Container Security
API Guide
Version 1.6

January 27, 2020
# Table of Contents

**About this Guide** .............................................................................................................. 5

- About Qualys .................................................................................................................... 5
- Qualys Support .................................................................................................................. 5
- About Container Security Documentation ......................................................................... 5

**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

**Containers** ......................................................................................................................... 9

- Fetch a list of containers in your account ........................................................................ 9
- Fetch container details ....................................................................................................... 11
- Fetch a list of software installed on a container ............................................................... 13
- Fetch vulnerability details for a container ........................................................................ 15
- Fetch vulnerability count for a container .......................................................................... 18
- Delete containers in your account .................................................................................. 19

**Images** ................................................................................................................................. 20

- Fetch a list of images in your account ............................................................................ 20
- Fetch image details ........................................................................................................... 23
- Fetch associations for an image ....................................................................................... 26
- Fetch a list of software installed on an image ................................................................. 27
- Fetch vulnerability details for an image ........................................................................... 29
- Fetch vulnerability count for an image ............................................................................ 31
- Delete images in your account ....................................................................................... 32

**Registries** ............................................................................................................................. 33

- Fetch a list of registries in your account ......................................................................... 34
- Fetch registry details ......................................................................................................... 35
- Fetch AWS account ID and external ID .......................................................................... 37
- Fetch a list of AWS connectors in your account ............................................................. 37
- Fetch a list of AWS connectors for a certain account ID ............................................... 38
- Create connector ............................................................................................................. 38
- Validate registry parameters ........................................................................................... 39
- Create registry .................................................................................................................. 41
- Update registry .................................................................................................................. 42
- Fetch a list of repositories in a registry ............................................................................ 43
- Fetch a list of schedules created for a registry ................................................................. 44
Create registry schedule....................................................................................................... 46
Update registry schedule...................................................................................................... 47
Cancel registry schedule....................................................................................................... 48
Delete a registry in your account ....................................................................................... 49
Delete multiple registries (bulk delete) in your account ................................................... 49
Delete a registry schedule .................................................................................................... 50
Delete multiple registry schedules (bulk delete) ............................................................... 51

**Sensors** ............................................................................................................. 52
Fetch a list of sensors in your account ............................................................................... 52
Fetch sensor details........................................................................................................... 55
Delete sensors in your account............................................................................................ 57
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 Vulnerability Analysis Plugin for Jenkins
- Qualys Vulnerability Analysis Plugin for Bamboo
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.
assrownedHostsCount 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,

```
```

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.
Accessing the APIs
Do I need to Authenticate to use Swagger?

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.

Simply, click Authorize and provide the user name and password. You can now try out the APIs!
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/v1.1/containers</td>
</tr>
<tr>
<td>Show details of a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}</td>
</tr>
<tr>
<td>Show software installed on a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/vuln/count</td>
</tr>
<tr>
<td>Delete containers in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/containers</td>
</tr>
</tbody>
</table>

Samples for various operations on containers:

- Fetch a list of containers in your account
- Fetch container details
- 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

/v1.1/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>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1.</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>
Containers
Fetch a list of containers in your account

API request:
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' 'https://<QualysURL>/csapi/v1.1/containers?pageNo=1&pageSize=50&sort=created%3Adesc'

Response:

{  
  "data": [  
    {  
      "imageId": "4ab4c602aa5e",
      "created": "1545223164000",
      "sha": "d1f7cccee5f36b3944056851a00403fe6e6e18e1aa9b727640712b124f20de0791",
      "uuid": "969e28b3-aa9d-3377-89bb-c475a4f92c8e",
      "name": "competent_fermat",
      "host": {  
        "sensorUuid": "054ae100-b243-4a28-a9db-eb9b0b605ff8",
        "hostname": "docker2",
        "ipAddress": "10.115.74.187",
        "uuid": null  
      },  
      "state": "STOPPED",
      "imageUuid": "f1b0c11f-ceb1-32a9-9186-04b06842d360",
      "containerId": "d1f7cccee5f3",
      "stateChanged": "1545223413425",
      "lastScanned": null,
      "vulnerabilities": {  
        "severity5Count": 4,
        "severity3Count": 1,
        "severity4Count": 0,
        "severity1Count": 0,
        "severity2Count": 2  
      }  
    },  
    {  
      "imageId": "16508e5c265d",
      "created": "1540449432000",
      "sha": "81031b661db4b9da51df4d5731b9675704c0f4875002510e7318a5b8c1131b34",
      "uuid": "46a14d1f-1f94-30c5-aaf4-5088c22a0f59",
      "name": "dazzling_leavitt",
      "host": {  
        "sensorUuid": "74a2e7b2-9cd9-4433-b6db-e0d8dd160a5",
      }  
    }  
  ]
}
Containers
Fetch container details

```
"hostname": "docker1",
"ipAddress": "10.115.74.188",
"uuid": "69c6aa4a-10a3-434b-92a9-7cff4e0470d4"
},
"state": "CREATED",
"imageUuid": "6fc320a1-be65-30f4-bc6b-f940f87515c0",
"containerId": "81031b661db4",
"stateChanged": "1540449432366",
"lastScanned": null,
"vulnerabilities": {
   "severity5Count": 0,
   "severity3Count": 1,
   "severity4Count": 2,
   "severity1Count": 0,
   "severity2Count": 1
}
```

