Clarifications on APIs usage

Overview

If you aim to interact with Xray data, directly or indirectly, there are several APIs you can use for different purposes.

But first, you need to answer:

1. **Am I using Xray on Jira Server/Datacenter or Xray on Jira Cloud?** This is crucial as these are different products, using different technologies, providing slightly different APIs.
2. **What do I aim to do?** This is important in order to select the proper API to invoke; the format of the API call may be different even if there are similar API endpoints for the other deployment type.
3. **Which authentication mechanism do I want to use on the API calls?** This is mostly dependent on the previous answers.

This page helps you answer these questions.

In the end, you'll find all relevant references with links to the proper documentation.

> **Please note**
>
> This article aims to provide you with the means to identify the Xray and Jira variants you have, along with the respective APIs that you can use. It also details the available authentication mechanisms.
>
> This is neither a "best practices" article nor extensive API documentation; please check the relevant API documentation for the latter (references at the end).

Xray Server/DC vs Xray Cloud

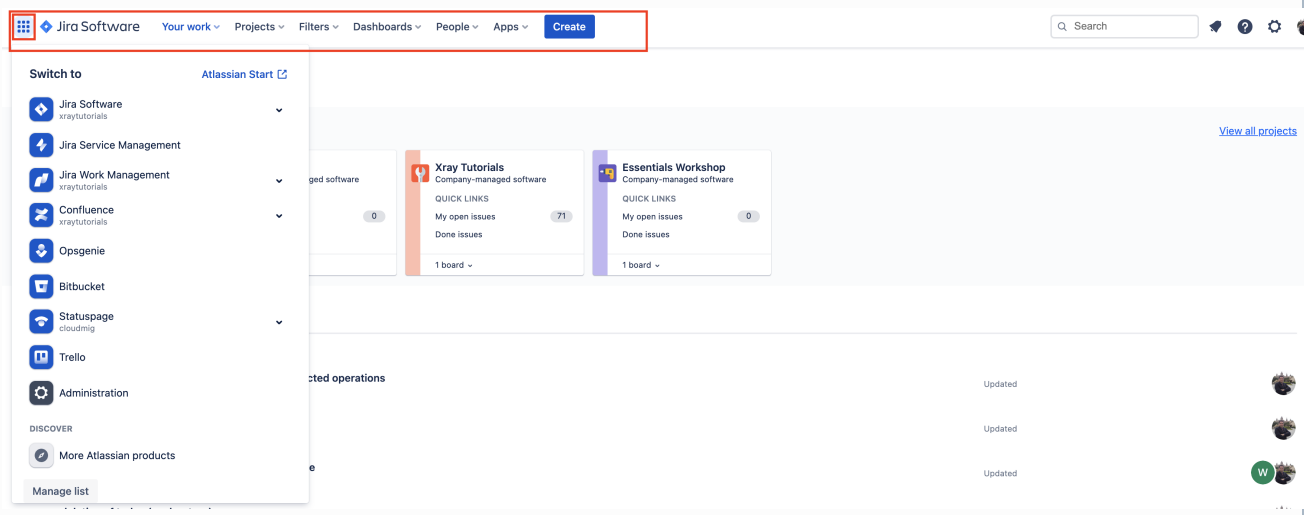
Although similar, Xray for Jira server/Data Center(DC) and Xray for Jira Cloud are different products, essentially because they are built on top of different infrastructures and application capabilities. Jira Server/Datacenter and Jira Cloud are also distinct products, with different roadmaps, built using distinct technologies, and providing different APIs. This has a consequence that apps for Jira Server/DC and for Jira Cloud are essentially totally different from an architecture standpoint, but eventually also from a feature perspective.

The Xray Cloud product closely integrates with Jira Cloud; each one uses its own infrastructure, the first managed by Xray and the latter managed by Atlassian.

