
"published": 1554890302000,
"ageInDays": 41,
"software": [
  {
    "name": "libudev1:amd64",
    "version": "229-4ubuntu21.16",
    "fixVersion": "229-4ubuntu21.21",
    "vulnerabilities": null
  },
  {
    "name": "systemd-sysv",
    "version": "229-4ubuntu21.16",
    "fixVersion": "229-4ubuntu21.21",
    "vulnerabilities": null
  },
  {
    "name": "libsystemd0:amd64",
    "version": "229-4ubuntu21.16",
    "fixVersion": "229-4ubuntu21.21",
    "vulnerabilities": null
  },
  {
    "name": "systemd",
    "version": "229-4ubuntu21.16",
    "fixVersion": "229-4ubuntu21.21",
    "vulnerabilities": null
  }
]
```
],
"driftVulns": null
},
"vulnSummary": {
  "confirmed": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 1,
    "sev4Count": 0,
    "sev3Count": 3
  },
  "potential": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 1,
    "sev4Count": 0,
    "sev3Count": 0
  }
},
"patchAvailability": {
  "confirmed": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 0,
    "sev4Count": 0,
    "sev3Count": 3
  },
  "potential": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 0,
    "sev4Count": 0,
    "sev3Count": 0
  }
}
```

Fetch vulnerability count for a container

/v1.3/containers/{containerId}/vuln/count

[GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerId</td>
<td>Specify the ID or SHA value of a specific container in the user's scope.</td>
</tr>
</tbody>
</table>
API request:

curl -X GET

Response:

{
  "severity5Count": 0,
  "severity3Count": 3,
  "severity4Count": 0,
  "severity1Count": 0,
  "severity2Count": 2
}

Delete containers in your account

/v1.3/containers

[DELETE]

You can choose to provide a request body without parameters or you can specify UUIDs or filter as input parameters in the API request. See samples.

When specifying request body:

Request Body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerDeleteRequest</td>
<td>(Required) user filters to query containers or provide one or more container UUIDs to delete. Filter can be applied by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

When specifying input parameters as part of path:

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerIds</td>
<td>One or more container UUIDs for the containers you want to delete. When specifying multiple containers in the same request, enter them in this way: containerIds=value1&amp;containerIds=value2&amp;containerIds=value3, and so on.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>
Containers
Delete containers in your account

Sample 1 - Use request body to specify containers to delete
In this sample, a request body is used as part of the API request.

API request:
curl -X DELETE
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/containers' -d
'{"containerIds":\"["a6025a31-bd86-37e6-9de7-5722af586b66", "c4032e71-5969-34a9-a8a9-ba2b69729673"]\"} --header 'Authorization: Bearer <token>'

Response:
Returns ("deletionJobId":"413b076e-01a8-4780-8e62-875b615a9a1f")
response code 200

Sample 2 - Delete single container using container UUID
In this sample, we’ll delete a single container by specifying the container UUID.

API request:
curl -X DELETE

Response:
{
  "deletionJobId": "951dca63-254b-4f5c-ab76-7671dd8f1528"
}

Sample 3 - Delete multiple containers using container UUIDs
In this sample, we’ll delete 2 containers in the same request. Specify multiple containers by entering containerIds=value1&containerIds=value2, and so on.

API request:
curl -X DELETE
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/containers?containerIds=68cc933e-4994-3d9b-8232-b1c78b3b3c9d&containerIds=4e7d40ce-6ddc-3fb7-ab55-e0b48a60208a' --header 'Authorization: Bearer <token>'

Response:
{
  "deletionJobId": "2b38a127-b413-4aa7-8421-61760487722c"
}
Sample 4 - Delete containers using a filter
In this sample, we’ll delete containers based on the filter parameter.

API request:
```
```

Response:
```
{
  "deletionJobId": "8a47532f-4f22-46e5-b772-58f9664173fe"
}
```
# Images

Here is the list of the APIs we currently support for images:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show a list of images in your account</td>
<td>GET</td>
<td>/csapi/v1.3/images</td>
</tr>
<tr>
<td>Show details of a single image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageSha}</td>
</tr>
<tr>
<td>Show a list of images in your account with image details</td>
<td>GET</td>
<td>/csapi/v1.3/images/list</td>
</tr>
<tr>
<td>Show associations for an image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageId}/association</td>
</tr>
<tr>
<td>Show repositories that contain this image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageId}/repos</td>
</tr>
<tr>
<td>Show software installed on an image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for an image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for an image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageId}/vuln/count</td>
</tr>
<tr>
<td>Delete images in your account</td>
<td>DELETE</td>
<td>/csapi/v1.3/images</td>
</tr>
</tbody>
</table>

**Samples for various operations on images:**

- Fetch a list of images in your account
- Fetch image details
- Fetch a detailed images list
- Fetch associations for an image
- Fetch a list of repositories that contain this image
- Fetch a list of software installed on an image
- Fetch vulnerability details for an image
- Fetch vulnerability count for an image
- Delete images in your account
Fetch a list of images in your account
/v1.3/images

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Optional) The page to be returned. Page numbers start with 1. For API v1.1 and v1.2, pageNumber and pageNo are both supported. If both are specified in the same request, then pageNumber takes precedence. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>[Required] The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example eventOccurred:desc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:


Response:

```json
{
  "data": [
    {
      "uuid": "2c8d6485-0c6d-3d7b-a2bd-86616ff78205",
      "sha": "27723ada671e9f25624a9ffdadde273038e4a48373bada6133371cb3bd7a9d3",
      "imageId": "27723ada671e",
      "repo": [
        {
          "registry": "docker.io",
          "repository": "qualysdemo/checkoutapp",
          "tag": "demo"
        }
      ],
      "repoDigests": [
        {
          "registry": "docker.io",
          "repository": "qualysdemo/checkoutapp",
```
"digest": "6d0f0a22ba1768ebed30dabba5d856fc5536609ec12fc2b23e7bec7a79ccd9b"
],
  "created": 1507592726000,
  "updated": 1603767703217,
  "associatedContainersCount": 0,
  "associatedHostsCount": 1,
  "lastScanned": null,
  "size": 718071042,
  "vulnerabilities": {
    "severity5Count": 0,
    "severity4Count": 0,
    "severity3Count": 0,
    "severity2Count": 0,
    "severity1Count": 0
  },
  "registryUuid": null,
  "source": [
    "GENERAL"
  ],
  "isDockerHubOfficial": false,
  "isInstrumented": false,
  "instrumentedFrom": null,
  "instrumentationState": null,
  "instrumentationErrors": null,
  "scanType": null,
  "scanErrorCode": null,
  "scanStatus": null,
  "lastFoundOnHost": {
    "sensorUuid": "cb9fa762-b161-43bb-9268-ebae5fc606af",
    "uuid": null,
    "hostname": "Test HostName",
    "ipAddress": "254.254.254.254",
    "lastUpdated": "2020-03-19T11:01:08.907Z"
  },
  "compliance": {
    "passCount": 3,
    "failCount": 1,
    "errorCount": 1
  },
  "lastComplianceScanned": 1507592707000
}
]}
"groups": []
}
Fetch image details
/v1.3/images/{imageSha}

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageSha</td>
<td>Specify the SHA value of a specific image in the user's scope.</td>
</tr>
</tbody>
</table>

API request:

```bash
curl -X GET 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/c64844065dcbbc3d0a90c365c1f56421766a5cebf05f7ecbd3377af410fff09fd' --header 'Authorization: Bearer <token>'
```

Response:

```json
{
    "created": "1603477517000",
    "updated": "1605017537578",
    "author": "Couchbase Docker Team <docker@couchbase.com>",
    "repo": [
    {
        "registry": "docker.io",
        "tag": "latest",
        "repository": "couchbase"
    }
    ],
    "repoDigests": [
    {
        "registry": "docker.io",
        "digest": "1d81b3c38283f70f0ccf02371a12d3671c1d5175bccc67e8c2a5c0bf48f976",
        "repository": "couchbase"
    }
    ],
    "label": null,
    "uuid": "5d48f83b-cddb-33ac-8fad-e8452dd116b1",
    "sha": "c64844065dcbbc3d0a90c365c1f56421766a5cebf05f7ecbd3377af410fff09fd",
    "operatingSystem": "Ubuntu Linux 16.04.7",
    "customerUuid": "192cc974-1e44-cb6c-806e-f78f6441cb0d",
    "dockerVersion": "18.09.7",
    "size": 1183790011,
    "layers": [
    {
        "size": "130553983",
        "createdBy": "ADD file:c1f3147c7b6710af5affd417ff822ee28df872d716003858d3d2e23d2277c981 in /",
    }
    ]
}
```
"created": "1603474388000",
"comment": "",
"id": null,
"sha": null,
"tags": null
},

{
"size": "0",
"createdBy": "rm -rf /var/lib/apt/lists/*",
"created": "1603474389000",
"comment": "",
"id": null,
"sha": null,
"tags": null
},

{
"size": "1930",
"createdBy": "COPY file:d816a67f62bfba76d2812cefeb9e9252afa13f3852775c3e68599df7741e90cb7 in /",
"created": "1603477517000",
"comment": "",
"id": null,
"sha": null,
"tags": null
}
],
"host": [ 
{
"sensorUuid": "fed79006-2fa9-4b67-8f5a-272b4e02f084",
"hostname": "host.qualys.com",
"ipAddress": "10.44.29.40",
"uuid": "6ba5be85-2758-4f44-814a-b690c9ed23ee",
"lastUpdated": "2020-11-10T14:10:29.218Z"
} 
],
"architecture": "amd64",
"imageId": "c64844065dcb",
"lastScanned": "1605017537578",
"registryUuid": null,
"source": [ 
"GENERAL"
],
"totalVulCount": "0",
"users": [ 
"root"
],
"isDockerHubOfficial": null,
"isInstrumented": null,
"instrumentedFrom": null,
"instrumentationState": null,
"scanType": "DYNAMIC",
"scanErrorCode": null,
"scanStatus": "SUCCESS"
Fetch a detailed images list

/v1.3/images/list

[GET]

When you make your API request, you’ll use the “limit” input parameter to specify the number of records to return. For example, you can choose to return details for 10 images at a time by specifying limit=10. The Response Header will include a unique link that you can specify in the next API request to get the next set of results. i.e. the next 10 images. Simply take the link as is for the new request, no additional parameters are needed. When there are no more records to return, the Response Header will show a value of “null” for the link.