... 

```
"portMapping": null,
"imageId": "5182e96772bf",
"created": "1543906019000",
"label": [
```

Fetch container details

/v1.1/containers/{containerId}

[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 --header 'Accept: application/json' --header 'Authorization: Basic dXN1cm5hbWU6cGFzc3dvcmQ==' 'https://<QualysURL>/csapi/v1.1/containers/d52b37423ce5'
```

Response:

```
{
   "portMapping": null,
   "imageId": "5182e96772bf",
   "created": "1543906019000",
   "label": [
Containers
Fetch container details

{
   "key": "org.label-schema.name",
   "value": "CentOS Base Image"
},
{
   "key": "org.label-schema.license",
   "value": "GPLv2"
},
{
   "key": "org.label-schema.schema-version",
   "value": "1.0"
},
{
   "key": "org.label-schema.vendor",
   "value": "CentOS"
},
{
   "key": "org.label-schema.build-date",
   "value": "20180804"
}
],
"uuid": "dfe980eb-6963-343e-8f1d-dd399322caf6",
"sha": "d52b37423ce5ab70f0e47e8689fbb71ae7a4411ba7c982ad899a780b8205ce2e",
"privileged": false,
"sensorUuid": "6bb7b4d8-f3b7-4f37-888c-08d226e7d20",
"path": "/bin/bash",
"imageSha": "5182e96772bf11f4b912658e265dfe0db8bd314475443b6434ea708784192892",
"macAddress": "",
"customerUuid": "92a3a277-60c2-4a41-8053-a6d480ccf8dc",
"ipv4": null,
"ipv6": null,
"name": "festive_wozniak",
"host": {
   "sensorUuid": "6bb7b4d8-f3b7-4f37-888c-08d226e7d20",
   "hostname": "docker1",
   "ipAddress": "10.115.74.188",
   "uuid": "186a40e0-b06e-44d4-ac16-e719a0e04f96"
},
"state": "STOPPED",
"imageUuid": "10837ba6-7717-3be8-b288-f98b364c78ec",
"containerId": "d52b37423ce5",
"stateChanged": "1543908325849"
Containers

Fetch a list of software installed on a container

```
"hostname": null,
"services": null,
"users": null,
"operatingSystem": null,
"lastScanned": null,
"environment": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
",
"arguments": null,
"command": "/bin/bash",
"drift": null,
"vulnerabilities": null,
"softwares": null,
"isDrift": false
]
```

Fetch a list of software installed on a container

/v1.1/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 “How to Search” 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>created:desc</code>. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
<tr>
<td>isDrift</td>
<td>Specify true if you are looking for drift containers. Default is false.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK=='
'https://qualysguard.qualys.com/csapi/v1.1/containers/cfe5171a75d3/software?isDrift=false'
```
Containers

Fetch a list of software installed on a container

Response:

{
  "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
      }
    },
    {
      "name": "libncursesw5:amd64",
      "version": "6.0+20160213-1ubuntu1",
      "fixVersion": null,
      "vulnerabilities": {
        "severity5Count": null,
        "severity4Count": null,
        "severity3Count": null,
        "severity2Count": null,
        "severity1Count": null
      }
    }
  ],
  "count": 134,
  "softwareCountBySeverity": {
    "severity5Count": 0,
    "severity3Count": 7,
    "severity4Count": 0,
    "severity1Count": 0,
    "severity2Count": 0
  }
}
Containers

Fetch vulnerability details for a container

/v1.1/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:

```
```

Response:

```
{
  "details": {
    "vulns": [
      {
        "vulnerability": null,
        "result": "libudev1 229-4ubuntu21.16 229-4ubuntu21.21

        \nlibsystemd 229-4ubuntu21.16 229-4ubuntu21.21

        \nnsystemd 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
}
Containers

Fetch vulnerability details for a container

```json
{
    "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
        }
    }
}
```
Containers
Fetch vulnerability count for a container

/v1.1/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 --header 'Accept: application/json' --header 'Authorization: Basic dXN1cm5hbWU6cGFzc3dvcmQK=='
'https://qualysguard.qualys.com/csapi/v1.1/containers/cfe5171a75d3/vuln/count'
```

Response:

```json
{
  "severity5Count": 0,
  "severity3Count": 3,
  "severity4Count": 0,
  "severity1Count": 0,
  "severity2Count": 2
}
```
Delete containers in your account

/v1.1/containers

[DELETE]

Input Parameters:

<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 “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
   "containerIds": [
       "a6025a31-bd86-37e6-9de7-5722af586b66", "c4032e71-5969-34a9-a8a9-ba2b69729673"
   ]
}
```

API request:

```
```

Response:

```
Returns ("deletionJobId":"413b076e-01a8-4780-8e62-875b615a9a1f")
```

response code 200
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/v1.1/images</td>
</tr>
<tr>
<td>Show details of an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}</td>
</tr>
<tr>
<td>Show associations for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/association</td>
</tr>
<tr>
<td>Show software installed on an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/vuln/count</td>
</tr>
<tr>
<td>Delete images in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/images</td>
</tr>
</tbody>
</table>

Samples for various operations on images:

- Fetch a list of images in your account
- Fetch image details
- Fetch associations for an 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**

/v1.1/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 &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1.</td>
</tr>
</tbody>
</table>
PageSize | (Required) The number of records per page to be included in the response.
--- | ---
Sort | Sort the results using a Qualys token. For example `eventOccurred:desc`. Refer to the "Sortable tokens" topic in the online help for more information.

**API request:**
```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' 'https://<QualysURL>/csapi/v1.1/images?filter=vulnerabilities.severity%3A%225%22&pageNumber=1&pageSize=50&sort=created%3Adesc'
```

**Response:**
```
{
  "data": [
    {
      "created": "1526592592000",
      "sha": "93c55587b0a54b626b6bedcf34a77387da69a21612ce823b69a11066f9dce90a",
      "repo": [
        {
          "registry": "docker.io",
          "tag": "baddoc",
          "repository": "qualysdemo/checkoutapp"
        },
        {
          "registry": "registry-1.docker.io",
          "tag": "baddoc",
          "repository": "qualysdemo/checkoutapp"
        }
      ],
      "uuid": "3fe02a98-36cf-317b-8e76-89a012c37235",
      "size": 1082856183,
      "vulnerabilities": {
        "severity5Count": 5,
        "severity3Count": 13,
        "severity4Count": 39,
        "severity1Count": 1,
        "severity2Count": 4
      },
      "imageId": "93c55587b0a5",
      "associatedContainersCount": 2,
      "associatedHostsCount": 2
    }
  ]
}
```
"lastScanned": "1546552299809",
"registryUuid": [
  "266a697b-6a4c-46d8-b36e-f1a910ca79c9",
  "428b40aa-25ab-41da-b89a-62e8e3fcb58f"
],
"source": [ 
  "REGISTRY"
]
},
{
  "created": "1509562706000",
  "sha": "abd4f451ddf707c8e68a36d695456a515cdd6f9581b7a8348a380030a6fd7689",
  "repo": [
    {
      "registry": "docker.io",
      "tag": "latest",
      "repository": "imiell/bad-dockerfile"
    }
  ],
  "uuid": "236f66f7-dc80-3e58-9a16-88ab987fc20f",
  "size": 1082855961,
  "vulnerabilities": {
    "severity5Count": 5,
    "severity3Count": 13,
    "severity4Count": 29,
    "severity1Count": 1,
    "severity2Count": 3
  },
  "imageId": "abd4f451dcb7",
  "associatedContainersCount": 1,
  "associatedHostsCount": 2,
  "lastScanned": "1540407394660",
  "registryUuid": null,
  "source": [ 
    "GENERAL"
  ]
}
],
"count": 2}
Fetch image details

/v1.1/images/[imageId]

[GET]