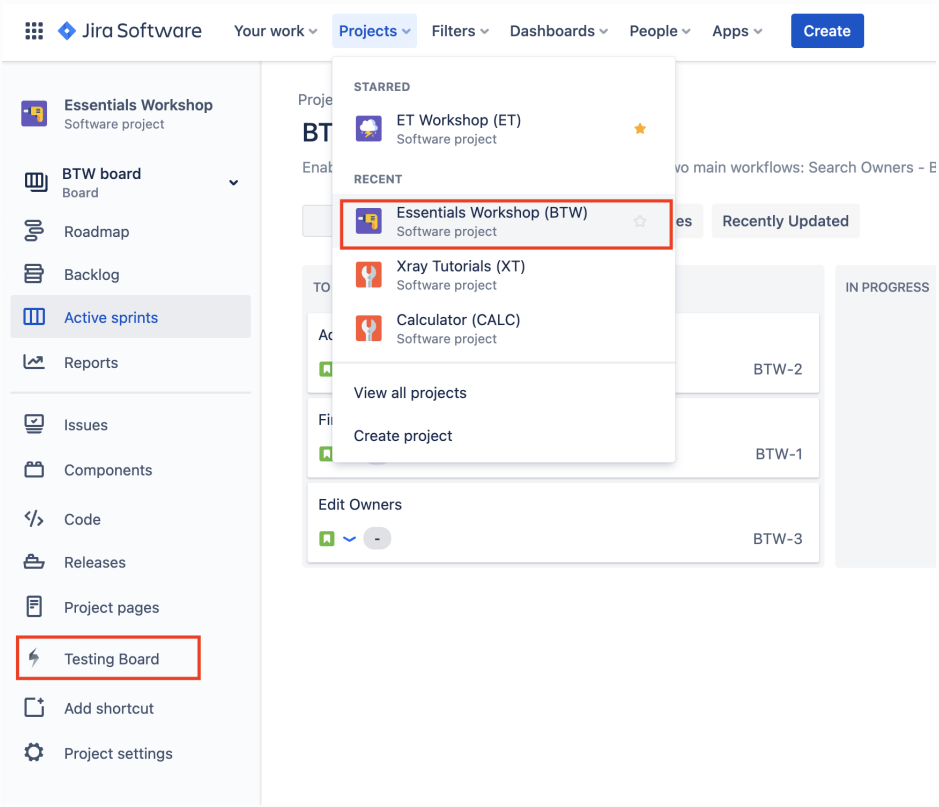
Xray server/DC works on top of (i.e., deployed into) Jira server/DC, reusing the Jira server/DC infrastructure.

> **Am I using Jira server/datacenter or Jira Cloud?**
>
> So, now that you know that you're using Jira or Xray but you don't know exactly which "flavor" (i.e., deployment type) you're using.
>
> First, you can ask your Jira administrator. You can also easily check for yourself.
>
> **Jira Cloud and Xray for Jira Cloud**

If you see a top menu like the following one, having on the top-left corner to switch between Atlassian apps, followed by "Your work", "Filters", ... "Apps", then you are using a Jira Cloud instance.



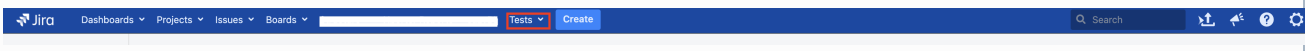
If you click on a project that is using Xray, then the project left side menu will show you a "Testing Board" shortcut (and no Xray Reports, Xray Test Repository, Xray Test Plan Board, or Automated Steps Library). This tells you are using Xray on a Jira Cloud instance (i.e., "Xray Cloud").



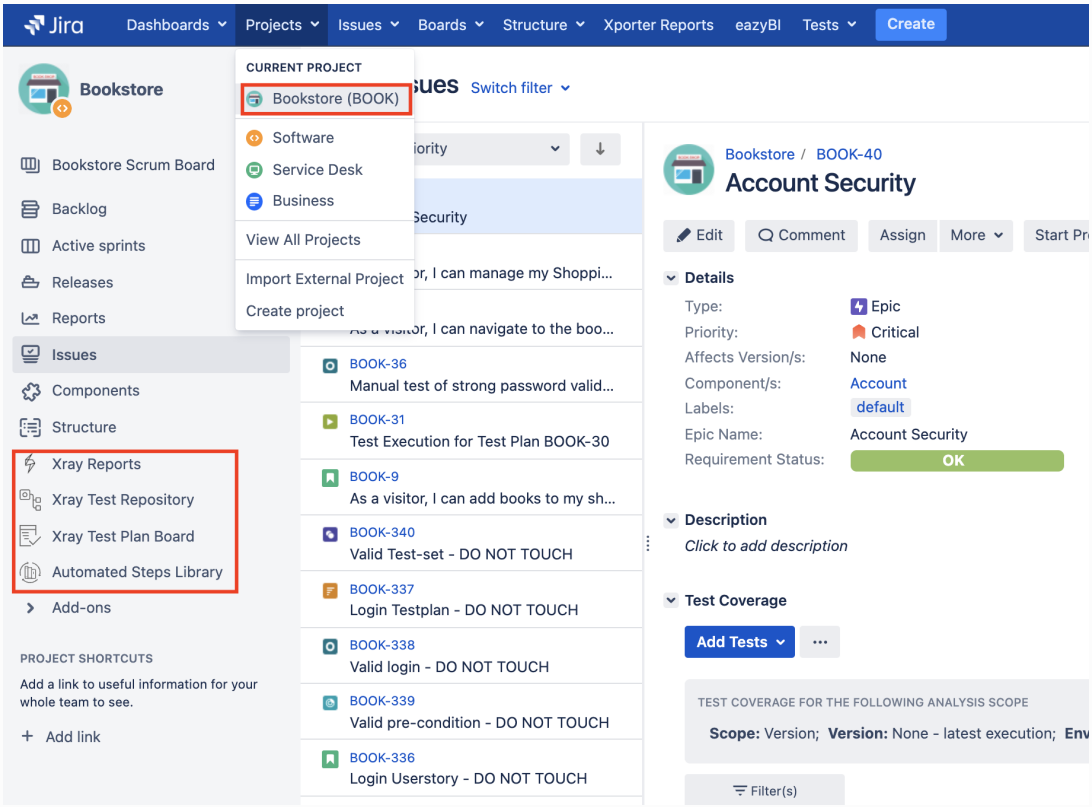
The documentation for Xray Cloud can be found here.