The details shown for each image in the list response will be the same as if you were fetching a single image, with a few exceptions. The “vulnerabilities” section of the response will only show qid, result and software. Instead of listing all hosts where an image was found, we’ll show details for the most recent host where the image was found. This appears in the response under “lastFoundOnHost”.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

---

"lastFoundOnHost": {
  "sensorUuid": "fed79006-2fa9-4b67-8f5a-272b4e02f084",
  "hostname": "host.qualys.com",
  "ipAddress": "10.44.29.40",
  "uuid": "6ba5be85-2758-4f44-814a-b690c9ed23ee",
  "lastUpdated": "2020-11-10T14:10:29.218Z"
},
"softwares": [
  {
    "name": "libncursesw5:amd64",
    "version": "6.0+20160213-1ubuntu1",
    "fixVersion": null,
    "vulnerabilities": null
  },
  {
    "name": "libgpg-error0:amd64",
    "version": "1.21-2ubuntu1",
    "fixVersion": null,
    "vulnerabilities": null
  }
],
"vulnerabilities": [],
"lastComplianceScanned": "1603477517000"
### Limit

Specify the number of records to include in the response. Enter a value from 1 to 250.

### Filter

Filter the images list by providing a query using Qualys syntax. Refer to "How to Search" topic in the online help for assistance with creating your query.

---

#### Example: List images with limit of 2 records

**API request:**

```bash
```

**Response:**

**Response Headers:**

```json
{
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 26 Aug 2020 07:41:21 GMT",
  "link": "<http://qualysapi.qualys.com/csapi/v1.3/images/list?limit=2&paginationQuery=updated%3E%3D1593731505425+AND+not+uuid%3A+%5Bbc092a61-caee-3ff1-a693-d9062361ad3c%5D>;rel=next",
  "referrer-policy": "same-origin",
  "server": "Qualys",
  "transfer-encoding": "chunked",
  "x-content-type-options": "nosniff",
  "x-frame-options": "sameorigin",
  "x-permitted-cross-domain-policies": "none",
  "x-powered-by": "Qualys",
  "x-xss-protection": "1; mode=block"
}
```

**Response JSON:**

```json
{
  "data": [
    {
      "created": "155753223000",
      "updated": "1593731440059",
      "author": "",
      "repo": [
        {
          "registry": "docker.io",
          "tag": "3.9",
          "repository": "alpine"
        }
      ],
      "repoDigests": [
        {
          "registry": "docker.io",
          "digest": "sha256:...."
        }
      ]
    }
  ]
}
```

---

35
"7746df395af22f04212cd25a92cd1d6db5a06a0ca9579a229ef43008d4d1302a",
  "repository": "alpine"
}
],
"label": null,
"layersCount": null,
"uuid": "cfd55a5d-28fd-39c3-9572-50b05b89bd1a",
"sha": "055936d3920576da37aa9bc460d70c5f212028bdalc0b8c0879aef03d7a66ea1",
"operatingSystem": "Alpine Linux 3.9.4",
"sensorUuid": null,
"customerUuid": "a1d17d51-03fb-c803-81a6-1048acccaca9",
"dockerVersion": "18.06.1-ce",
"size": 5533335,
"layers": null,
"architecture": "amd64",
"imageId": "055936d39205",
"lastScanned": "1593731440058",
"registryUuid": null,
"source": [ "GENERAL"
],
"users": [ "root",
  "operator",
  "postgres"
],
"lastFoundOnHost": null,
"isDockerHubOfficial": null,
"isInstrumented": null,
"instrumentedFrom": null,
"instrumentationState": null,
"scanType": "DYNAMIC",
"softwares": [ {
  "name": "musl",
  "version": "1.1.20-r4",
  "fixVersion": null
},
{ "name": "libtls-standalone",
  "version": "2.7.4-r6",
  "fixVersion": null
},
{ "name": "alpine-keys",
  "version": "2.1-r1",
  "fixVersion": null
},
...
],
"vulnerabilities": [ 

"qid": 105144,
"result": "shadow:x:42:\n\n/etc/shadow\n\n-rw-r-----    1 root
shadow         441 May  9  2019 /etc/shadow",
"software": null
]}

"created": "1591815652000",
"updated": "1593731505425",
"author": "",
"repo": [
{
  "registry": "docker.io",
  "tag": "1",
  "repository": "hello_world_ps"
}
],
"repoDigests": null,
"label": [
{
  "key": "org.label-schema.name",
  "value": "CentOS Base Image"
},
{
  "key": "org.label-schema.license",
  "value": "GPLv2"
}
...],
"layersCount": null,
"uuid": "bc092a61-caee-3ff1-a693-d9062361ad3c",
"sha": "016ad20049fb86f4f027cc93efecce7a9c546ee5a30139b1d62c225161954581",
"operatingSystem": "CentOS Linux 7.7.1908",
"sensorUuid": null,
"customerUuid": "a1d17d51-03fb-c803-81a6-1048acccaca9",
"dockerVersion": "19.03.5",
"size": 457819642,
"layers": null,
"architecture": "amd64",
"imageId": "016ad20049fb",
"lastScanned": "1593731505424",
"registryUuid": null,
"source": [
  "GENERAL"
],
"users": [
  "root"
],
"lastFoundOnHost": null,
"isDockerHubOfficial": null,
"isInstrumented": null,
"instrumentedFrom": null,
"instrumentationState": null,
"scanType": "DYNAMIC",
"softwares": [
{
"name": "util-linux",
"version": "2.23.2-61.el7",
"fixVersion": null
},
{
"name": "tar",
"version": "1.26-35.el7",
"fixVersion": null
},
...],
"vulnerabilities": [
{
"gid": 256906,
"result": "#table cols="3"\nPackage Installed_Version
Required_Version\nnbind-license 9.11.4-16.P2.el7.noarch 9.11.4-16.P2.el7__8.6",
"software": [
{
"name": "bind-license",
"version": "9.11.4-16.P2.el7",
"fixVersion": "9.11.4-16.P2.el7__8.6"
}
]
}
],
"limit": 2

Example 2: Get the next set of results for images list
In this example, the link from the Response Header from the previous API request is specified as part of the new request in order to get the next set of results. The response will be similar to the previous example.

API request:
curl -X GET
ationQuery=updated%3E%3D1593731505425+AND+not+uuid%3A+%5Bbc092a61-cae3
3ff1-a693-d9062361ad3c%5D' --header 'Authorization: Bearer <token>'

Example 3: Response with no link
Here's a sample response where there are no more records to return after this, so there is no link provided in the Response Header section. You'll see a value of "null" for the link.
Fetch associations for an image

/v1.3/images/{imageId}/association

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageId</td>
<td>Specify the ID or SHA value of a specific image in the user’s scope.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the images list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>type</td>
<td>Specify the type of information to be fetched: Container, Host, Drift, All.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/{imageId}/association?imageId=5d556c82899c&type=ALL' --header 'Authorization: Bearer <token>'

Response:

```
{
    "containers": [],
    "driftContainers": [],
    "hosts": [
    {
      "sensorUuid": "2a9726f2-69d1-4255-b6ba-2f4d0c7bb596",
      "hostname": "qualys-virtual-machine",
      "ipAddress": "10.115.67.98",
    }
```
"uuid": null,
"runningContainerCount": null,
"stoppedContainerCount": null,
"createdContainerCount": null,
"pausedContainerCount": null,
"runningContainerCountForAssociatedImage": null,
"stoppedContainerCountForAssociatedImage": null,
"createdContainerCountForAssociatedImage": null,
"pausedContainerCountForAssociatedImage": null
}
}

Fetch a list of repositories that contain this image

/v1.3/images/{imageId}/repos

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>imageId</strong></td>
<td>Specify the ID or SHA value of a specific image in the user's scope.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/30def87e0731/repos'
  --header 'Authorization: Bearer <token>'
```

Response:

```
{
    "registryUuid": "f3927b96-dbb4-473b-bf10-18460c97736c",
    "registry": "registrytest2.azurecr.io",
    "repo": "couchbase",
    "tag": "latest"
}
```
Fetch a list of software installed on an image

/v1.3/images/{imageId}/software

[GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageId</td>
<td>Specify the ID or SHA value of a specific image in the user's scope.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the images list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example <code>qid:asc</code>. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

**API request:**

```bash
curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/5d556c82899c/software'
--header 'Authorization: Bearer <token>'
```

**Response:**

```json
{
  "data": [
  {
    "name": "perl-base",
    "version": "5.26.1-6ubuntu0.3",
    "fixVersion": null,
    "vulnerabilities": {
      "severity5Count": null,
      "severity4Count": null,
      "severity3Count": null,
      "severity2Count": null,
      "severity1Count": null
    }
  },
  {
    "name": "hostname",
    "version": "3.20",
    "fixVersion": null,
    "vulnerabilities": {
      "severity5Count": null,
      "severity4Count": null,
      "severity3Count": null,
      "severity2Count": null,
      "severity1Count": null
    }
  },
  ...

  ...

  ...
```


Fetch vulnerability details for an image

/v1.3/images/{imageId}/vuln

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageId</td>
<td>Specify the ID or SHA value of a specific image in the user’s scope.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the images list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>type</td>
<td>Specify the type of information to be fetched: Summary, Details, All.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example qid:asc. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

```bash
curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/5d556c82899c/vuln?
type=ALL&sort=qid%3Aasc' --header 'Authorization: Bearer <token>'
```

Response:

```json
{
  "details": [
    {
      "vulnerability": null,
      "result": "sysct1 net.ipv4.ip_forward\nnet.ipv4.ip_forward = 1",
      "lastFound": "1557406965964",
      "firstFound": "1557406965964",
      "fixed": null,
      "severity": 2,
      "customerSeverity": 2,
      "port": null,
      "typeDetected": "POTENTIAL",
      "status": null,
```
"nonRunningKernel": null,
"nonExploitableConfig": null,
"runningService": null,
"risk": 20,
"category": "Local",
"os": null,
"discoveryType": [
  "AUTHENTICATED"
],
"authType": [
  "UNIX_AUTH"
],
"supportedBy": [
  "VM",
  "CA-Linux Agent",
  "CA-Mac Agent"
],
"product": [],
"vendor": [],
"cveids": [
  "CVE-1999-0511"
],
"threatIntel": {
  "activeAttacks": null,
  "zeroDay": null,
  "publicExploit": null,
  "highLateralMovement": null,
  "easyExploit": true,
  "highDataLoss": null,
  "noPatch": true,
  "denialOfService": null,
  "malware": null,
  "exploitKit": null,
  "publicExploitNames": null,
  "malwareNames": null,
  "exploitKitNames": null
},
"qid": 115284,
"title": "IP Forwarding Enabled",
"cvssInfo": {
  "baseScore": "7.5",
  "temporalScore": "6.8",
  "accessVector": "Network"
},
"cvss3Info": {
  "baseScore": "3.7",
  "temporalScore": "3.6"
},
"patchAvailable": false,
"published": 1127977200000,
"ageInDays": 4983,
"software": null
...
"vulnSummary": {
  "confirmed": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 1,
    "sev4Count": 0,
    "sev3Count": 0
  },
  "potential": {
    "sev1Count": 0,
    "sev5Count": 0,
    "sev2Count": 1,
    "sev4Count": 0,
    "sev3Count": 0
  },
  "patchAvailability": {
    "confirmed": {
      "sev1Count": 0,
      "sev5Count": 0,
      "sev2Count": 0,
      "sev4Count": 0,
      "sev3Count": 0
    },
    "potential": {
      "sev1Count": 0,
      "sev5Count": 0,
      "sev2Count": 0,
      "sev4Count": 0,
      "sev3Count": 0
    }
  }
}

Fetch vulnerability count for an image

/v1.3/images/{imageId}/vuln/count

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageld</td>
<td>Specify the ID or SHA value of a specific image in the user’s scope.</td>
</tr>
</tbody>
</table>
API request:

curl -X GET

Response:

```
{
  "severity5Count": 0,
  "severity3Count": 0,
  "severity4Count": 0,
  "severity1Count": 0,
  "severity2Count": 2
}
```

Delete images in your account

/v1.3/images

[DELETE]

Images with active containers (CREATED, RUNNING, STOPPED, PAUSED) associated with them, cannot be deleted.

You can choose to provide a request body without parameters or you can specify UUIDs or filter as input parameters in the API request. See samples.

When specifying request body:

Request Body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageDeleteRequest</td>
<td>(Required) user filters to query images or provide one or more image UUIDs to delete. Filter can be applied by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

When specifying input parameters as part of path:

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageIds</td>
<td>One or more image UUIDs for the images you want to delete. When specifying multiple images in the same request, enter them in this way: imageIds=value1&amp;imageIds=value2&amp;imageIds=value3, and so on.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the images list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>
Sample 1 - Use request body to specify images to delete
In this sample, a request body is used as part of the API request.

API request:
```
'{"imageIds":["e3e4cca0-8305-3835-810a-b334dcb65a33"]}' --header
'Authorization: Bearer <token>'
```

Response:
```
Returns ("deletionJobId":"980ce235-5677-4997-81ca-3905e63471bb")
response code 200
```

Sample 2 - Delete single image using image UUID
In this sample, we’ll delete a single image by specifying the image UUID.

API request:
```
curl -X DELETE
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images?imageIds=8b261e4e-47f3-3b6a-a5a7-668ff0d6e3eb' --header 'Authorization: Bearer <token>'
```

Response:
```
{
  "deletionJobId": "ee295423-af59-4f5c-a4a1-7cb035dae61b"
}
```

Sample 3 - Delete multiple images using image UUIDs
In this sample, we’ll delete 2 images in the same request. Specify multiple images by entering imageIds=value1&imageIds=value2, and so on.

API request:
```
curl -X DELETE
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images?imageIds=8b261e4e-47f3-3b6a-a5a7-668ff0d6e3eb&imageIds=9b251e3e-52f3-2b6a-a6a7-678ff0d5e2eb' --header 'Authorization: Bearer <token>'
```

Response:
```
{
  "deletionJobId": "1b54a117-b413-4aa8-8511-61860487619c"
}
```
Sample 4 - Delete images using a filter
In this sample, we’ll delete images based on the filter parameter.

API request:
curl -X DELETE

Response:
{
   "deletionJobId": "ee295423-af59-4f5c-a4a1-7cb035da61b"
}
Here is the list of the APIs we currently support for registries:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show a list of registries in your account</td>
<td>GET</td>
<td>/csapi/v1.3/registry</td>
</tr>
<tr>
<td>Show details of a registry</td>
<td>GET</td>
<td>/csapi/v1.3/registry/[registryId]</td>
</tr>
<tr>
<td>Show a list of ACR connectors in your account</td>
<td>GET</td>
<td>/v1.3/registry/acr/ connectors</td>
</tr>
<tr>
<td>Show ACR connector details</td>
<td>GET</td>
<td>/v1.3/registry/acr/connector/ [connectorId]</td>
</tr>
<tr>
<td>Create new ACR connector</td>
<td>POST</td>
<td>/v1.3/registry/acr/connector</td>
</tr>
<tr>
<td>Fetch AWS account ID and External ID for your account</td>
<td>GET</td>
<td>/csapi/v1.3/registry/aws-base</td>
</tr>
<tr>
<td>Show a list of AWS connectors in your account</td>
<td>GET</td>
<td>/csapi/v1.3/registry/aws/ connectors</td>
</tr>
<tr>
<td>Show a list of AWS connectors for an AWS account ID</td>
<td>GET</td>
<td>/csapi/v1.3/registry/aws/connectors/[accountId]</td>
</tr>
<tr>
<td>Create new AWS connector</td>
<td>POST</td>
<td>/csapi/v1.3/registry/aws/ connector</td>
</tr>
<tr>
<td>Show a list of GCR connectors in your account</td>
<td>GET</td>
<td>/v1.3/registry/gcr/connectors</td>
</tr>
<tr>
<td>Show GCR connector details</td>
<td>GET</td>
<td>/v1.3/registry/gcr/connector/ [connectorId]</td>
</tr>
<tr>
<td>Create new GCR connector</td>
<td>POST</td>
<td>/v1.3/registry/gcr/connector</td>
</tr>
<tr>
<td>Validate information for new registry</td>
<td>POST</td>
<td>/csapi/v1.3/registry/validate</td>
</tr>
<tr>
<td>Create a new registry</td>
<td>POST</td>
<td>/csapi/v1.3/registry</td>
</tr>
<tr>
<td>Update existing registry in your account</td>
<td>PUT</td>
<td>/csapi/v1.3/registry/[registryId]</td>
</tr>
<tr>
<td>Show a list of repositories in a registry</td>
<td>GET</td>
<td>/csapi/v1.3/registry/[registryId]/repository</td>
</tr>
<tr>
<td>Show a list of schedules created for a registry</td>
<td>GET</td>
<td>/csapi/v1.3/registry/[registryId]/schedule</td>
</tr>
<tr>
<td>Create a new registry scan schedule</td>
<td>POST</td>
<td>/csapi/v1.3/registry/[registryId]/schedule</td>
</tr>
<tr>
<td>Update existing registry schedule in your account</td>
<td>PUT</td>
<td>/csapi/v1.3/registry/[registryId]/schedule/[scheduleId]</td>
</tr>
<tr>
<td>Cancel registry schedule in your account</td>
<td>POST</td>
<td>/csapi/v1.3/registry/[registryId]/schedule/[scheduleId]/cancel</td>
</tr>
<tr>
<td>Delete registry in your account</td>
<td>DELETE</td>
<td>/csapi/v1.3/registry/</td>
</tr>
<tr>
<td>Operation</td>
<td>API Method</td>
<td>API Endpoint</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Delete registry schedule in your account</td>
<td>DELETE</td>
<td>DELETE /csapi/v1.3/registry/{registryId}/schedule/{scheduleId}</td>
</tr>
<tr>
<td>Delete multiple registry schedules in your account</td>
<td>DELETE</td>
<td>DELETE /csapi/v1.3/registry/{registryId}/schedule/</td>
</tr>
</tbody>
</table>

Samples for various operations on registries:

- Fetch a list of registries in your account
- Fetch registry details
- Fetch a list of ACR connectors in your account
- Fetch ACR connector details
- Create a new ACR connector
- Fetch AWS account ID and external ID
- Fetch AWS US GovCloud account ID and external ID
- Fetch a list of AWS connectors in your account
- Fetch a list of AWS connectors for a certain account ID
- Create a new AWS connector
- Fetch a list of GCR connectors in your account
- Fetch GCR connector details
- Create a new GCR connector
- Validate registry parameters
- Create registry
- Update registry
- Fetch a list of repositories in a registry
- Fetch a list of schedules created for a registry
- Create registry schedule
- Update registry schedule
- Cancel registry schedule
- Delete registries in your account
- Delete a registry schedule
- Delete multiple registry schedules (bulk delete)
Fetch a list of registries in your account

/v1.3/registry

[GET]

Here’s sample request and output to fetch a list of registries in your account.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the registries list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1. For API v1.1 and v1.2, only pageNo is supported. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example created:desc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

**API request:**

```bash
```

**Response:**

```json
{
  "data": [
    {
      "registryUuid": "1ec77e7b-2243-49d1-ac5b-06090ff896e4",
      "registryType": "V2_PRIVATE",
      "repoCount": 2,
      "totalImages": 0,
      "totalScannedImages": 0,
      "totalVulnerableImages": 0,
      "lastScanned": "1536301443647",
      "scheduleStatusList": {
        "Completed": 3
      },
      "created": "1536237658094",
      "updated": "1536237658094",
      "dockerHubOrg": null,
      "providerType": null
    }
  ]
}
```
Fetch registry details
/v1.3/registry
[GET]
Here’s sample request and output to fetch details of a registry in your account.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) ID/UUID of the registry you want to fetch the details for.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X GET
'https://gateway.q2.apps.qualys.com/csapi/v1.3/registry/1ad23456-789f-0a12-3456-78bd901a2e34' --header 'Authorization: Bearer <token>'
```

Response:

```
{ "data": [
{ "registryUuid": "1ad23456-789f-0a12-3456-78bd901a2e34"
},...
"count": 6,
"groups": {}
}
```
Registries

Fetch a list of ACR connectors in your account

/v1.3/registry/acr/connectors

[GET]

You can get a list of ACR connectors to help you create a registry.

