Qualys Jira Connector

User Guide

February 23, 2023

Qualys, Inc.
919 E Hillsdale Blvd
4th Floor
Foster City, CA 94404
1 (650) 801 6100

Table of Contents

About this guide

Welcome to Qualys Cloud Platform! We'll show you how to use the Jira Connector Application to manage tracking host and the vulnerabilities detected in Qualys platform

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 a 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 support information at www.qualys.com/support/

Welcome to Qualys Jira Connector

The Qualys Jira Connector is an application that offers customers a convenient solution to manage host and the vulnerabilities detected in Qualys platform. We are translating the host detections into meaningful action items on Jira and along with that, it has the added benefit of creating reports on Jira dashboards, irrespective of whether Jira instance is hosted on Jira On-Cloud or On-Premise.

Let's look at how you can get started with Jira Connector.

Pre-requisites

Software Requirements

- You should have installed Docker compose.

Qualys Requirements

- A valid Qualys subscription with API access enabled for host detection and knowledgebase APIs

Roles and Permissions for Available APIs

Refer below to find the API URLs for Knowledgebase and Host Detection with their required roles and permissions

Knowledgebase

/api/2.0/fo/knowledge_base/vuln/?action=list

Role	Permissions
Manager, Unit Manager, Scanner, Reader	Download vulnerability data from the KnowledgeBase.
Auditor	No permission to download vulnerability data from the KnowledgeBase.

Host Detections

/api/2.0/fo/asset/host/vm/detection/

Role	Permissions
Managers	View all VM scanned hosts in subscription
Unit Managers	View VM scanned hosts in the user's assigned business unit.
Scanners and Readers	View VM scanned hosts in the user's account.
Auditors	Have no permission to view VM scanned hosts.

Note:

- This API is available to Express Lite users.
- API only returns information for hosts that are assigned to each user through asset groups in VM/VMDR.

Refer to Qualys API (VM, PC) User Guide for more.

Jira Requirements

- A valid Jira user with admin privileges and Jira API token created using same admin user.

- The Jira project where tickets are to be created must be of type 'Company Managed'.

- It is recommended to not set any of the fields in the Jira project as mandatory to avoid ticket creation failure.

- We support both Jira On-Cloud and On-Premise. The above requirements are sufficient for Jira On-Cloud. Whereas, On-Premise requires additional pre-requisites that can be referred to in For Jira Instance On-Premise.

-Ensure you have minimum supported Jira On-Premise version of v8.22.0#822000-sha1:cef2cc4 or above.

Before you begin with the rest of the configurations, lets understand the types of tickets/issues the Jira connector creates in your instance.

Understanding Ticketing Schemes

This section helps you understand the available ticketing schemes to give you clarity on the configurations required in the following steps.

If you have a Jira On-Premise setup, based on the selection of your ticketing scheme, you can fulfill the additional pre-requisites specific to Jira On-Premise.

The type of ticketing schemes you select will help you with managing the tickets in your Jira.

The types of tickets that can be created for host assets and detections are saved as templates in the ticketing scheme template JSON file. Refer below to understand what the ticketing schemes are and how you can leverage them.

The Jira connector supports two ticketing schemes currently:

Ticketing Scheme 1: Host_Vuln_Linking_Ticket_Scheme

A parent ticket (Vulnerable Host ticket) is created for every host detected and synced by the host detection API. A child ticket (Vulnerability ticket) is then created for each unique combination of QID and Port.

The Vulnerability tickets are:

- Linked to the Vulnerable Host ticket if respective detection status are in New/Active/Reopen.

- Unlinked from the Vulnerable Host ticket if they are re-synced in a Fixed status.

If all the Vulnerability tickets under a Vulnerable Host ticket are unlinked and there no other linked tickets, then the Vulnerable Host tickets are closed.

Ticketing scheme 2: Per_Detection_Separate_Ticket_Scheme

A single ticket (Host Vulnerability) is created for every unique combination of detected Host, QID and Port. Only if the detection status are in New/Active/Reopen.

