Table of Contents

About This Guide .................................................................................................................. 4
About Qualys .......................................................................................................................... 4
Qualys Support ..................................................................................................................... 4

Welcome to Qualys Jira Connector ...................................................................................... 5
Prerequisites ........................................................................................................................... 5
  Software Requirements ....................................................................................................... 5
  Qualys Requirements ......................................................................................................... 5
  Jira Requirements ............................................................................................................. 6
Understanding Ticketing Schemes ..................................................................................... 7
  Ticketing Scheme 1: Host_Vuln_Linking_Ticket_Scheme ............................................... 7
  Ticketing scheme 2: Per_Detection_Separate_Ticket_Scheme .................................... 8
For Jira Instance On-Premise ............................................................................................... 9
  Create Custom Issue Types ............................................................................................ 9
  Adding Issue Types to Issue Type Scheme ..................................................................... 10
  Creating Custom Fields ................................................................................................. 10

Get Started ........................................................................................................................... 11
Install the App ....................................................................................................................... 11
Post-Installation .................................................................................................................... 11
Jira Connector Services ....................................................................................................... 12
Configurations ..................................................................................................................... 11
Known Issues ....................................................................................................................... 12
Things to Know ................................................................................................................... 12

Debugging and Troubleshooting ......................................................................................... 14

Frequently Asked Questions .............................................................................................. 19

Quick JQLs For Your Reference .......................................................................................... 26

Custom Fields for Ticketing Schemes ................................................................................. 27
  Ticketing Scheme 1 – Host-Vuln Linking .................................................................... 27
  Ticketing Scheme 2 – Per Detection Separate ............................................................ 29

What’s New .......................................................................................................................... 31
Jira Connector v1.0.1 ......................................................................................................... 31
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.

Prerequisites

The following requirements are necessary to get started with the Jira Connector service.

Software Requirements

Required software to get started with the connector.

- You should have installed Docker compose.

Qualys Requirements

Required Qualys configurations to get started with the connector.

- 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 URL: /api/2.0/fo/knowledge_base/vuln/?action=list

<table>
<thead>
<tr>
<th>Role</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager, Unit Manager, Scanner, Reader</td>
<td>Download vulnerability data from the KnowledgeBase.</td>
</tr>
<tr>
<td>Auditor</td>
<td>No permission to download vulnerability data from the KnowledgeBase.</td>
</tr>
</tbody>
</table>
Host Detections

**API URL:** /api/2.0/fo/asset/host/vm/detection/

<table>
<thead>
<tr>
<th>Role</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managers</td>
<td>View all VM scanned hosts in subscription</td>
</tr>
<tr>
<td>Unit Managers</td>
<td>View VM scanned hosts in the user’s assigned business unit.</td>
</tr>
<tr>
<td>Scanners and Readers</td>
<td>View VM scanned hosts in the user’s account.</td>
</tr>
<tr>
<td>Auditors</td>
<td>Have no permission to view VM scanned hosts.</td>
</tr>
</tbody>
</table>

**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**

Required Jira configurations to get started with the connector.

- 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’.
- Select Default Screen scheme for projects where the tickets are to be created.
- 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 prerequisites For Jira Instance On-Premise.

The type of ticketing schemes you select helps 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.

On successful creation of tickets with Ticketing Scheme 1, your host and vulnerability details are displayed as shown below.

Vulnerable Host Ticket:
Vulnerability Ticket:

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.

- 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 failure while creating tickets.
- However, ‘workflowMappings’ section can be updated as per the workflow mapping in your Jira instance.

On successful creation of tickets with Ticketing Scheme 2, your host and vulnerability details are displayed as shown below.
Host Vulnerability:

For Jira Instance On-Premise

Now that you are aware of the available ticketing schemes, let’s look at the additional prerequisites 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 are:
   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 are:
   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 prerequisites 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

Follow the steps below to install the Jira Connector application.