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageUrl</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 --header 'Accept: application/json' --header 'Authorization: Basic dXNlc3NhbWU6cGFzc3dvcmQK==' 'https://qualysapi.qualys.com/csapi/v1.1/images/5d556c82899c'

Response:

```json
{
    "created": "1557406536000",
    "author": "Name <email.id@here>",
    "repo": [
        {
            "registry": "docker.io",
            "tag": "cicd",
            "repository": "ubuntu"
        }
    ],
    "label": null,
    "layersCount": 8,
    "uuid": "6ac22614-cf15-31a1-ba05-ef485aaa8c29",
    "sha": "5d556c82899c2ebbb8f5ba985dcecd5ba847fba35a1e0ecbb20d242c020f2314b",
    "operatingSystem": "Ubuntu Linux 18.04.2",
    "sensorUuid": [2a9726f2-69d1-4255-b6ba-2f4d0c7bb596],
    "customerUuid": "283f2b9a-1db4-e06a-814e-851eb53825eb",
    "dockerVersion": "17.06.0-ce",
    "size": 101749224,
    "layers": [
        {
            "size": "101749224",
            "createdBy": "ADD file:7ce84f13f1609a50ece7823578159412e2299c812746d1fled5db0728b"
        }
    ]
}
```
d37 in /

```
{
    "created": "1556317262000",
    "comment": "",
    "id": null,
    "sha": null,
    "tags": null
},
...
{
    "size": "0",
    "createdBy": "ENTRYPOINT ["/usr/bin/redis-server"]",
    "created": "1557406536000",
    "comment": "",
    "id": "5d556c82899c",
    "sha": "5d556c82899cebb8f5ba985dceccd5ba847fba35a1e0eccb20d242c020f2314b",
    "tags": [
        "ubuntu:cicd",
        "qualys_scan_target:5d556c82899c"
    ]
}
}

"host": [  
    {
        "sensorUuid": "2a9726f2-69d1-4255-b6ba-2f4d0c7bb596",
        "hostname": "qualys-virtual-machine",
        "ipAddress": "10.115.67.98",
        "uuid": null
    }
],
"architecture": "amd64",
"imageId": "5d556c82899c",
"lastScanned": "1557406965966",
"registryUuid": null,
"source": [  
    "CICD"
],
"totalVulCount": "2",
"users": [  
    "root"
],
"softwares": [  
    {
        "name": "libgcrypt20:amd64",
        "version": "1.8.1-4ubuntu1.1",
```
"fixVersion": null,
"vulnerabilities": null
},
{
"name": "libncurses5:amd64",
"version": "6.1-1ubuntu1.18.04",
"fixVersion": null,
"vulnerabilities": null
},
...

"vulnerabilities": [
{
"vulnerability": null,
"result": "Exact OS obtained: Ubuntu Linux 18.04.2\n",
"lastFound": "1557406965965",
"firstFound": "1557406965964",
"fixed": null,
"severity": 2,
"customerSeverity": 2,
"port": null,
"typeDetected": "CONFIRMED",
"status": null,
"nonRunningKernel": null,
"nonExploitableConfig": null,
"runningService": null,
"risk": 20,
"category": "General remote services",
"os": null,
"discoveryType": [
   "REMOTE"
],
"authType": [],
"supportedBy": [
   "VM"
],
"product": [],
"vendor": [],
"cveids": [],
"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": 38510,
"title": "CA Agent Discloses Exact Operating System Version",
"cvssInfo": {
  "baseScore": "5.0",
  "temporalScore": "4.8",
  "accessVector": "Network"
},
"cvss3Info": {
  "baseScore": "5.3",
  "temporalScore": "4.4"
},
"patchAvailable": false,
"published": null,
"ageInDays": null,
"software": null
},
...
}

## Fetch associations for an image

/v1.1/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>
Fetch a list of software installed on an image

/v1.1/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 <strong>qid:asc</strong>. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>
API request:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK=='
'https://qualysapi.qualys.com/csapi/v1.1/images/5d556c82899c/software'

Response:

{
  "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
      }
    },
    ...
  ],
  "count": 89,
  "softwareCountBySeverity": {
    "severity5Count": 0,
    "severity3Count": 0,
    "severity4Count": 0,
    "severity1Count": 0,
    "severity2Count": 0
  },
  "driftSoftwareCountBySeverity": null,
  "driftSoftwareCountByDriftReason": null
}
Fetch vulnerability details for an image

/v1.1/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 <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:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' 
'https://qualysapi.qualys.com/csapi/v1.1/images/5d556c82899c/vuln?type=AL&sort=qid%3Aasc'