Jira Server/Datacenter and Xray for Jira Server/Datacenter

If you see a top menu like the following one, having on the top-left corner an icon, just to go some Jira page (depending on your profile settings), followed by "Dashboard", "Projects", "Issues", ..., and then "Tests", then you are using a Jira server/datacenter instance. The "Tests" top menu shortcut only appears if you have Xray (for Jira server/datacenter) installed.



If you click on a project that is using Xray, then the project left side menu will show you some options from Xray. Xray Reports, Xray Test Repository, Xray Test Plan Board, and Automated Steps Library are great tips that tell you you are using Xray on a Jira server/datacenter instance.



The documentation for Xray server/DC can be found here.

Now that you know the Jira "flavor" (i.e., deployment type) being used and the corresponding Xray product variant (whether Xray Cloud or Xray server /DC), let's see which APIs are available in that context, how they work and for that, and what they can be used for.

Available APIs

There are several APIs that users can take advantage of.

APIs can either be provided by Xray or by Jira itself.

Most users will use the REST API provided by Xray to import test automation results during CI/CD. Please have in mind that, in this case, Xray Cloud's REST API is a different API than the Xray Server/DC REST API.

The Jira REST API is only used in case you want to perform operations on Jira data, such as standard CRUD operations (Create, Read, Update, Delete) on issues of, for example, search for issues using JQL. Jira Cloud and Jira Server/DC have similar (not equal) REST APIs.

APIs available in Xray Cloud context

Xray Cloud provides a REST API and also a, more advanced, GraphQL API.

Besides, there is also Jira's REST API in case you aim to interact with Jira itself and, for example, do CRUD operations on issues (except for testing data).

	API provided by	type of API	versions	URL syntax	authentication	purpose	notes

1	Xray	REST API	v1, v2	https://xray.cloud.getxray.app/api/v1/<resource_name> https://xray.cloud.getxray.app/api/v2/<resource_name>	managed by Xray Cloud • API key (i.e., pair of Client Id + Client Secret created by Jira administrator)	• **authentication** • **importing test results** • importing tests • import/exporting Cucumber scenarios • backups	This is the most used API, as it is used for importing test automation results.
2	Xray	GraphQL	v2	https://xray.cloud.getxray.app/api/v2/graphql	managed by Xray Cloud • API key (i.e., pair of Client Id + Client Secret created by Jira administrator) Note: to make GraphQL requests, an initial REST API call needs to be made to the authentication endpoint	• CRUD operations for Xray entities, with access to all data, including the creation of tests • obtain Xray entities and other entities related to them, and manage these associations • export test results	This API cannot be used to import test automation results. This API is more advanced and usually only required whenever implementing some custom scenarios.
3	Jira	REST API	v3	https://<site_url>/rest/api/3/<resource_name> Note: <site_url> can be something like <xxx>.atlassian.net or it can be your own domain, in case you have it defined for your Jira cloud instance.	managed by Jira Cloud • basic-authentication (using a Jira username) • OAuth 2.0	• standard CRUD operations on Jira entities, including Jira issues (which includes Xray issue-based entities)	This API does not provide ways of accessing/modifying internal Xray data, such as test steps, for example.

API available in Xray server/DC context

Xray Server/DC only provides a REST API, used to import test automation results and also to obtain and manage relations between Xray entities.

Besides, there is also Jira's REST API in case you aim to interact with Jira itself and, for example, do CRUD operations on issues (except for testing data).