1. Go to the GitHub repository of the Jira connector application
2. Copy the docker-compose.yml file to your local directory on the machine where you intend to setup the Jira connector application.
3. 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 for a sample output of the above commands.

4. To verify installation, execute the command, `docker ps`

   ![Sample Output]

   **Note:** As a part of the behavior of Docker compose, your local directory name with the docker-compose.yml are 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

Now that you have the Jira Connector service installed, you can find the following in your machine.

1. 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

2. The Jira connector mounts a docker volume named 'qualys-jira-volume' in your environment. You can check the volume by executing the command, `docker volume ls`
3. 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>*

4. The docker volume contains a directory '_data' which has the following dedicated child directories.
   a) Config
   b) Db
   c) Logs
   d) Output
   e) Templates

5. The Jira connector also creates a network in your environment named 'qualys-jira-connector', this network is 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 are 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.

1. Navigate to your Jira Connector Docker volume > Open '_data' > 'config'
2. Select config.json.template > Rename to config.json
3. 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.

   **Note:** Upon updating and saving the config.json files, the Qualys, Jira and proxy passwords are encrypted and replaced with a static text.

   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.

4. 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.

**Note:** If you prefer uploading your own config.json file rather than updating the existing template, you might need to provide additional permissions for the file. We recommend setting the minimum permission as 706. You can set this permission by running `chmod 706 config.json` from Jira connector's config directory. Learn more.

**Known Issues**

Issues identified with the connector service and solutions you can implement.

- Due to limitation of SQLite DB, the Jira Connector may run into DBLock/SQLite-readonly error temporarily. Please wait until services are able to access the DB for further read/write operations.

- Uploading a config.json file requires you to set additional write permissions to the file. To set the permission on Linux distributions, run the command `chmod 706 config.json`.

- Due to limitation from the watcher service, the Jira Connector may not read config files updated through the Linux editor. We recommend using tools like Winscp to update the file.

**Things to Know**

We have tested Jira connector against following versions.

<table>
<thead>
<tr>
<th>Service</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jira Cloud</td>
<td>It does not have a version, we have conducted our testing on the latest available Jira cloud as of Jan'2023 (<a href="https://community.atlassian.com/t5/Jira-questions/How-to-find-out-the-current-Jira-cloud-version/qaq-p/837432">https://community.atlassian.com/t5/Jira-questions/How-to-find-out-the-current-Jira-cloud-version/qaq-p/837432</a>)</td>
</tr>
<tr>
<td>Jira On-Premise</td>
<td>v9.4.0#940000-sha1:da47b38</td>
</tr>
<tr>
<td>Docker Compose</td>
<td>v2.13.0</td>
</tr>
</tbody>
</table>

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-Software-questions/Remove-Commas-from-Custom-field-Number-type/qa-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”**

**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.

**Jira Client logs error as “Field cannot be set. It is not on the appropriate screen, or unknown”**

**Cause:**
1. The error in Jira connector’s Jira client service refers to the default screen scheme.
2. If any of the custom fields or default fields (eg. description) are not part of this 'Default screen scheme' then, while creating a ticket, the Jira API used to create issue can not find this field and throws the error mentioned above.

**Solution:**
Check the fields for which error being thrown in logs

1. We print Error response from Jira API as is which contains only field id hence to check what custom field id refer to which field name can be findout using following curl call.

   Curl call to fetch all the fields in Jira instance

   ```bash
   curl --request GET --url '<jira-instance-url>/rest/api/latest/field' --user 'email@email.com:<API-token>' -H 'Accept: application/json' > output.json
   ```

   2. output.json created in above curl call contains all the fields in your Jira instance

   3. You can search using custom field id in this json file to find out the field which is in error.
Now, that you have list of your fields ready with you, Go to jira instance (needs admin login here)

4. Navigate to Project setting > Screens
5. On this screen, click **Edit** option in front of 'Default Screen Scheme'.

6. On 'Configure Screen Scheme — Default Screen Scheme' click **Default Screen**.

7. Now, you land on 'Configure Screen'.
You can see a list of all the field on this screen. We need to add the fields for errors that are being thrown to this list.