API request:

curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/acr/connectors'
--header 'Authorization: Bearer <token>'

Response:

{
"name": "New ACR connector",
"description": "This is a new connector for ACR",
"acrConnectorId": "025b5e82-0338-4920-b091-6c028b4b39a4"
},
{
"name": "New ACR connector 2",
"description": "This is a new connector for ACR",
"acrConnectorId": "0fae7c97-02d0-4a97-9790-672f3a0e6897"
}
**Fetch ACR connector details**

/v1.3/registry/acr/connector/{connectorId}

[GET]

You can get the details for an ACR connector.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectorId</td>
<td>Provide the ACR connector Id to get the details for.</td>
</tr>
</tbody>
</table>

**API request:**

```
```

**Response:**

```
{
  "name": "New ACR connector 4",
  "description": "This is a new connector for ACR",
  "acrConnectorId": "66f9c9bf-9b04-4631-bb64-d0abfa44e6f6"
}
```

**Create a new ACR connector**

/v1.3/registry/acr/connector

[POST]

Use this API to create a new ACR connector.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicationId</td>
<td>Provide the Azure application ID.</td>
</tr>
<tr>
<td>clientSecret</td>
<td>Provide the Azure client secret.</td>
</tr>
<tr>
<td>name</td>
<td>Connector name.</td>
</tr>
<tr>
<td>description</td>
<td>(Optional) Connector description.</td>
</tr>
</tbody>
</table>

**API request:**

```
curl -X POST 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/acr/connector' -d '{"applicationId":"66f9c9bf-9b04-4631-bb64-d0abfa44e6f6",
"clientSecret":"Q]_RHUUGbgtCgQd8DWS8U6UJ18sTqK","description":"This is a new connector for ACR",
"name":"New ACR connector 4"}' --header
```
Registries
Fetch AWS account ID and external ID

'Authorization: Bearer <token>'

Response:

response code 200
{
  "name": "New ACR connector 4",
  "description": "This is a new connector for ACR",
  "acrConnectorId": "66f9c9bf-9b04-4631-bb64-d0abfa44e6f6"
}

Fetch AWS account ID and external ID
/v1.3/registry/aws-base
[GET]
Use this API to get your AWS account ID and external ID to help you create an ARN.

API request:


Response:

{
  "accountId": "20576771xxxx",
  "externalId": 27738xxxx
}

Fetch AWS US GovCloud account ID and external ID
/v1.3/registry/aws-gov-base
[GET]
Use this API to get your AWS US GovCloud account ID and external ID.

API request:


Response:

{
  "accountId": "12345678xxxx",
  "externalId": 12345xxxx
}
Fetch a list of AWS connectors in your account
/v1.3/registry/aws/connectors

[GET]

You can get a list of AWS connectors to help you create a registry. You’ll see accountType as Global, US_Gov or null for each connector in the API response.

API request:
```
curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/aws/connectors'
--header 'Authorization: Bearer <token>'
```

Response:
```
[

  {
    "name": "AWSC1",
    "arn": "arn:aws:iam::205767712438:role/abcd",
    "description": "AWS connector 1",
    "accountType": "Global"
  },

  {
    "name": "AWSC2",
    "arn": "arn:aws:iam::383031258652:role/testabcd",
    "description": "AWS connector 2",
    "accountType": "US_Gov"
  }
]
```

Fetch a list of AWS connectors for a certain account ID
/v1.3/registry/aws/connectors/[accountId]

[GET]

You can get a list of AWS connectors for an account ID to help you create a registry.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accountId</td>
<td>Provide the AWS account Id to get a list of connectors.</td>
</tr>
</tbody>
</table>

API request:
```
curl -X GET
--header 'Authorization: Bearer <token>'
```

Response:
```
[
]
Create a new AWS connector

/v1.3/registry/aws/connector

[POST]

Use this API to create a new AWS connector.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arn</td>
<td>ARN number of the account ID.</td>
</tr>
<tr>
<td>externalId</td>
<td>The externalId of your organization.</td>
</tr>
<tr>
<td>name</td>
<td>Connector name.</td>
</tr>
<tr>
<td>accountType</td>
<td>Specify the AWS account type as Global or US_Gov.</td>
</tr>
<tr>
<td>description</td>
<td>(Optional) Connector description.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X POST
```

Response:

response code 200

Fetch a list of GCR connectors in your account

/v1.3/registry/gcp/connectors

[GET]
You can get a list of GCR connectors to help you create a registry.

**API request:**
```bash
curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/gcp/connectors'
--header 'Authorization: Bearer <token>'
```

**Response:**
```json
{
   "name": "New GCR connector",
   "description": "This is a new connector for GCR",
   "gcpConnectorId": "10470225660035614490",
   "gcpProjectId": "my-second-project-239110"
},
{
   "name": "New GCR connector 2",
   "description": "This is a new connector for GCR",
   "gcpConnectorId": "107686475468845834251",
   "gcpProjectId": "my-second-project-239110"
}
```

**Fetch GCR connector details**

/v1.3/registry/gcp/connector/{connectorId}

[GET]

You can get the details for a GCR connector.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectorId</td>
<td>Provide the GCR connector Id to get the details for.</td>
</tr>
</tbody>
</table>

**API request:**
```bash
curl -X GET
--header 'Authorization: Bearer <token>'
```

**Response:**
```json
{
   "name": "New GCR connector",
   "description": "This is a new connector for GCR",
   "gcpConnectorId": "10470225660035614490",
   "gcpProjectId": "my-second-project-239110"
}
Create a new GCR connector

`/v1.3/registry/gcp/connector`

[POST]

Use this API to create a new GCR connector.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Connector name.</td>
</tr>
<tr>
<td>description</td>
<td>(Optional) Connector description.</td>
</tr>
<tr>
<td>Service Account JSON</td>
<td>Provide the configuration JSON file.</td>
</tr>
</tbody>
</table>

**API request:**

```bash
```

**Response:**

```
{
  "name": "New GCR connector",
  "description": "This is a new connector for GCR",
  "gcpConnectorId": "1047022566035614490",
  "gcpProjectId": "my-second-project-239110"
}
```

Validate registry parameters

`/v1.3/registry/validate`

[POST]

Use this API to validate parameters for a registry you intend to create. You can validate if a registry already exists, whether AWS/Azure/GCP account ID exists, if the credentials provided are correct, and so on.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR connectorId</td>
<td>Provide the ACR connector Id if your registry will be hosted on Azure. This parameter is required when the registryType is Azure and you want to create a new ACR connector.</td>
</tr>
</tbody>
</table>

AWS
accountId | Provide the AWS account Id if your registry will be hosted on AWS. Parameters accountId, arn, and region are required when the registryType is AWS ECR and you want to create a new AWS connector.

arn | ARN number of the account ID. Specify the ARN if you want to use an existing AWS connector, or if you want to create a new connector.

region | Region where your AWS account belong to.

accountType | Specify the AWS account type as Global or US_Gov.

**GCP**

connectorId | Provide the GCR connector Id if your registry will be hosted on GCP. Parameters connectorId, location, and projectId are required when the registryType is GCP and you want to create a new GCR connector.

location | Location where your GCP account belongs to.

projectId | Your GCP project ID, which is alphanumeric.

**GCAR**

connectorId | Provide the GCAR connector Id if your registry will be hosted on GCP. Parameters connectorId, projectId, and registry are required when the registryType is GOOGLE_ARTIFACT_REGISTRY and you want to create a new GCAR connector.

projectId | Your GCP project ID, which is alphanumeric (e.g. cs-registry).

registry | Name of the Google Artifact Registry repository (e.g. docker-v2-repo).

**Others**

username | Username to connect to the registry. Should be in base64 format.

password | Password to connect to the registry. Should be in base64 format.

credentialType | None, Token, BasicAuth, DockerHub, AWS, GCP. Note: This parameter is not required while creating Azure or GCP registries.

dockerHubOrgName | (Optional) Organization name if the registryType is DockerHub.

registryType | AWS ECR, Azure, GCP, DockerHub, Docker V2, Docker V2-Private, GOOGLE_ARTIFACT_REGISTRY.

registryUri | URL of the registry to connect to.

registryUuid | UUID of the registry you are validating. This is a mandatory parameter.
API request:
```
curl -X POST
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/validate' -d
'{"aws": {"accountId": "383031258652", "arn": "arn:aws:iam::383031258652:role/testabcd", "region": "us-east-2", "accountType": "Global"}, "credentialType": "AWS", "registryType": "AWS", "registryUri": "https://383031258652.dkr.ecr.us-east-2.amazonaws.com"}' --
header 'Authorization: Bearer <token>'
```

Response:
```
response code 200
```

Create registry

/v1.3/registry

[POST]

Use this API to create a new registry.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR</td>
<td></td>
</tr>
<tr>
<td>connectorId</td>
<td>Provide the ACR (Azure Container Registry) connector Id if your registry will be hosted on Azure. This parameter is required when the registryType is Azure and you want to create a new ACR connector.</td>
</tr>
<tr>
<td>AWS</td>
<td></td>
</tr>
<tr>
<td>accountId</td>
<td>Provide the AWS account Id if your registry will be hosted on AWS. Parameters accountId, arn, and region are required when the registryType is AWS ECR and you want to create a new AWS connector.</td>
</tr>
<tr>
<td>arn</td>
<td>ARN number of the account ID. Specify the ARN if you want to use an existing AWS connector, or if you want to create a new connector.</td>
</tr>
<tr>
<td>region</td>
<td>Region where your AWS account belongs. You must specify a US GovCloud region when accountType is US_Gov.</td>
</tr>
<tr>
<td>accountType</td>
<td>Specify the AWS account type as Global or US_Gov.</td>
</tr>
<tr>
<td>GCP</td>
<td></td>
</tr>
<tr>
<td>connectorId</td>
<td>Provide the GCR connector Id if you registry will be hosted on GCP. Parameters connectorId, location, and projectId are required when the registryType is GCP and you want to create a new GCR connector.</td>
</tr>
<tr>
<td>location</td>
<td>Location where your GCP account belong to.</td>
</tr>
</tbody>
</table>
**Registries**

**Create registry**

---

**projectId**

Your GCP project ID, which is alphanumeric.

---

**GCAR**

**connectorId**

Provide the GCAR connector Id if your registry will be hosted on GCP. Parameters connectorId, projectId, and registry are required when the registryType is GOOGLE_ARTIFACT_REGISTRY and you want to create a new GCAR connector.

---

**projectId**

Your GCP project ID, which is alphanumeric (e.g. cs-registry).

---

**registry**

Name of the Google Artifact Registry repository (e.g. docker-v2-repo).

---

**Others**

**username**

Username to connect to the registry. Should be in base64 format.

**password**

Password to connect to the registry. Should be in base64 format.

**credentialType**

None, Token, BasicAuth, DockerHub, AWS, GCP. Note: This parameter is not required while creating Azure or GCP registries.

**dockerHubOrgName**

(Optional) Organization name if the registryType is DockerHub.

**registryType**

AWS ECR, Azure, GCP, DockerHub, Docker V2, Docker V2-Private, GOOGLE_ARTIFACT_REGISTRY.

**registryUri**

URL of the registry to connect to.

**registryUuid**

This parameter should be empty during connector creation.

---

**API request (sample for AWS):**

```
curl -X POST 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry' -d '
"aws": {"accountId":"383031258652", "arn":"arn:aws:iam::383031258652:role/testabcd", "region":"us-east-2", "accountType":"Global"}, "credentialType":"AWS", "registryType":"AWS", "registryUri":https://383031258652.dkr.ecr.us-east-2.amazonaws.com}' --header 'Authorization: Bearer <token>'
```

**Response:**