Response:

```json
{
  "details": [
    {
      "vulnerability": null,
      "result": "sysctl 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,
Images

Fetch vulnerability count for an image

/v1.1/images/{imageId}/vuln/count

[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>
</tbody>
</table>

API request:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK=='
'https://qualysapi.qualys.com/csapi/v1.1/images/5d556c82899c/vuln/count'

Response:

```json
{
  "severity5Count": 0,
  "severity3Count": 0,
  "severity4Count": 0,
  "severity1Count": 0,
  "severity2Count": 2
}
```
Delete images in your account

/v1.1/images

[DELETE]

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

Input Parameters:

<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 “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
   "imageIds": [
      "e3e4cca0-8305-3835-810a-b334dcb65a33"
   ]
}
```

API request:

```bash
```

Response:

Returns {"deletionJobId":"980ce235-5677-4997-81ca-3905e63471bb"}

response code 200
## Registries

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/v1.1/registry</td>
</tr>
<tr>
<td>Show details of a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/[registryId]</td>
</tr>
<tr>
<td>Fetch AWS account ID and External ID for your account</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws-base</td>
</tr>
<tr>
<td>Show a list of AWS connectors in your account</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws/connectors</td>
</tr>
<tr>
<td>Show a list of AWS connectors for an AWS account ID</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws/connectors/[accountId]</td>
</tr>
<tr>
<td>Create new AWS connector</td>
<td>POST</td>
<td>/csapi/v1.1/registry/aws/connector</td>
</tr>
<tr>
<td>Validate information for new registry</td>
<td>POST</td>
<td>/csapi/v1.1/registry/validate</td>
</tr>
<tr>
<td>Create a new registry</td>
<td>POST</td>
<td>/csapi/v1.1/registry</td>
</tr>
<tr>
<td>Update existing registry in your account</td>
<td>PUT</td>
<td>/csapi/v1.1/registry/[registryId]</td>
</tr>
<tr>
<td>Show a list of repositories in a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/[registryId]/repository</td>
</tr>
<tr>
<td>Show a list of schedules created for a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/[registryId]/schedule</td>
</tr>
<tr>
<td>Create a new registry scan schedule</td>
<td>POST</td>
<td>/csapi/v1.1/registry/[registryId]/schedule</td>
</tr>
<tr>
<td>Update existing registry schedule in your account</td>
<td>PUT</td>
<td>/csapi/v1.1/registry/[registryId]/schedule/ {scheduleId}</td>
</tr>
<tr>
<td>Cancel registry schedule in your account</td>
<td>POST</td>
<td>/csapi/v1.1/registry/[registryId]/schedule/ {scheduleId}/cancel</td>
</tr>
<tr>
<td>Delete registry in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/[registryId]</td>
</tr>
<tr>
<td>Delete multiple registries in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry</td>
</tr>
<tr>
<td>Delete registry schedule in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/[registryId]/schedule/ {scheduleId}</td>
</tr>
<tr>
<td>Delete multiple registry schedules in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/[registryId]/schedule/ {scheduleId}</td>
</tr>
</tbody>
</table>

Samples for various operations on registries:

- Fetch a list of registries in your account
- Fetch registry details
- Fetch AWS account ID and external ID
Fetch a list of registries in your account
Fetch a list of AWS connectors for a certain account ID
Create 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 a registry in your account
Delete multiple registries (bulk delete) in your account
Delete a registry schedule
Delete multiple registry schedules (bulk delete)

Fetch a list of registries in your account
/v1.1/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 &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 0.</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:**

Response:
{
  "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,
      "awsAccountId": null,
      "awsRegion": null
    },
    {
      "registryUuid": "57739abc-ee35-43ab-9f74-2157c15a0ae4",
      "registryUri": "https://registry-1.docker.io",
      "registryType": "DockerHub",
      "repoCount": 0,
      "totalImages": 0,
      "totalScannedImages": 0,
      "totalVulnerableImages": 0,
      "lastScanned": "1536134457859",
      "scheduleStatusList": {
        "Completed": 3
      },
      "created": "1536237658094",
      "updated": "1536237658094",
      "dockerHubOrg": null,
      "providerType": null,
      "awsAccountId": null,
      "awsRegion": null
    },
    ...
  ],
  "count": 6,
  "groups": {}
}