To do so, move down to bottom of the page, you can find an option to add field.

Type the field name, it should suggest the respective field, select the suggested field.
Once all the fields are added to this list, from this point onwards, Jira client should not face the screen-related error.

**Jira client is throwing the an error while creating ticket as = "Profile ID or Project key is null. Cannot proceed further."**

**Cause:**

A DB lock error prevents the Jira Client Service from updating the profile information (Profile ID, Project, etc) in the database.

**Solution:**

Restart the Jira Client Service to update the DB with the profile information. Follow the below commands to achieve this.

Stop the Jira Client service,

```
docker container stop <Jira-client-container id/name>
```

or

```
docker-compose stop <Jira-client-service-container-name>
```

Start the container again,

```
docker container start <Jira-client-container id/name>
```

or

```
docker-compose start <Jira-client-service-container-name>
```

**After uploading config.json, I observed ‘config.json (Permission denied)’ error in Qualys client Service Logs**

**Cause:**

As of Jira connector v1.0.1, passwords provided in config.json are encrypted by Qualys client service. After encryption, the service replaces actual passwords in config.json with a static string saying, "Password has been encrypted".

Default owner of the config.json file is 'app' (Jira connector App user - 555) with permissions `rw-r--r--` (644). When the file is uploaded to the docker-volume-path/_date/config/, its owner changes from 'app' (Jira connector App user - 555) to root.

Hence, when Qualys client service tries to replace the 'password' in config.json with static string post encrypting your password, it runs into 'Insufficient permissions to write into config.json file'.

**Solution:**

Provide required permissions to the config.json file at 'docker-volume-path/_date/config/' dir

Minimum set of permissions required are 706, which you can provide from the Linux terminal by using the command `chmod 706 config.json`
On winscp, your permissions matrix for file should look like as follows
How do I upgrade Jira connector application to use the latest docker images?

If you are using docker compose,

1. Stop both of the running containers of Jira Connector application using 'docker-compose down' command.

2. Update your docker-compose.yml file with latest images of Jira connector application, (you can get the latest images in docker-compose.yml file maintained on Github.

3. V1.0.1 also supports 'QDS' score, hence if you want pull QDS information, you can update your respective ticketing scheme template as,
   a) Navigate to from <docker-volume-dir>/_data/templates/
   b) Open your selected ticketing scheme template json file
   c) For ticketing scheme 2,
      • Open Per_Detection_Separate_Ticket_Scheme.json
      • Navigate to its 'custom_fields' section and add "QDS": "${QDS.content}" 
      • Save the file
   a) For ticketing scheme 1
      • Open Host_Vuln_Linking_Ticket_Scheme.json
      • Navigate to 'relatedTicket' section > 'custom_fields' section and add "QDS": "${QDS.content}" 
      • Save the file

4. Spin the Jira connector application once again using 'docker-compose up' or 'docker-compose up -d'.

5. This downloads a new images in your environment and also bring the new changes (if any) into your docker volume for eg. addition of new table in application db.

If you are not using docker compose,

1. if you are not using docker compose, then you use this command to stop both of the running containers 'docker container stop <container name>'

2. Check the new images released for Jira connector application, (you can get the latest images in docker-compose.yml file maintained on Github.

3. Pull latest images into your environment.

4. Steps to update the template file for QDS support are same as above in step 3 of docker compose configuration.

5. Spin up the containers once again using the following commands,
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,

1. Create a bridged network using the command
   
   ```
   docker network create -d bridge qualys-jira-connector
   ```

2. Create named volume using command

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

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>
   ```

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 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. Example,

```bash
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 separate volumes and network are 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.

Can I update the ticketing template file?
Yes, you can. You can either customize the template with or without adding any fields.