```
{"registryUuid":"95b715e0-0fc7-4dac-b4de-2e1b92fc527d"}
```

**Response Code:** 200
API request (sample for GCAR):

curl -X POST 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry' -d '{"gcarRequest":{"connectorId":"1131360358474747331","projectId":"qualys-cs","registry":"cstest"},"credentialType":"GCP","registryType":"GOOGLE_ARTIFACT_REGISTRY","registryUri":"https://us-central1-docker.pkg.dev","registryUuid":null}' --header 'Authorization: Bearer <token>'

Response:

{"registryUuid":"b36965d6-c111-4964-a0ef-6d817454c3c1"}

Response Code: 200

**Update registry**

/v1.3/registry/{registryId}

[PUT]

Use this API to update an existing registry.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectorId</td>
<td>Provide the ACR (Azure Container Registry) connector Id if your registry is hosted on Azure. This parameter is required when the registryType is Azure and you want to update an ACR connector.</td>
</tr>
<tr>
<td>accountId</td>
<td>Provide the AWS account Id if your registry is hosted on AWS. Parameters accountId, arn, and region are required when the registryType is AWS ECR and you want to update an AWS connector.</td>
</tr>
<tr>
<td>arn</td>
<td>ARN number of the account ID. Specify the ARN if you want to use an existing AWS connector, or if you want to create a new connector.</td>
</tr>
<tr>
<td>region</td>
<td>Region where your AWS account belongs. You must specify a US GovCloud region when accountType is US_Gov.</td>
</tr>
<tr>
<td>accountType</td>
<td>Specify the AWS account type as Global or US_Gov.</td>
</tr>
<tr>
<td>connectorId</td>
<td>Provide the GCR connector Id if you registry is hosted on GCP. Parameters connectorId, location, and projectId are required when the registryType is GCP and you want to update a GCR connector.</td>
</tr>
<tr>
<td>location</td>
<td>Location where your GCP account belong to.</td>
</tr>
</tbody>
</table>
### Registries

#### Update registry

| API request: | curl -X PUT  
| Response: | Returns the same registry ID with response code 200. |

| **projectId** | Your GCP project ID, which is alphanumeric. |
| **connectorId** | Provide the GCAR connector Id if your registry will be hosted on GCP. Parameters connectorId, projectId, and registry are required when the registryType is GOOGLE_ARTIFACT_REGISTRY and you want to create a new GCAR connector. |
| **projectId** | Your GCP project ID, which is alphanumeric (e.g. cs-registry). |
| **registry** | Name of the Google Artifact Registry repository (e.g. docker-v2-repo). |

#### Others

| **username** | Username to connect to the registry. Should be in base64 format. |
| **password** | Password to connect to the registry. Should be in base64 format. |
| **credentialType** | None, Token, BasicAuth, DockerHub, AWS, GCP. Note: This parameter is not required while creating Azure or GCP registries. |
| **dockerHubOrgName** | (Optional) Organization name if the registryType is DockerHub. |
| **registryType** | AWS ECR, Azure, DockerHub, Docker V2, Docker V2-Private, GOOGLE_ARTIFACT_REGISTRY. |
| **registryUri** | URL of the registry to connect to. |
| **registryUuid** | UUID of the registry you are updating. This is a mandatory parameter. |
Fetch a list of repositories in a registry

/v1.3/registry/{registryId}/repository

[GET]

Here’s sample request and output to fetch a list of repositories in a registry.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) The ID of the registry for which you want to list the repositories.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the registries list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1. For API v1.1 and v1.2, only pageNo is supported. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example, repositoryName:asc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

**API request:**

```
```

**Response:**

```
{
    "data": [
        {
            "repoName": "qualys/cms/cms-processor-service",
            "totalImages": 4,
            "totalScannedImages": 4,
            "totalVulnerableImages": 2
        },
        {
            "repoName": "qualys/cms/cms-api-service",
            "totalImages": 4,
            "totalScannedImages": 4,
            "totalVulnerableImages": 1
        }
    ],
    "count": 2
}
```
Fetch a list of schedules created for a registry

/v1.3/registry/[registryId]/schedule

[GET]

Here’s sample request and output to fetch a list of schedules created for a registry.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) The ID of the registry for which you want to list the schedules.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the registries list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1. For API v1.1 and v1.2, only pageNo is supported. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example <code>created:desc</code>. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

```bash
```

Response:

```json
{
  "data": [
    {
      "scheduleUuid": "f98a64e7-4a62-40ae-b155-4a0cd445b42b",
      "onDemand": true,
      "created": "1537174851067",
      "updated": "1537174851067",
      "jobStartDate": "1537174851192",
      "jobCompletionDate": null,
      "name": null,
      "errors": null,
      "schedule": "00:00",
    }
  ]
}
```
"filters": [ 
  
  "repoTags": [ 
    
    "repo": ".*", 
    "tag": null 
  
  ],
  "days": null 
  
  ],
  "status": "Running" 

  },
  "scheduleUuid": "397b65f8-4b91-4698-8d7b-245b667981c3",
  "onDemand": true, 
  "created": "1536745686008",
  "updated": "1536745686008",
  "jobStartDate": "1536745686923",
  "jobCompletionDate": "1536745690933",
  "name": null,
  "errors": null,
  "schedule": "00:00",
  "filters": [ 
    
    "repoTags": [ 
      
      "repo": "qualys/cms/cms-api-service",
      "tag": "1.2.0.0" 

    ],
    "days": 7 

    ],
  "status": "Completed" 

  ],
  "count": 5,
  "groups": []}
Create registry schedule

/v1.3/registry/{registryId}/schedule

[POST]
Use this API to create a schedule to pull and scan a registry.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) The ID of the registry you want to scan.</td>
</tr>
<tr>
<td>OnDemand</td>
<td>Specify true if you want to enable OnDemand scan. Otherwise, Automatic scan will be triggered everyday at a set time.</td>
</tr>
<tr>
<td>repo</td>
<td>Provide the name of the repository you want to scan. You can add one more repositories to a single scan schedule.</td>
</tr>
<tr>
<td>tag</td>
<td>For OnDemand scan, include image tags you want to include in the scan (comma separated values).</td>
</tr>
<tr>
<td>days</td>
<td>For OnDemand scan, include images created a certain number of days ago. 1 to 7 days / 14 (for last two weeks).</td>
</tr>
<tr>
<td>schedule</td>
<td>For Automatic scan, specify time in UTC to scan at a set time every day. For example 19:30.</td>
</tr>
</tbody>
</table>

API request:

curl -X POST
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/995a2ab4-48dc-48ef-905d-9ecf846d63cb/schedule' -d '{"name":"scheduleTest", "onDemand":true, "filters": [{"repoTags": [{"repo":"qualys/cms/cms-api-service", "tag":"1.2.0.0"}], "days":7}], "schedule":null}' --header
'Authorization: Bearer <token>'

Response:

{"scheduleUuid":"397b65f8-4b91-4698-8d7b-245b667981c3"}
# Update registry schedule

/v1.3/registry/{registryId}/schedule/{scheduleId}

[PUT]

Use this API to update an existing registry schedule. Jobs in running state cannot be updated.

## Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) ID/UUID of the registry you want to update.</td>
</tr>
<tr>
<td>scheduleId</td>
<td>(Required) ID/UUID of the schedule you want to update.</td>
</tr>
<tr>
<td>OnDemand</td>
<td>Specify true if you want to enable OnDemand scan. Otherwise, Automatic scan will be triggered everyday at a set time. This parameter is not updatable, as you can update only Automatic scan jobs.</td>
</tr>
<tr>
<td>repo</td>
<td>Provide the name of the repository you want to scan. You can add one more repositories to a single scan schedule. This parameter is not updatable.</td>
</tr>
<tr>
<td>tag</td>
<td>For OnDemand scan, include image tags you want to include in the scan (comma separated values). This parameter is not updatable, as you can update only Automatic scan jobs.</td>
</tr>
<tr>
<td>days</td>
<td>For OnDemand scan, include images created certain days ago. 1 to 7 days, or 14 for last two weeks. This parameter is not updatable, as you can update only Automatic scan jobs.</td>
</tr>
<tr>
<td>schedule</td>
<td>For Automatic scan, specify time in UTC to scan at a set time every day. For example 19:30.</td>
</tr>
</tbody>
</table>

## API request:

```
curl -X PUT
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/995a2ab4-48dc-48ef-905d-9ecf846d63cb/schedule/5c4c7e21-9119-48f7-9458-57cededae000' -d
'"name":"scheduleTest", "onDemand":true, "filters": [{"repoTags": [{"repo":"cms-auth","tag":""}], "days":14}, {"schedule":"00:00"}]
```

## Response:

Returns the same schedule ID with response code 200.
Cancel registry schedule

/v1.3/registry/[registryId]/schedule/[scheduleId]/cancel

[POST]

Use this API to cancel a registry schedule.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) ID/UUID of the registry you want to cancel the schedule for.</td>
</tr>
<tr>
<td>scheduleId</td>
<td>(Required) ID/UUID of the schedule you want to cancel.</td>
</tr>
</tbody>
</table>

API request:


Response:

response code 200

Delete registries in your account

/v1.3/registry/

[DELETE]

Use this API to delete one or more registries in your account. Note that you cannot delete a registry whose schedules are in “Running” state.

You can choose to provide a request body without parameters or you can specify UUIDs as input parameters in the API request.

When specifying the request body:

Request Body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryDeleteRequest</td>
<td>(Required) Provide one or more registry UUIDs for the registries you want to delete.</td>
</tr>
</tbody>
</table>
Registries
Delete registries in your account

When specifying input parameters as part of the path:

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryIds</td>
<td>Provide one or more registry UUIDs for the registries you want to delete. When specifying multiple registries in the same request, enter them in this way: registryIds=value1&amp;registryIds=value2&amp;registryIds=value3, and so on.</td>
</tr>
</tbody>
</table>

**Sample 1 - Use request body to specify registries to delete**

In this sample, a request body is used as part of the API request.

**API request:**

```
curl -X DELETE 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry' -d '({"registryIds": ["95b715e0-0fc7-4dac-b4de-2e1b92fc527d"]})' --header 'Authorization: Bearer <token>'
```

**Response:**

```
Returns {"deletionJobId": "980ce235-5677-4997-81ca-3905e63471bb"}
response code 200
```

**Sample 2 - Delete a single registry using registry UUID**

In this sample, we'll delete a single registry by specifying the UUID as input parameter.

**API request:**