Fetch registry details
/v1.1/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 --header 'Accept: application/json' --header 'Authorization: Basic dXNlc3NhbWU6cGFzc3dvcmQK' 'https://qualysapi.qualys.com/csapi/v1.1/registry/ada5f044-f177-43c7-a306-353697c6d5a0'
```

**Response:**

```
{
    "registryUuid": "ada5f044-f177-43c7-a306-353697c6d5a0",
    "registryUri": "https://art-hq.intranet.qualys.com:5001",
    "registryType": "V2",
    "repoCount": 252,
    "totalImages": 966,
    "totalScannedImages": 0,
    "totalVulnerableImages": 0,
    "scheduleStatusList": {
        "Completed": 2,
        "Running": 5
    },
    "created": "1537253984965",
    "updated": "1537253984965",
    "lastScanned": "1537720005089",
    "dockerHubOrg": null,
    "providerType": null,
    "aws": null,
    "credential": {
        "username": "anonymous"
    },
    "connectors": null
}
```
Fetch AWS account ID and external ID
/v1.1/registry/aws-base
[GET]
You can get your AWS account ID and external ID to help you create an ARN.

API request:
```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK' 
'https://qualysapi.qualys.com/csapi/v1.1/registry/aws-base'
```

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

Fetch a list of AWS connectors in your account
/v1.1/registry/aws/connectors
[GET]
You can get a list of AWS connectors to help you create a registry.

API request:
```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' 
'https://qualysapi.qualys.com/csapi/v1.1/registry/aws/connectors'
```

Response:
```
[
  {
    "arn": "arn:aws:iam::205767712438:role/abcd",
    "name": "AWSC1",
    "description": "AWS connector 1"
  },
  {
    "arn": "arn:aws:iam::383031258652:role/testabcd",
    "name": "AWSC2",
    "description": "AWS connector 2"
  }
]
```
Fetch a list of AWS connectors for a certain account ID

/v1.1/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>(Required) Provide the AWS account Id to get a list of connectors.</td>
</tr>
</tbody>
</table>

**API request:**

```
```

**Response:**

```
[
  {
    "arn": "arn:aws:iam::205767712438:role/abcd",
    "name": "AWSC1",
    "description": "AWS connector 1"
  },
  {
    "arn": "arn:aws:iam::205767712438:role/testabcd",
    "name": "AWSC2",
    "description": "AWS connector 2"
  }
]
```

Create connector

/v1.1/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>name</td>
<td>Connector name.</td>
</tr>
<tr>
<td>description</td>
<td>Connector description.</td>
</tr>
<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>
</tbody>
</table>
Input parameters can be provided in following format if you are using swagger:

```json
{
  "arn": "arn:aws:iam::205767712438:role/abcd",
  "externalId": "903805594",
  "name": "TestAWS",
  "description": "Testing of AWS account"
}
```

**API request:**

```bash
```

**Response:**

response code 200

---

**Validate registry parameters**

/v1.1/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 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>accountld</td>
<td>Provide the AWS account Id if your registry will be hosted on AWS. Parameters accountld, 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 belong to.</td>
</tr>
<tr>
<td>username</td>
<td>Username to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>password</td>
<td>Password to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>credentialType</td>
<td>None, Token, BasicAuth, DockerHub, AWS.</td>
</tr>
</tbody>
</table>
dockerHubOrgName  (Optional) Organization name if the registryType is DockerHub.

registryType  AWS ECR, DockerHub, Docker V2, Docker V2-Private.

registryUri  URL of the registry to connect to.

Input parameters can be provided in following format if you are using swagger:

```json
{
    "aws": {
        "accountId": "383031258652",
        "arn": "arn:aws:iam::383031258652:role/testabcd",
        "region": "us-east-2"
    },
    "credentialType": "AWS",
    "registryType": "AWS",
    "registryUri": "https://383031258652.dkr.ecr.us-east-2.amazonaws.com"
}
```

**API request:**

```bash
```

**Response:**

```json
{
    "errorCode": "CMS-1504",
    "message": "Registry arn:aws:iam::383031258652:role/testabcd already exists in system.",
    "timestamp": 1536744107711
}
```
Create registry

/v1.1/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>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 belong to.</td>
</tr>
<tr>
<td>username</td>
<td>Username to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>password</td>
<td>Password to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>credentialType</td>
<td>None, Token, BasicAuth, DockerHub, AWS.</td>
</tr>
<tr>
<td>dockerHubOrgName</td>
<td>(Optional) Organization name if the registryType is DockerHub.</td>
</tr>
<tr>
<td>registryType</td>
<td>AWS ECR, DockerHub, Docker V2, Docker V2-Private.</td>
</tr>
<tr>
<td>registryUri</td>
<td>URL of the registry to connect to.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
    "aws": {
        "accountId": "383031258652",
        "arn": "arn:aws:iam::383031258652:role/testabcd",
        "region": "us-east-2"
    },
    "credentialType": "AWS",
    "registryType": "AWS",
    "registryUri": "https://383031258652.dkr.ecr.us-east-2.amazonaws.com"
}
```