**Customizing the existing template without adding any new field**
1. Navigate to ticketing template-> Open the template that you have selected (for ticketing scheme 1-> Host_Vuln_Linking_Ticket_Scheme.json and ticketing scheme 2-> Per_Detection_Separate_Ticket_Scheme.json).
2. Update the template as per requirement.
3. Save the template file.
4. These template changes are effective upon the next scheduled run in which the Qualys client creates new output files.

**Customizing the template to add new fields in the template**
1. Users can update only the ‘custom_fields’ section of the template with additional fields.
2. Navigate to the ticketing template -> Open the template that you have selected (for ticketing scheme 1 -> Host_Vuln_Linking_Ticket_Scheme.json and ticketing scheme 2 -> Per_Detection_Separate_Ticket_Scheme.json.)
3. Users can select any fields from the given list to update the template with additional fields.

<table>
<thead>
<tr>
<th>Host</th>
<th>NETWORK_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NETBIOS</td>
</tr>
<tr>
<td></td>
<td>QG_HOSTID</td>
</tr>
<tr>
<td></td>
<td>LAST_VM_AUTH_SCANNED_DATE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detection</th>
<th>LAST_TEST_DATETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FIRST_FOUND_DATETIME</td>
</tr>
<tr>
<td></td>
<td>LAST_FIXED_DATETIME</td>
</tr>
<tr>
<td></td>
<td>IS_DISABLED</td>
</tr>
<tr>
<td></td>
<td>TIMES_FOUND</td>
</tr>
<tr>
<td></td>
<td>SSL</td>
</tr>
<tr>
<td></td>
<td>IS_IGNORED</td>
</tr>
<tr>
<td></td>
<td>LAST_PROCESSED_DATETIME</td>
</tr>
<tr>
<td></td>
<td>LAST_UPDATE_DATETIME</td>
</tr>
</tbody>
</table>

4. Now, in the template file, add a field name to the 'custom_fields' section and for its expected value, select the field from the above list

   a) Eg. To add the field TIMES_FOUND, you can add an entry similar to the following in the template file,

   • "No of times Detection found":"${TIMES_FOUND}".

5. These template changes are effective upon the next scheduled run in which the Qualys client creates new output files.

6. Restart your Jira client service container for the field to populate in Jira Connector’s database

   a) For Jira cloud, user-added fields are created by the Jira client service of default type 'Text field- single line'.

7. For Jira on-premise, fields are expected to be created by users on their Jira instance.

   **Note:** While rephrasing, the template must not use ' '-'<hyphen> as the JQL used in API throws error for hyphen character.

**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 re-initiates 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.
What is jira-connector-config? Do I need to make any changes to it manually?

The jira-connector-config directory contains the file jira_connector_default_config.json. As a user, you should not make any changes to this file.

Jira connector uses a flag (true/false) in this file as the communication bridge between Qualys Client service container and Jira Client service container to provide status of the encryption.

My Ticketing scheme selection is 1 and In Jira Client service I see count logs for total tickets processed higher than the actual count.

For ticketing scheme 1, each Host Vulnerability ticket is processed/updated for each related Vulnerability ticket. Every time a Host Vulnerability ticket is updated, the count of processed ticket is increased.

So, the final 'Total tickets processed count' is higher than actual created/updated tickets.

Can I remove the Keys from config.json file which am not using

No, you can not. Removal of any keys leads to errors when running the app.

How would a valid config.json file with all inputs correctly filled look like?