```
curl -X DELETE 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry?registryIds=8b261e4e-47f3-3b6a-a5a7-668ff0d6e3eb' --header 'Authorization: Bearer <token>'
```

**Response:**

```
{
  "deletionJobId": "8b04c56a-33ad-44ba-a128-fc643b0af3c7"
}
```

**Sample 3 - Delete multiple registries using registry UUIDs**

In this sample, we'll delete 2 registries in the same request. Specify multiple registries by entering registryIds=value1&registryIds=value2, and so on.

**API request:**

```
curl -X DELETE 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry?registryIds=8b261e4e-47f3-3b6a-a5a7-668ff0d6e3eb&registryIds=9b251e3e-52f3-2b6a-a6a7-678ff0d5e2eb' --header 'Authorization: Bearer <token>'
```

70
Delete a registry schedule

/v1.3/registry/{registryId}/schedule/{scheduleId}

[DELETE]

Use this API to delete an existing registry schedule.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) ID/UUID of the registry you want to delete the schedule for.</td>
</tr>
<tr>
<td>scheduleId</td>
<td>(Required) ID/UUID of the schedule you want to delete. Note: You cannot delete a schedule which is in &quot;Running&quot; state.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X DELETE 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/8eadf73e-357a-4282-9351-1ff453e4131d/schedule/aeab1ccd-a2ae-4bd9-807d-d387b0555fbe' --header 'Authorization: Bearer <token>'
```

Response:

response code 200

Delete multiple registry schedules (bulk delete)

/v1.3/registry/{registryId}/schedule

[DELETE]

Use this API to delete multiple registry schedules.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryId</td>
<td>(Required) ID/UUID of the registry you want to delete the schedules for.</td>
</tr>
<tr>
<td>scheduleIds</td>
<td>(Required) ID/UUIDs of the schedules you want to delete. Should be in the form of an array. Note: You cannot delete schedules that are in &quot;Running&quot; state.</td>
</tr>
</tbody>
</table>
API request:

curl -X DELETE
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/registry/fc129b85-e23c-4236-9fd2-47a257746208/schedule/' -d '["9a9468c9-33d7-49aa-8275-a1e92b30c916","8843983c-f0c6-441b-b8ed-4d00acf195b3"]' --header
'Authorization: Bearer <token>'

Response:

Returns {"deletedScheduleUids": ["9a9468c9-33d7-49aa-8275-a1e92b30c916","8843983c-f0c6-441b-b8ed-4d00acf195b3"]}

response code 200
Sensors

Here is the list of the APIs we currently support for sensors:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show a list of sensors in your account</td>
<td>GET</td>
<td>/csapi/v1.3/sensors</td>
</tr>
<tr>
<td>Show details of a sensor</td>
<td>GET</td>
<td>/csapi/v1.3/sensors/[sensorId]</td>
</tr>
<tr>
<td>Delete sensors in your account</td>
<td>DELETE</td>
<td>/csapi/v1.3/sensors</td>
</tr>
</tbody>
</table>

Samples for various operations on sensors:
- Fetch a list of sensors in your account
- Fetch sensor details
- Delete sensors in your account

Fetch a list of sensors in your account

/v1.3/sensors

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the sensors list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber or pageNo</td>
<td>(Optional) The page to be returned. Page numbers start with 1. If both are specified in the same request, then pageNumber takes precedence. For API v1.3, only pageNumber is supported.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example created:desc. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>
API request:

curl -X GET

Response:
{
  "data": [
    {
      "uuid": "d8bfff8e-4819-4828-b293-cc92e3e27647",
      "activationUuid": "038a9bfe-9f91-4bb0-a7e5-3f9615538642",
      "hostname": "qradar_vm",
      "customerUuid": "283f2b9a-1db4-e06a-814e-851eb53825eb",
      "dockerVersion": "17.09.1-ce",
      "ipv4": "10.113.198.178",
      "os": "Ubuntu 16.04.3 LTS",
      "ipv6": "",
      "sensorVersion": "1.2.5-22",
      "platform": "LINUX_SENSOR",
      "lastCheckedIn": "1558518764356",
      "label": [
        {
          "key": "name",
          "value": "Qualys Sensor Image"
        },
        {
          "key": "org.label-schema.name",
          "value": "CentOS Base Image"
        },
        {
          "key": "VersionInfo",
          "value": "Qualys Sensor Version 1.2.5-22"
        },
        {
          "key": "org.label-schema.license",
          "value": "GPLv2"
        },
        {
          "key": "org.label-schema.schema-version",
          "value": "1.0"
        },
        {
          "key": "vendor",
          "value": "Qualys, Inc"
        },
        {
          "key": "org.label-schema.build-date",
          "value": "20181006"
        },
        {
          "key": "org.label-schema.vendor",
          "value": "CentOS"
        }
      ]
    }
  ]
}
Sensors

Fetch a list of sensors in your account

},
{
  "key": "build-date",
  "value": "Wed Mar 20 10:35:21 UTC 2019"
},
{
  "key": "maintainer",
  "value": "devops <devops@qualys.com>"
}
],
"privileged": "false",
"macAddress": "00:50:56:9f:41:f8",
"vulnSigVersion": null,
"hostUuid": "2d5f7c15-a3e7-4bef-8183-2fd03f60aebe",
"configurationProfile": null,
"status": "Running",
"registry": "docker.io",
"sha": "9187496ccb3f96217ed2a4c3ce01d39aebaa442f9fa46a4fa131f8d054edd9a4",
"sensorId": "9187496ccb3f",
"name": "qualys-container-sensor",
"created": "1557467066000",
"imageId": "d6c910425801",
"imageSha": "d6c910425801a703530b92f943575b8ea9daa520f77f96e891993f1549a27073",
"binaryVersion": null,
"containerIpv4": null,
"containerIpv6": null,
"containerMacAddress": "",
"sensorType": "CICD"
},
...
],
"count": 4
}
Fetch sensor details
/v1.3/sensors/{sensorId}

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorId</td>
<td>Specify the ID of a specific sensor in the user’s scope.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/sensors/29d0a0bf6c6f' --
header 'Authorization: Bearer <token>'

Response:

```
{
  "uuid": "07ed65f6-474e-4d4c-9393-591b551f87ac",
  "activationUuid": "43f2d8e2-0551-48e0-a8e0-102c7d531922",
  "hostname": "cloudagent",
  "customerUuid": "77d66479-27e6-71bf-8143-35087e107fc9",
  "dockerVersion": "19.03.5",
  "ipv4": "10.11.12.13",
  "ipv6": "",
  "sensorVersion": "1.7.0-6",
  "platform": "LINUX_SENSOR",
  "lastCheckedIn": "161540575433",
  "label": [
    {
      "key": "build-date",
      "value": "Tue Jan  5 18:28:18 UTC 2021"
    },
    {
      "key": "image-source",
      "value": "SJC-USA"
    }
  ],
  "privileged": "false",
  "macAddress": "00:50:56:aa:fb:9a",
  "vulnSigVersion": null,
  "hostUuid": "1fcf95c1-f881-4584-a452-febe3f55809a",
  "configurationProfile": null,
  "status": "Unknown",
  "registry": "docker.io",
  "sha": "23d0a0bf6c6f9e513d15b17a4533e1feec976288f422b0f60238326941ebad65",
  "sensorId": "29d0a0bf6c6f",
  "name": "qualys-container-sensor",
  "created": "1615380389000",
  "imageId": "9e9ff9b51e30",
}
```
"imageSha": "9e9fe9b52e30f0997de62aabb796fc60e12e357d4d2c7e0c58298833c3f788d",
"binaryVersion": null,
"containerIpv4": null,
"containerIpv6": null,
"containerMacAddress": "",
"sensorType": "GENERAL",
"containerRuntime": "DOCKER",
"containerRuntimeVersion": "19.03.5",
"cluster": { 
  "type": "kubernetes",
  "version": "v1.1.1",
  "project": "k8s-project",
  "node": { 
    "name": "k8s-node",
    "isMaster": true
  },
  "pod": { 
    "name": "pod-name",
    "uuid": "6325d4f-bf0c-3488-8144-89c7d03dfacf",
    "namespace": "cs",
    "label": [{
        "key": "com.docker.compose.container-number",
        "value": "1"
      },
      {
        "key": "com.docker.compose.service",
        "value": "lb"
      }]
  }
} "controllers": [ 
  { 
    "uuid": "a3145d4f-bf0c-3488-8144-89c7d03dfacf",
    "name": "deployment-name",
    "type": "DEPLOYMENT"
  },
  {
    "uuid": "b3145d4f-bf0c-3488-8144-89c7d03dfacf",
    "name": "replicaset-name",
    "type": "REPLICASET"
  }
]
Delete sensors in your account

/v1.3/sensors

[DELETE]

You can only delete sensors with UNKNOWN status.

You can choose to provide a request body without parameters or you can specify UUIDs or filter as input parameters in the API request. See samples.

When specifying request body:

Request Body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorDeleteRequest</td>
<td>(Required) user filters to query sensors or provide one or more sensor UUIDs to delete. Filter can be applied by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

When specifying input parameters as part of path:

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorIds</td>
<td>One or more sensor UUIDs for the sensors you want to delete. When specifying multiple sensors in the same request, enter them in this way: sensorIds=value1&amp;sensorIds=value2&amp;sensorIds=value3, and so on.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter the sensors list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Sample 1 - Use request body to specify sensors to delete

In this sample, a request body is used as part of the API request.

API request:

```
```

Response:

```
Returns ("deletionJobId":"bbaac4c7-6263-4e2f-b391-bcb032975206")
```

response code 200
Sample 2 - Delete single sensor using sensor UUID
In this sample, we will delete a single sensor by specifying the sensor UUID.

**API request:**
```
curl -X DELETE
```

**Response:**
```
{
  "deletionJobId": "8b04c56a-33ad-44ba-a128-fc643b0af3c7"
}
```

Sample 3 - Delete multiple sensors using sensor UUIDs
In this sample, we’ll delete 2 sensors in the same request. Specify multiple sensors by entering sensorIds=value1&sensorIds=value2, and so on.

**API request:**
```
curl -X DELETE
```

**Response:**
```
{
  "deletionJobId": "6e04c56a-42ad-44ba-a678-fc642b0af3c9"
}
```

Sample 4 - Delete sensors using a filter
In this sample, we will delete sensors based on the filter parameter.

**API request:**
```
curl -X DELETE
```

**Response:**
```
{
  "deletionJobId": "7a93e37e-b2c6-40ac-8f18-70f87fcf3957"
}
```
Activation Key

Use this API to get the keys required to activate a new sensor. The API response includes your customerId, activationId, and platformUrl.