Note:

- For either ticketing schemes, if the detection is synced for the first time in Fixed status, then a ticket is not created.
- The Jira connector only creates tickets with the fields described in the ticketing template files. Adding or updating fields in the template JSON leads to failue while creating tickets.
- However, 'workflowMappings' section can be updated as per the workflow mapping in your Jira instance.

For Jira Instance On-Premise

Now that you are aware of the available ticketing schemes, let's look at the additional pre-requisites for Jira On-Premise.

Note: If you have Jira On-Cloud setup, you can skip this section and move to Install the App.

Create Custom Issue Types

First, let's create issue types based on your ticketing scheme selection.

1. Open your Jira Instance in browser.
2. Navigate to **Project Settings** > **Issue types** > **Add issue type**.
3. If your ticketing scheme selection is '1' (Host_Vuln_Linking_Ticket_Scheme), the issue types will be:

   a) Name: Vulnerable Host, Type: Standard Issue Type

   b) Name: Vulnerability, Type: Standard Issue Type
4. Or, for ticketing scheme selection '2' (Per_Detection_Separate_Ticket_Scheme), the issue type will be

   a) Name: Host Vulnerability, Type: Standard Issue Type

Adding Issue Types to Issue Type Scheme

1. Navigate to **Project Settings** > **Issue types** > **Schemes**.

2. Select the project where tickets are to be created and click **Edit**.
3. On '**Modify issue types scheme**', move newly created issue types from the '**Available issue type**' table to the '**Issue types for current scheme**' table.
4. Save.

Creating Custom Fields

Now that you have created issue types and added them to an issue type scheme, let us create custom fields needed for the tickets.

Custom Fields for Ticketing Schemes has the list of custom fields required for ticketing. Ensure you refer to this list while creating the custom fields by following the steps described below.

1.Navigate to Project Settings > Fields > Custom fields.

2.Click Add Custom fields.

3.Configure the custom field name and click Next.

4.Make the following selections,

a.Select issue types: Issue type as per ticketing scheme selection and ticket type.

b.Choose context: Apply to the issues in selected projects.

c.Select Projects: Project where tickets are to be created.

5.Next, select the create, edit/view and resolve issue screen of the respective project for newly created custom field type.

6.Click Update.

Now that you have all of the pre-requisites in place. Let's begin with the installation of the application.

Get Started

Follow the steps to install the Jira Connector application.

Install the App

- Go to the GitHub repository of the Jira connector application

- Copy the docker-compose.yml file to your local directory on the machine where you intend to setup the Jira connector application.

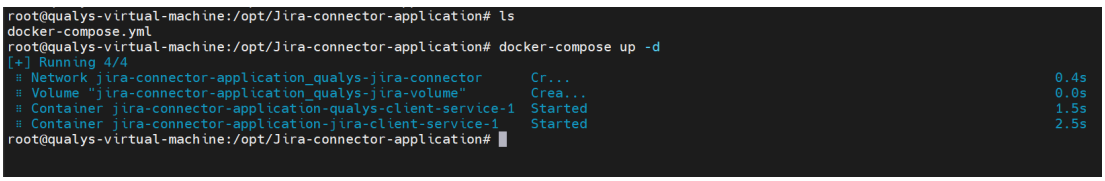
- Execute the following commands from your local directory:

a) If you are using the docker compose standalone installation, *docker-compose up*

b) If you are using the docker compose plugin, *docker compose up*

c) Spin up the container in detached mode using the parameter, *-d*

Refer below image to understand how the output will look like.



- To verify installation, execute the command, *docker ps*

Note: As a part of the behavior of Docker compose, your local directory name with the docker-compose.yml will be prefixed on to the container, volume, and network name.

Now, that you have installed the application, let's see what the Jira connector application has introduced in your environment.