We recommend not removing any input field from the json file. If a field has no value, leave it empty like this : "filter": ""

Valid config.json sample with authenticated proxy and API filters

```json
{
  "credentials" : {
    "qualys" : {
      "apiUrl" : "https://qualysapi.qg2.apps.qualys.com",
      "username" : "qualys-user",
      "password" : "qualys-password"
    },
    "jira" : {
      "instanceUrl" : "https://jira-instance.atlassian.net/",
      "username" : "admin@jira.com",
      "apiToken" : "jira-api-token",
      "instanceType" : "onCloud"
    },
    "proxy" : {
      "host" : "10.10.10.10",
      "port" : "3128",
      "username" : "proxy-user",
      "password" : "proxy-password"
    }
  },
}```
"profile" : [ {
    "profileId" : "HD",
    "name" : "Host Detection",
    "active" : true,
    "frequencyInMinutes" : 30,
    "filter" : "show_asset_id=1&show_tags=1",
    "projectKey" : "TS1",
    "ticketingScheme" : 1
  }, {
    "profileId" : "KB",
    "name" : "Knowledgebase",
    "active" : true,
    "frequencyInMinutes" : 30,
    "filter" : "",
    "projectKey" : "TS1",
    "ticketingScheme" : 1
  }
]

Valid config.json sample with un-authenticated proxy and No API filters

{
    "credentials" : {
        "qualys" : {
            "apiUrl" : "https://qualysapi.qg2.apps.qualys.com",
            "username" : "qualys-user",
            "password" : "qualys-password"
        },
        "jira" : {
            "instanceUrl" : "https://jira-instance.atlassian.net/",
            "username" : "admin@jira.com",
            "apiToken" : "jira-api-token",
            "instanceType" : "onCloud"
        },
        "proxy" : {
            "host" : "10.10.10.10",
            "port" : "3128",
            "username" : "",
            "password" : ""
        }
    },
    "profile" : [ {
        "profileId" : "HD",
        "name" : "Host Detection",
        "active" : true,
        "frequencyInMinutes" : 30,
        "filter" : ""
    }
]
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

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

### Ticketing Scheme 1 – Host-Vuln Linking

<table>
<thead>
<tr>
<th>Ticketing Scheme</th>
<th>Issue Type</th>
<th>Field Name</th>
<th>Field Type</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Host_Vuln_Linking_Ticket_Scheme</td>
<td>Vulnerable Host</td>
<td>Host ID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asset ID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPV6</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracking Method</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last Scan Datetime</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last VM Scanned Date</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asset Tag</td>
<td>Labels</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Vulnerability</td>
<td>QID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Port</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Severity</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vuln Type</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Patchable</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCI Flag</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom Field</td>
<td>Data Type</td>
<td>Nullable?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vuln Category</td>
<td>Text - single line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Published Datetime</td>
<td>Text - single line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS Base</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS Temporal</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS V3 Base</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS V3 Temporal</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Service Modification Datetime</td>
<td>Text - single line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVEs</td>
<td>Text - Multi line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Text - Multi line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consequence</td>
<td>Text - Multi line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution</td>
<td>Text - Multi line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Ticketing Scheme 2 – Per Detection Separate

<table>
<thead>
<tr>
<th>Ticketing Scheme</th>
<th>Issue Type</th>
<th>Field Name</th>
<th>Field Type</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 - Per Detection Separate</td>
<td>Host Vulnerability</td>
<td>Host ID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td>Ticket Scheme</td>
<td></td>
<td>Asset ID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPV6</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracking Method</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OS</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last Scan Datetime</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last VM Scanned Date</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Asset Tag</td>
<td>Labels</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QID</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Port</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Severity</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vuln Type</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Patchable</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCI Flag</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vuln Category</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Published Datetime</td>
<td>Text - single line</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVSS Base</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVSS Temporal</td>
<td>Number</td>
<td>Yes</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detection Status</td>
<td>Text - single line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS V3 Base</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVSS V3 Temporal</td>
<td>Number</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Service Modification Date-time</td>
<td>Text - single line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVEs</td>
<td>Text - Multi line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Text - Multi-line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consequence</td>
<td>Text - Multi-line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution</td>
<td>Text - Multi-line</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s New

Jira Connector v1.0.1
- The passwords provided as part of the Qualys and Jira client setup are now encrypted and replaced with static text. The config.json file no longer displays passwords in plain text.
- The Jira Connector supports syncing QDS field for creating issues in Jira.