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get activation key</td>
<td>GET</td>
<td>/csapi/v1.3/activationkey</td>
</tr>
</tbody>
</table>

Get activation key
/csapi/v1.3/activationkey

[GET]

API request:

curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/activationkey' --header
'Authorization: Bearer <token>'

Response:

{
  "customerId": "192cc974-1e44-cb6c-806e-f78f6441cb0d",
  "activationId": "192cc974-1e44-cb6c-806e-f78f6441cb0d",
  "platformUrl": "https://qualysguard.qualys.com/"
}
Vulnerability Reports

Create and manage Image and Container Vulnerability Reports using the following API endpoints.

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a report request</td>
<td>POST</td>
<td>/csapi/v1.3/reports</td>
</tr>
<tr>
<td>Fetch a list of reports</td>
<td>GET</td>
<td>/csapi/v1.3/reports</td>
</tr>
<tr>
<td>Download a report</td>
<td>GET</td>
<td>/csapi/v1.3/reports/{reportUuid}/download</td>
</tr>
<tr>
<td>Delete reports</td>
<td>DELETE</td>
<td>/csapi/v1.3/reports</td>
</tr>
</tbody>
</table>

Jump to a section to see API samples:

- Create a report request
- Fetch a list of reports
- Download a report
- Delete reports

Create a report request

/csapi/v1.3/reports

[POST]

Use this API to create a new report request.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Required) Specify a title for your report (up to 150 characters).</td>
</tr>
<tr>
<td>description</td>
<td>Specify a description for your report (up to 250 characters).</td>
</tr>
<tr>
<td>templateName</td>
<td>(Required) Specify the template for the report you want to create. Valid values are: CS_IMAGE_VULNERABILITY or CS_CONTAINER_VULNERABILITY</td>
</tr>
</tbody>
</table>
**Vulnerability Reports**

Create a report request

---

Filter

Filter the images or containers list for the report by providing a query using Qualys syntax. Refer to the "How to Search" topic in the online help for assistance with creating your query.

**displayColumns**

Specify the columns to include in the report. Multiple columns should be comma-separated. When unspecified, ALL report columns will be included. When an empty value is provided, only default columns will be included.

When the template CS_IMAGE_VULNERABILITY is used, you can include any of these columns: repo, imageld, sha, uuid, created, updated, qid, title, severity, cveids, vendorReference, cvssBase, cvssTemporal, cvss3Base, cvss3Temporal, threat, impact, solution, exploitability, associatedMalwares, category, software, result. Default columns are: qid, imageld.

When the template CS_CONTAINER_VULNERABILITY is used, you can include any of these columns: name, containerId, uuid, imageld, created, hostName, hostIp, state, stateChanged, updated, qid, title, severity, cveids, vendorReference, cvssBase, cvssTemporal, cvss3Base, cvss3Temporal, threat, impact, solution, exploitability, associatedMalwares, category, software, result. Default columns are: qid, containerId.

---

**API request:**

```bash
curl -X POST 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/reports' -d '
{"description":"Demo Report","name":"My Container Report","templateName":"CS_CONTAINER_VULNERABILITY","filter":"status:running","displayColumns":{"containerid","uuid","qid"} }
' --header 'Authorization: Bearer <token>
```

**Response:**

```json
response code 200
{
  "reportUuId": "037570f0-0193-11ea-9327-8fbbd2104c9c"
}
```
Fetch a list of reports

/csapi/v1.3/reports

[GET]

Use this API to fetch a list of reports from your account.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the reports list by providing a query using Qualys syntax. Currently, you can only filter by reportName. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageSize</td>
<td>The number of records per page to be included in the response. The default value is 50.</td>
</tr>
<tr>
<td>pageNumber</td>
<td>The page to be returned. Page numbers start with 1. The default value is 1.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. You can sort by status or reportName only. Default value is status:desc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET

Response:

```json
{
  "data": [
    {
      "reportUuid": "4738d060-5b02-11eb-a819-9b3f1886eb36",
      "createdAt": "2021-01-20T09:31:27.000Z",
      "reportName": "sample-image-report",
      "description": null,
      "fileFormat": "csv",
      "templateName": "CS_IMAGE_VULNERABILITY",
      "status": "FAILED",
      "reportType": "ON_DEMAND",
      "filter": null,
      "displayColumns": ["ALL"]
    },
    {
      "reportUuid": "42699db0-9eb1-11eb-bbb6-833197dc0c5b",
      "createdAt": "2021-04-16T12:42:52.000Z",
      "reportName": "sample-container-report",
      "description": null,
      "fileFormat": "csv",
    }
  ]
}```
Download a report

/csapi/v1.3/reports/{reportUuid}/download

[GET]
Use this API to download a specific report.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportUuid</td>
<td>(Required) The UUID for the report you want to download.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/reports/903db2f0-e450-11e9-9cdc-8362fe1d0c3a/download' --header 'Authorization: Bearer <token>'

Response:

response code 200
Delete reports

/csapi/v1.3/reports

[DELETE]

Use this API to delete one or more reports from your account.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportUuuids</td>
<td>(Required) One or more report UUIDs for the reports you want to delete. When specifying multiple reports to delete, enter them in this way: reportUuuids=value1&amp;reportUuuids=value2&amp;reportUuuids=value3, and so on.</td>
</tr>
</tbody>
</table>

API request:

```
```

Response:

```
response code 200
{
  "data": null,
  "message": "Report Successfully deleted."
}
```
Compliance

Here is the list of the APIs we currently support for policy compliance posture:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show PC compliance posture for an image</td>
<td>GET</td>
<td>/csapi/v1.3/images/{imageSha}/compliance</td>
</tr>
<tr>
<td>Fetch PC compliance posture for a container</td>
<td>GET</td>
<td>/csapi/v1.3/containers/{containerSha}/compliance</td>
</tr>
<tr>
<td>Fetch details for a control</td>
<td>GET</td>
<td>/csapi/v1.3/controls/{controlId}</td>
</tr>
<tr>
<td>Fetch a list of controls</td>
<td>GET</td>
<td>/csapi/v1.3/controls</td>
</tr>
</tbody>
</table>

Samples for various operations on compliance:

- Fetch compliance posture for an image
- Fetch compliance posture for a container
- Fetch control details
- Fetch a list of controls

**Fetch compliance posture for an image**

/v1.3/images/{imageSha}/compliance

[GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageSha</td>
<td>Specify the SHA value of a specific image in the user's scope.</td>
</tr>
</tbody>
</table>

**API request:**

```shell
curl -X GET 'https://gateway.qg2.apps.qualys.com/csapi/v1.3/images/c64844065dcabc3d0a90c365c1f56421766a5cebf05f7ecbd3377af410fff09fd/compliance' --header 'Authorization: Bearer <token>'
```

**Response:**

```json
{
   "uuid": "5d48f83b-cddb-33ac-8fad-e8452dd116b1",
   "sha": "c64844065dcabc3d0a90c365c1f56421766a5cebf05f7ecbd3377af410fff09fd",
   "customerUuid": "192cc974-1e44-cb6c-806e-f78f6441cb0d",
   "created": "1603477517000",
   "updated": "1605017537578",
   "controls": [
```
Fetch compliance posture for a container

/v1.3/containers/{containerSha}/compliance

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerSha</td>
<td>Specify the SHA value of a specific container in the user’s scope.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X GET "https://gateway.qg2.apps.qualys.com/csapi/v1.3/containers/b87b645dffda05e59bb80ac20678b1c1f051c4a1286bafe8c55a58e523d49af5/compliance' --header 'Authorization: Bearer <token>''
```

Response:

```
{
   "uuid": "fd011da7-9fed-314f-9884-76713eb66156",
   "sha": "b87b645dffda05e59bb80ac20678b1c1f051c4a1286bafe8c55a58e523d49af5",
   "customerUuid": "192cc974-1e44-cb6c-806e-f78f6441cb0d",
   "created": "1604869118000",
   "updated": "1604922275091",
   "controls": [
      {
         "controlId": 10808,
         "policyUuid": "e18b623d-3f07-485b-a754-5a1c31727df3",
         "technologyId": 0,
         "criticality": "CRITICAL",
      }
   ]
}
```
"posture": "SETTING_NOT_FOUND",
"lastEvaluated": "1604869185266",
"datapoints": [ 
  { 
    "key": "dockersensor00.container.capdrop",
    "value": "161803399999999"
  }
],
},
{
  "controlId": 10716,
  "policyUuid": "e18b623d-3f07-485b-a754-5a1c31727df3",
  "technologyId": 0,
  "criticality": "SERIOUS",
  "posture": "SETTING_NOT_FOUND",
  "lastEvaluated": "1604869185266",
  "datapoints": [ 
    { 
      "key": "dockersensor00.container.pidmode",
      "value": "161803399999999"
    }
  ],
},
"lastComplianceScanned": "1604869185266"

Fetch control details
/v1.3/controls/{controlId}

[GET]
Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlId</td>
<td>Specify the ID of a compliance control for which you want to get details.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET
'https://gateway.qg2.apps.qualys.com/csapi/v1.3/controls/10808' --header
'Authorization: Bearer <token>'

Response:

{ 
  "id": 10808,
  "statement": "Status of the 'cap-drop' flag settings on Docker
containers on the host system",
  "criticality": "CRITICAL",
}
"comments": "STMT: Status of the 'cap-drop' flag settings on Docker containers on the host system.\n\nRAT: Linux Capabilities allows dividing privileges associated with superuser into distinct group of smaller units, known as capabilities. By default, Docker containers are started with a restricted set of capabilities where each one can be independently enabled and disabled. This enables the processes running inside a container to perform almost all the specific areas where root privileges are usually needed without having to have them run as root. Unrestricted Linux capabilities could allow unauthorized access to containers which could potentially lead to attacks such as privilege escalation exploits. Linux Capabilities on Docker containers should be restricted as appropriate to the needs of the business to have only those that are required for the containers to perform their function.\n\nCIS_Docker_1.11.0_Benchmark_v1.0.0: 5.3 Restrict Linux Kernel Capabilities within containers\nCIS_Docker_1.12.0_Benchmark_v1.0.0: 5.3 Restrict Linux Kernel Capabilities within containers",
"deprecated": "Control is not deprecated",
"category": "Access Control Requirements",
"subCategory": "Authorization (Single-user ACL/role)",
"technologies": [
{
  "technologyId": 283,
  "technologyName": "Docker Containers/Images",
  "rational": "Linux Capabilities allows dividing privileges associated with superuser into distinct group of smaller units, known as capabilities. By default, Docker containers are started with a restricted set of capabilities where each one can be independently enabled and disabled. This enables the processes running inside a container to perform almost all the specific areas where root privileges are usually needed without having to have them run as root. Unrestricted Linux capabilities could allow unauthorized access to containers which could potentially lead to attacks such as privilege escalation exploits. Linux Capabilities on Docker containers should be restricted as appropriate to the needs of the business to have only those that are required for the containers to perform their function."
},