Post-Installation

- After installation, two different containers are spun up on your machine.

a) A container named qualys-client-service-1

b) Another one named jira-client-service-1

- The Jira connector will mount a docker volume named 'qualys-jira-volume' in your environment. You can check the volume by executing the command, *docker volume ls*

- The Docker volume by default is located at '/var/lib/docker/volume/qualys-jira-volume' unless the Docker environment has custom settings.

a) You can check the docker volume path by executing the command, *docker volume inspect <volume-name>*

- The docker volume contains a directory '_data' which has the following dedicated child directories.

a) Config

b) Db

c) Logs

d) Output

e) Templates

- The Jira connector will also create a network in your environment named 'qualys-jira-connector', this network will be used by both containerized services to communicate with each other. You can verify this by executing the command, *docker network ls*.

Jira Connector Services

Let's see what the newly spun up containerized services are capable of.

Qualys Client service:

This service is responsible for bringing in the host detection data from Qualys platform and transform that data into JSON format periodically. These files will be placed in the output directory of the mounted Jira Connector volume.

Jira Client Service:

- Based on your Jira Instance type, the Jira Client performs the following actions:

Jira onCloud: Create custom issue types and custom field into your jira instance with respect to your preferred ticketing scheme.

Jira onPremise: Begins fetching issue types and custom fields information from your Jira Instance.

Note: The above actions are a one-time task for the Jira Client Service.

- Next, the Jira Client reads the JSON files in the output directory of the mounted Jira connector volume to create, update or transition tickets into your Jira Instance.

Now that you know what the installed services do, let's look at the configurations required for the services to perform the above actions.

Configurations

By this point, you should be ready with your installed application. So let's begin by providing the necessary configurations.

- Navigate to your Jira Connector Docker volume > Open '_data' > 'config'

- Select config.json.template > Rename to config.json

- Open the renamed config.json file via text editor and provide the following inputs:

a. Add your Qualys credentials

b. Specify whether the Jira is hosted “onCloud” or “onPremise”

c. Add your Jira credentials

d. Add proxy network information, if required.

e. Set the Profile parameters:

- “active”: Set this parameter to true or false to enable/disable creating tickets for that profile.
- “frequencyInMinutes”: Set the interval (in minutes) for Qualys client service to make periodic API calls to fetch data from the qualys platform .
- “filter”: You can set filters to fetch specific data from the Qualys platform. The filters must be provided in a URL-encoded format.

Note: For Host Detection, default filters are *action=list&status=New,Active,Re-Opened, Fixed&output_format=XML&vm_processed_after=<timestamp>&vm_processed_before=<timestamp>* and these should not be re-entered in the config file.

- “projectKey” : Set the project key from your Jira instance under which the tickets are to be created.
- “ticketingScheme” : You can choose between inputs “1” or “2” for this parameter. Refer to Understanding Ticketing Schemes to get a better understanding of ticketing schemes.

Note: We do not create tickets for knowledgebase. We only create tickets for host detection but we still use the knowledgebase data to fetch few fields for detection tickets such as diagnosis or solution.

- Save the file.

When the config.json file is saved, both containerized services - Qualys client service and Jira client service begins reading it and performs the actions as mentioned in Jira Connector Services.

Known Issues

Due to limitation of SQLite DB, service may run into DBLock/SQLite-readonly error temporarily until DB is available for read/write operation.

Things to Know

- We have tested Jira connector against following versions.

	Version
Jira Cloud	It does not have a version, we have conducted our testing on the latest available Jira cloud as of Jan'2023 https://community.atlassian.com/t5/Jira-questions/How-to-find-out-the-current-Jira-cloud-version/qaq-p/837432
Jira On-Premise	v9.4.0#940000-sha1:da47b38
Docker Compose	v2.13.0

- All the number fields in Jira by default shows comma in it. Hence, host and QID is displayed with comma.

Refer the links below to learn more.

https://community.atlassian.com/t5/Jira-Service-Management/How-to-remove-commas-in-a-Number-type-Custom-Field/qaq-p/1750542

https://community.atlassian.com/t5/Jira-Software-questions/Remove-Commas-from-Custom-field-Number-type/qaq-p/1749175

Debugging and Troubleshooting

Why do I see this error while executing docker-compose up command "yaml: line 6: did not find expected key"

Cause: Issue with syntax of the docker-compose.yml file.