**API request:**

```bash
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' -d '{
    "aws": {
        "accountId": "383031258652",
        "arn": "arn:aws:iam::383031258652:role/testabcd",
        "region": "us-east-2"
    },
    "credentialType": "AWS",
    "registryType": "AWS",
    "registryUri": "https://383031258652.dkr.ecr.us-east-2.amazonaws.com"
}'
```
Update registry

/v1.1/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>registryId</td>
<td>(Required) ID/UUID of the registry you want to update.</td>
</tr>
<tr>
<td>accountId</td>
<td>Provide a new AWS account Id for this registry.</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>username</td>
<td>Username to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>password</td>
<td>Password to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>credentialType</td>
<td>None, Token, BasicAuth, DockerHub, AWS.</td>
</tr>
<tr>
<td>dockerHubOrgName</td>
<td>(Optional) Organization name if the registryType is DockerHub.</td>
</tr>
<tr>
<td>registryType</td>
<td>AWS ECR, DockerHub, Docker V2, Docker V2-Private. This field is not updatable, but should be provided as input during registry update.</td>
</tr>
<tr>
<td>registryUri</td>
<td>URL of the registry to connect to. This field is not updatable, but should be provided as input during registry update.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{"credential": {
    "username": "dXNlcm5hbWU=",
    "password": "cGFzc3dvcmQ="
},
"credentialType": "BasicAuth",
"registryType": "V2",
"registryUri": "https://383039876789.dkr.ecr.us-east-1.amazonaws.com"
}
```
API request:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' -d '{"credential": { "username": "dXNlcm5hbWU=", "password": "cGFzc3dvcmQ=" }, "credentialType": "BasicAuth", "registryType": "V2", "registryUri": "https://383039876789.dkr.ecr.us-east-1.amazonaws.com" }' 'https://qualysapi.qualys.com/csapi/v1.1/registry/b994c2e6-8961-4133-a889-662d8cf52310'
```

Response:

Returns the same registry ID with response code 200.

### Fetch a list of repositories in a registry

```
/v1.1/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 &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 0.</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>repositoryName:asc</code>. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

```
```

Response:

```
{
  "data": [
    
  
```
Fetch a list of schedules created for a registry

/v1.1/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>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 0.</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 <strong>created:desc</strong>. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

API request:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK=='

Response:

```json
{
  "data": [
```
Registries

Fetch a list of schedules created for a registry

```json
[
  {
    "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
/v1.1/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 certain 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>

Input parameters can be provided in following format if you are using swagger:

```
{
    "name": "scheduleTest",
    "onDemand": true,
    "filters": [
        {
            "repoTags": [{
                "repo": "qualys/cms/cms-api-service",
                "tag": "1.2.0.0"
            }],
            "days": 7
        }
    ],
    "schedule": null
}
```

API request:
```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' -d '{ "name": "scheduleTest", "onDemand": true, "filters": [ { "repoTags": [{ "repo": "qualys/cms/cms-api-service", "tag": "1.2.0.0" }], "days": 7 } ], "schedule": null }' 'https://qualysapi.qualys.com/csapi/v1.1/registry/995a2ab4-48dc-48ef-905d-9ecf846d63cb/schedule'
```
Response:

```json
{"scheduleUuid":"397b65f8-4b91-4698-8d7b-245b667981c3"}
```

**Update registry schedule**

```
/v1.1/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>

Input parameters can be provided in following format if you are using swagger:

```json
{
   "name": "scheduleTest",
   "onDemand": true,
   "filters": [
   {
      "repoTags": [{
         "repo": "cms-auth",
         "tag": "",
      }],
      "days": 14
   }
}
```
Registries
Cancel registry schedule

API request:
```bash
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ==' -d '{ "name": "scheduleTest", "onDemand": true, "filters": [ { "repoTags": [ { "repo": "cms-auth", "tag": "" } ], "days": 14 }, { "schedule": "00:00" } ]}' 'https://qualysapi.qualys.com/csapi/v1.1/registry/995a2ab4-48dc-48ef-905d-9ecf846d63cb/schedule/5c4c7e21-9119-48f7-9458-57cededae000'
```

Response:
Returns the same schedule ID with response code 200.

Cancel registry schedule

/v1.1/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:
```bash
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ==' 'https://qualysapi.qualys.com/csapi/v1.1/registry/4d319710-1e1c-48c3-a19f-548bb8e438a5/schedule/93efaf06-364d-401f-b4fd-e0c9810bd8e6/cancel'
```

Response:
response code 200
Delete a registry in your account

/v1.1/registry/{registryId}

[DELETE]

Use this API to delete an existing 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 delete. Note: You cannot delete a registry whose schedules are in &quot;Running&quot; state.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X DELETE --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK==' 'https://qualysapi.qualys.com/csapi/v1.1/registry/95b715e0-0fc7-4dac-b4de-2e1b92fc527d'
```

Response:

response code 200

Delete multiple registries (bulk delete) in your account

/v1.1/registry

[DELETE]

Use this API to delete multiple registries in your account.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryIds</td>
<td>(Required) ID/UUIDs of the registries you want to delete. Should be in the form of an array. Note: You cannot delete registries whose schedules are in &quot;Running&quot; state.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```
["fc129b85-e23c-4236-9fd2-47a257746208", "fe066970-0efc-4b04-91f4-b21870c61136"]
```

API request:

```
curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK' -d '["fc129b85-e23c-4236-9fd2-47a257746208", "fe066970-0efc-4b04-91f4-b21870c61136"]' 'https://qualysapi.qualys.com/csapi/v1.1/registry'
```
Delete a registry schedule

/v1.1/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 “Running” state.</td>
</tr>
</tbody>
</table>

API request:

curl -X DELETE --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK' 'https://qualysapi.qualys.com/csapi/v1.1/registry/8eadf73e-357a-4282-9351-1ff453e4131d/schedule/aeab1ccd-a2ae-4bd9-807d-d387b0555fbe'

Response:

response code 200
Delete multiple registry schedules (bulk delete)

/v1.1/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>

Input parameters can be provided in following format if you are using swagger:

```
["9a9468c9-33d7-49aa-8275-a1e92b30c916","8843983c-f0c6-441b-b8ed-4d00acf195b3"]
```

API request:

```
curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK' -d '']=='9a9468c9-33d7-49aa-8275-a1e92b30c916','8843983c-f0c6-441b-b8ed-4d00acf195b3"' 'https://qualysapi.qualys.com/csapi/v1.1/registry/fc129b85-e23c-4236-9fd2-47a257746208/schedule/
```

Response:

```
Returns {"deletedScheduleUuids":["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/v1.1/sensors</td>
</tr>
<tr>
<td>Show details of a sensor</td>
<td>GET</td>
<td>/csapi/v1.1/sensors/{sensorId}</td>
</tr>
<tr>
<td>Delete sensors in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/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

/v1.1/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 “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 1.</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:

Sensors

Fetch a list of sensors in your account

Response:

```json
{
  "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"
},
{
"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": "9187496ccb3f96217ed2a4c3ce01d39abeaa442f9fa46a4fa131f8d054edd9a4",
"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

/v1.1/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:


Response:

```json
{
  "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"
    }
  ]
}```
Sensors

Fetch sensor details

```json
{
  "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"
},
{
  "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": "9187496ccb3f96217ed2a4c3ce01d39abeaa442f9fa46a4fa131f8d054edd9a4",
"sensorId": "9187496ccb3f",
"name": "qualys-container-sensor",
"created": "1557467066000",
"imageId": "d6c910425801",
"imageSha": "d6c910425801a703530b92f943575b8ea9daa520f77f96e891993f1549a27073",
"binaryVersion": null,
"containerIpv4": null,
"containerIpv6": null,
"containerMacAddress": ""
```
"sensorType": "CICD"}

Delete sensors in your account

/v1.1/sensors

[DELETE]

You can only delete sensors with UNKNOWN status.

Input Parameters:

<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 &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
  "filter": "hostname:cms"
}
```

API request:

```
curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK=' -d '{
  "filter": "hostname:cms"
}'
'https://qualysapi.qualys.com/csapi/v1.1/sensors'
```

Response:

Returns {"deletionJobId":"bbaac4c7-6263-4e2f-b391-bcb032975206"}

response code 200