"remediation": "Run the following command to verify that the added and dropped Linux Kernel Capabilities are in line with the ones needed for container process for each container instance.\n\ndocker ps --quiet | xargs docker inspect --format '{}: CapAdd=} CapDrop=}' |\n\nRun the following command to add needed capabilities:\n\ndocker run --cap-add={"Capability 1"},{"Capability 2"} <run-arguments> <container-image-name-or-ID>\n\nFor example,\n\ndocker run --interactive --tty --cap-add={"NET_ADMIN","SYS_ADMIN"} centos:latest /bin/bash\n\nTo drop unneeded capabilities, run the following command:\n\ndocker run --cap-drop={"Capability 1"},{"Capability 2"} <run-arguments> <container-image-name-or-ID> <command>\n\nFor example,\n\ndocker run --interactive --cap-drop={"SETUID","SETGID"} centos:latest /bin/bash\n\nAlternatively, drop all capabilities and add only add only those that are the needed:\n\ndocker run --cap-drop=all --cap-add={"Capability 1"},{"Capability 2"} <run-arguments> <container-image-name-or-ID> <command>\n\nFor example,\n\ndocker run --interactive --tty --cap-drop=all --cap-add={"NET_ADMIN","SYS_ADMIN"} centos:latest /bin/bash"
Fetch a list of controls

/v1.3/controls

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the controls list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNumber</td>
<td>(Required) The page to be returned. The default is 1.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response. The default value is 50.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example, <code>created:desc</code>. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

```
```

Response:

```
{
  "data": [
    {
      "id": 18990,
      "statement": "Status of the host devices and their exposure",
      "criticality": "SERIOUS",
      "comments": "STMT: Status of the host devices and their exposure\nRAT: Host devices can be directly exposed to containers at runtime. Do not directly expose host devices to containers, especially to containers that are not trusted. The --device option exposes host devices to containers and as a result of this, containers can directly access these devices. The container would not need to run in privileged mode to access and manipulate them, as by default, the container is granted this type of access. Additionally, it would possible for containers to remove block devices from the host. You therefore should not expose host devices to containers directly. If for some reason you wish to expose the host device to a container you should consider which sharing permissions you wish to use on a case by case base as appropriate to your organization.\n\nOld mappings before granular NIST was added: New"
```

---

New

Old mappings before granular NIST was added: New
Compliance
Fetch a list of controls

control

Granular mapping:
AC-1, ACCESS CONTROL POLICY AND PROCEDURES,
Delete Old Mapping:N/A
Granular Mapping: Yes
Granular Mapping Review: No",
"deprecated": "Control is not deprecated",
"category": "OS Security Settings",
"subCategory": "System Settings (OSI layers 6-7)",
"technologies": [

"id": 181,
"name": "Docker CE/EE",
"rational": "Host devices can be directly exposed to containers at runtime. Do not directly expose host devices to containers, especially to containers that are not trusted. The --device option exposes host devices to containers and as a result of this, containers can directly access these devices. The container would not need to run in privileged mode to access and manipulate them, as by default, the container is granted this type of access. Additionally, it would possible for containers to remove block devices from the host. You therefore should not expose host devices to containers directly. If for some reason you wish to expose the host device to a container you should consider which sharing permissions you wish to use on a case by case base as appropriate to your organization."
,
"remediation": "You should not directly expose host devices to containers. If you do need to expose host devices to containers, you should use granular permissions as appropriate to your organization:
For example, do not start a container using the command below:
docker run --interactive --tty --device=/dev/tty0:/dev/tty0:rwm --device=/dev/temp_sda:/dev/temp_sda:rwm centos bash
You should only share the host device using appropriate permissions:
docker run --interactive --tty --device=/dev/tty0:/dev/tty0:rw --device=/dev/temp_sda:/dev/temp_sda:r centos bash"
},

"id": 283,
"name": "Docker Containers/Images",
"rational": "Host devices can be directly exposed to containers at runtime. Do not directly expose host devices to containers, especially to containers that are not trusted. The --device option exposes host devices to containers and as a result of this, containers can directly access these devices. The container would not need to run in privileged mode to access and manipulate them, as by default, the container is granted this type of access. Additionally, it would possible for containers to remove block devices from the host. You therefore should not expose host devices to containers directly. If for some reason you wish to expose the host device to a container you should consider which sharing permissions you wish to use on a case by case base as appropriate to your organization."
,
"remediation": "You should not directly expose host devices to containers. If you do need to expose host devices to containers, you should use granular permissions as appropriate to your organization:
For example, do not start a container using the command below:
docker run --interactive --tty --device=/dev/tty0:/dev/tty0:rwm --device=/dev/temp_sda:/dev/temp_sda:rwm centos bash
You should only share the host device using appropriate permissions:

Fetch a list of controls

- tty --device=/dev/tty0:/dev/tty0:rw --
device=/dev/temp_sda:/dev/temp_sda:r centos bash


Compliance

Status of the 'cap-drop' flag settings on Docker containers on the host system

criticality: "CRITICAL",

comments: "STMT: Status of the 'cap-drop' flag settings on Docker containers on the host system

RAT: Linux Capabilities allows dividing privileges associated with superuser into distinct group of smaller units, known as capabilities. By default, Docker containers are started with a restricted set of capabilities where each one can be independently enabled and disabled. This enables the processes running inside a container to perform almost all the specific areas where root privileges are usually needed without having to have them run as root. Unrestricted Linux capabilities could allow unauthorized access to containers which could potentially lead to attacks such as privilege escalation exploits. Linux Capabilities on Docker containers should be restricted as appropriate to the needs of the business to have only those that are required for the containers to perform their function.

CIS_Docker_1.11.0_Benchmark_v1.0.0: 5.3 Restrict Linux Kernel Capabilities within containers
CIS_Docker_1.12.0_Benchmark_v1.0.0: 5.3 Restrict Linux Kernel Capabilities within containers",

deprecated: "Control is not deprecated",

category: "Access Control Requirements",

subCategory: "Authorization (Single-user ACL/role)"

remediation: "Run the following command to verify that the added and dropped Linux Kernel Capabilities are in line with the ones needed for container process for each container instance.\n\n# docker ps --quiet | xargs docker inspect --format '}: CapAdd=} CapDrop=}'\n\nRun the following command to add needed capabilities:\n\n# docker run --cap-add={"Capability 1"},"Capability 2"} <run-arguments> <container-image-name-or-ID> <command>\n\nFor example,\n\n# docker run --interactive --tty --cap-add={"NET_ADMIN"","SYS_ADMIN"} centos:latest /bin/bash\n\nTo drop
unneeded capabilities, run the following command:

```
$ docker run --cap-drop="\"Capability 1\"",\"Capability 2\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-drop="\"SETUID\",\"SETGID\"" centos:latest /bin/bash
```

Alternatively, drop all capabilities and add only those that are the needed:

```
$ docker run --cap-drop=all --cap-add="\"Capability 1\"",\"Capability 2\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-drop=all --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" centos:latest /bin/bash
```

Alternatively, drop all capabilities and add only those that are the needed:

```
$ docker run --cap-drop=all --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-drop=all --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" centos:latest /bin/bash
```

### 283. Docker Containers/Images

**Rationale:** Linux Capabilities allows dividing privileges associated with superuser into distinct group of smaller units, known as capabilities. By default, Docker containers are started with a restricted set of capabilities where each one can be independently enabled and disabled. This enables the processes running inside a container to perform almost all the specific areas where root privileges are usually needed without having to have them run as root. Unrestricted Linux capabilities could allow unauthorized access to containers which could potentially lead to attacks such as privilege escalation exploits. Linux Capabilities on Docker containers should be restricted as appropriate to the needs of the business to have only those that are required for the containers to perform their function.

**Remediation:** Run the following command to verify that the added and dropped Linux Kernel Capabilities are in line with the ones needed for container instance:

```
$ docker ps --quiet | xargs docker inspect --format '}: CapAdd=} CapDrop=}'
```

Run the following command to add needed capabilities:

```
$ docker run --cap-add="\"Capability 1\"",\"Capability 2\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" centos:latest /bin/bash
```

To drop unneeded capabilities, run the following command:

```
$ docker run --cap-drop="\"Capability 1\"",\"Capability 2\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-drop="\"SETUID\",\"SETGID\"" centos:latest /bin/bash
```

Alternatively, drop all capabilities and add only those that are the needed:

```
$ docker run --cap-drop=all --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" <run-arguments> <container-image-name-or-ID> <command>
```

For example:

```
$ docker run --interactive --tty --cap-drop=all --cap-add="\"NET_ADMIN\",\"SYS_ADMIN\"" centos:latest /bin/bash
```
needed without having to have them run as root. Unrestricted Linux capabilities could allow unauthorized access to containers which could potentially lead to attacks such as privilege escalation exploits. Linux Capabilities on Docker containers should be restricted as appropriate to the needs of the business to have only those that are required for the containers to perform their function.

"remediation": "Run the following command to verify that the added and dropped Linux Kernel Capabilities are in line with the ones needed for container process for each container instance.

```bash
# docker ps --quiet | xargs docker inspect --format '}: CapAdd={} CapDrop={}'}
```

Run the following command to add needed capabilities:

```bash
# docker run --cap-add={"Capability 1","Capability 2"} <run-arguments> <container-image-name-or-ID> <command>
```

For example,

```bash
# docker run --interactive --tty --cap-add={"NET_ADMIN","SYS_ADMIN"} centos:latest /bin/bash
```

To drop unneeded capabilities, run the following command:

```bash
# docker run --cap-drop={"Capability 1","Capability 2"} <run-arguments> <container-image-name-or-ID> <command>
```

For example,

```bash
# docker run --interactive --tty --cap-drop={"SETUID","SETGID"} centos:latest /bin/bash
```

Alternatively, drop all capabilities and add only the needed:

```bash
# docker run --cap-drop=all --cap-add={"Capability 1","Capability 2"} <run-arguments> <container-image-name-or-ID> <command>
```

For example,

```bash
# docker run --interactive --tty --cap-drop=all --cap-add={"NET_ADMIN","SYS_ADMIN"} centos:latest /bin/bash
```