Solution: Correct the docker-compose.yml file. The easiest way would be to copy paste the yml content from Jira connector Github as is. Alternatively, use the syntax validators yamllint on your linux machine or the online yaml syntax validator to correct the yaml syntax.

Why do I see this error when I execute docker-compose up command with valid YAML file “Error response from daemon: manifest for <image-name:tag> not found: manifest unknown: manifest unknown"

Cause: Image mentioned in the docker-compose.yml file is not present on docker hub.

Solution: Verify if the image names used in your local docker-compose.yml file and Jira connector's Github docker-compose.yml file have same image names.
If both of them are correct, check if your host needs proxy to connect with docker hub which could be blocking docker pull operation.

Frequently Asked Questions

Why do I need docker compose? Can I use a simple docker run instead?

Using docker compose is recommended as

- the Jira connector application requires multiple containers to run at the same time

- there is a startup dependency between the two containers that spin for this application

However, if you still lean towards, docker run, you can execute the following commands to achieve the same,

Step 1: Create a bridged network using the command

```
docker network create -d bridge qualys-jira-connector
```

Step 2: Create named volume using command

```
docker volume create qualys-jira-volume
```

Step 3: Spin Qualys client service container using command

```
docker run -d --name=qualys-client -itd --network=qualys-jira-
connector -v qualys-jira-volume:/opt/qualys/common/jiraconnector/
<qualys-client-service-image-name:tag>
```

Step 4: Spin Qualys Jira client service container using command

```
docker run -d --name=jira-client -itd --network=qualys-jira-
connector -v qualys-jira-volume:/opt/qualys/common/jiraconnector/
<qualys-jira-client-service-image-name:tag>
```

After step 4, your application should be up and running.

As soon as I rename the config.json.template file from linux terminal, error is logged for both of the services, why is it so?

Cause: File is renamed correctly but at this point config.json file does not contain valid input yet hence error is logged.

Solution: Refer to Configurations.

How do both the service containers connect?

They will connect through the network defined in docker-compose file. Refer to Post-Installation.

How do I run multiple copies of a Compose file on the same host?

Docker compose has command line option of '-p' using which you can specify the project name. Eg. docker-compose -p jira-connector-1 up.

To run multiple instance using the same compose file, all you have to do is change the value of '-p' parameter. But, be careful of,

- not abusing the api limit for both Qualys and Jira APIs.

- additional consumption of resources such as memory, disk space etc as for each new instance of application seperate volumes and network will be created.

Hence, it is only advised to run multiple instances when keeping the above points in mind.

Do I lose my data when the Jira connector's service containers exits?

Not at all! Any data that the application writes to the volume placed on your host gets preserved in its volume directory until you explicitly delete the volume.

What if I re-updated the config.json file?

Every instance of re-updating the config.json file triggers both of the services from scratch. Qualys Client service will re-initiate the schedule and fetch for Host detection/knowledgebase data. Whereas, Jira Client service tries to create custom issue types and fields in the instance, unless they are already present in the instance.

How do I report any issue with Qualys?

You can contact Qualys support to report any issue related to the Jira connector.

While reporting an issue, make sure you provide the following details:

- A detailed summary of the issue
- The docker-compose.yml file you are using
- Following evidences from docker volume (qualys-jira-volume/_data)
    - Complete logs for Qualys client and Jira client services.
    - config file (post removing passwords in it)
    - db
    - ticketing template files used
    - output files, if any (even if in errored state)

- In case of a startup error, provide complete error triggered during startup, along with:
    - Base OS of the host where you are setting up the Jira Connector application
    - Docker Version
    - Docker compose installation type and version
    - Socket on which docker runs (Eg. UNIX, TCP)
- Output of the following commands:
    - docker inspect volume <jira-connector-volume-name>
    - docker inspect network <jira-connector-network-name>
    - docker inspect <qualys-client-service-container>
    - docker inspect <jira-client-service-container>

Quick JQLs For Your Reference

Sr. no	Usage	JQL
1	Find issues created between a specified timeframe.	project = "<projectkey>" AND issuetype = "Vulnerable Host" AND created >= "2023-01-09" AND created <= "2023-01-13" ORDER BY created DESC
2	Find Issues which are 10 days old.	project = "<projectkey>" AND issuetype = "Host Vulnerability" AND created < -10d
3	Find issues created in last 30 mins.	project = "<projectkey>" AND issuetype = "Vulnerable Host" AND created < -0.5h
4	Find the list of linked detection tickets for given host ticket.	project = “<projectkey>” AND issue in linkedIssues("JP2-1326")
5	Fetch list of all the hosts and related tickets.	project = "<projectkey>" AND issuetype IN ("Vulnerable Host", "Vulnerability") AND issueLinkType = "relates to"
6	Find the list of issues for which detection severity falls in certain range.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "Severity[Number]" IN (3, 4,5)
7	Find the list of issues of based on certain severity.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "Severity[Number]" = 5
8	Find detection issues that are patchable.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "Patchable[Short text]" ~ yes
9	Find detection issues that are NOT patchable.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "Patchable[Short text]" !~ yes
10	Find detection issues that are of PCI detections.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "PCI Flag[Short text]" ~ yes
11	Find detection issues that are NOT of PCI detections.	project = "<projectkey>" AND issuetype IN ("<custom-issue-type>") AND "PCI Flag[Short text]" !~ yes

Custom Fields for Ticketing Schemes

Ticketing Scheme 1 – Host-Vuln Linking

Ticketing Scheme	Issue Type	Field Name	Field Type	Search-able
1-Host_Vuln_Linking_Tick-et_Scheme	Vulnera-ble Host	Host ID	Number	Yes
		Asset ID	Number	Yes
		IP	Text - single line	Yes
		IPV6	Text - single line	Yes
		Tracking Method	Text - single line	Yes
		OS	Text - single line	Yes
		Last Scan Datetime	Text - single line	Yes
		Last VM Scanned Date	Text - single line	Yes
		Asset Tag	Labels	Yes
	Vulnera-bility	QID	Number	Yes
		Port	Number	Yes
		Severity	Number	Yes
		Vuln Type	Text - single line	Yes
		Patchable	Text - single line	Yes
		PCI Flag	Text - single line	Yes

		Vuln Category	Text - single line	Yes
		Published Datetime	Text - single line	Yes
		CVSS Base	Number	Yes
		CVSS Temporal	Number	Yes
		CVSS V3 Base	Number	Yes
		CVSS V3 Temporal	Number	Yes
		Last Service Modification Datetime	Text - single line	Yes
		CVEs	Text - Multi line	Yes
		Diagnosis	Text - Multi-line	Yes
		Consequence	Text - Multi-line	Yes
		Solution	Text - Multi-line	Yes

Ticketing Scheme 2 – Per Detection Separate

Ticketing Scheme	Issue Type	Field Name	Field Type	Search-able
2 - Per Detection Separate Ticket Scheme	Host Vul-nerability	Host ID	Number	Yes
		Asset ID	Number	Yes
		IP	Text - single line	Yes
		IPV6	Text - single line	Yes
		Tracking Method	Text - single line	Yes
		OS	Text - single line	Yes
		Last Scan Datetime	Text - single line	Yes
		Last VM Scanned Date	Text - single line	Yes
		Asset Tag	Labels	Yes
		QID	Number	Yes
		Port	Number	Yes
		Severity	Number	Yes
		Vuln Type	Text - single line	Yes
		Patchable	Text - single line	Yes
		PCI Flag	Text - single line	Yes
		Vuln Cate-gory	Text - single line	Yes
		Published Datetime	Text - single line	Yes
		CVSS Base	Number	Yes
		CVSS Tem-poral	Number	Yes

	Detection Status	Text-single line	Yes
	CVSS V3 Base	Number	Yes
	CVSS V3 Temporal	Number	Yes
	Last Service Modification Date-time	Text - single line	Yes
	CVEs	Text - Multi line	Yes
	Diagnosis	Text - Multi-line	Yes
	Conse-quence	Text - Multi-line	Yes
	Solution	Text - Multi-line	Yes