Please note

- Xray server/DC doesn't provide a GraphQL API
- Xray server/DC REST API is different from Xray Cloud REST API; some endpoints may be similar, like for importing test automation results, but there will be differences
- Xray server/DC doesn't provide API Keys (pair client id + client secret), as it happens with Xray Cloud. Please check instead Personal Access Tokens (since v8.14, for Jira DC only)

	API provided by	type of API	versions	URL syntax	authentication	purpose	notes
1	Xray	REST API	v1.0, v2.0	<jira_base_url>/rest/raven/1.0/api/<resource_name> <jira_base_url>/rest/raven/2.0/api/<resource_name>	managed by Jira server/DC • Basic authentication (using a Jira username) • Personal Access Tokens (since v8.14, for Jira DC only) • OAuth 1.0a	• **importing test results** • import/exporting Cucumber scenarios • obtain Xray entities and other entities related to them, and manage these associations • export test results	This is the most used API, as it is used for importing test automation results.
2	Jira	REST API	latest	<jira_base_url>/rest/api/latest/<resource_name>	managed by Jira server/DC • Basic authentication (using a Jira username) • Personal Access Tokens (since v8.14, for Jira DC only) • OAuth 1.0a	• standard CRUD operations on Jira entities, including Jira issues (which includes Xray issue-based entities)	This API does not provide ways of accessing/modifying internal Xray data, such as test steps, for example.

Xray REST API v1 and v2 on Xray server/DC. What are the differences?

The REST API v2.0 for Xray server/DC is an extension to the original v1.0 of the REST API. Therefore, all existing endpoints on v1.0 also exist on v2.0 (even if you don't see that in the docs); you just need to replace the "1.0" by "2.0" on the URL.

Example:

<jira_base_url>/rest/raven/1.0/api/test/{key}/testexecutions **becomes** <jira_base_url>/rest/raven/2.0/api/test/{key}/testexecutions

Use cases

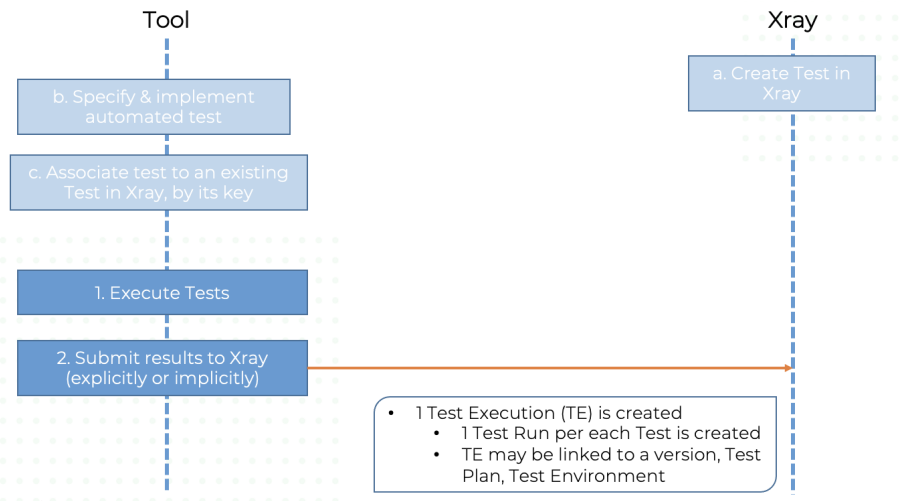
Importing test automation results to Xray using Xray JSON format

Importing automation results into Xray is a very common use case.

There are a bunch of test report formats supported (e.g., JUnit XML, Xray JSON, etc), which may be slightly different for Xray Cloud and for Xray server/DC . Depending on the report format and whether you're using Xray Cloud or server/DC, different capabilities will be available (server/DC, cloud).

In this case, we'll use the Xray JSON format as means to encapsulate the test results.

At a high level, the flow can be depicted in the following diagram.



Xray JSON format has the same syntax but to the distinct nature of Jira Server/DC and Jira Cloud (and also Xray), there are some nuances.

	Xray Server/DC	Xray Cloud	Notes
values for "status" attribute, for each "test" and "step" elements	"PASS", "FAIL", "TODO", "EXECUTING"	"PASSED", "FAILED", "TO DO", "EXECUTING"	
"user" attribute	username of Jira user	name of Jira user.	Usually, you don't need to specify the user since by default it is assigned to the user related to the given credentials. With Xray Cloud, you can use the "userId" attribute instead and in that case, you need to use the hash of the user (you can see it as the ending part of the URL of the user profile)

In this case, we're assuming:

- Test issue(s) already exist in Xray (they could have been created as usual by a user or by some automation process)
- An automated test equivalent exists in the external tool/code, for each Test

And we want to...

- report test automation result back to the existent Test issue(s)

The following request shows how to import results for two tests, identified by their issue keys. The first one failed while the second passed.

Both tests were performed against version 1.0 of the SUT, using the Chrome browser; since it is relevant for us to analyze the results later on from each browser perspective, we map the browser name to a Test Environment.

On the failed test, we leave a comment on the Test Run and also attached a screenshot as evidence.

Xray Server/DC

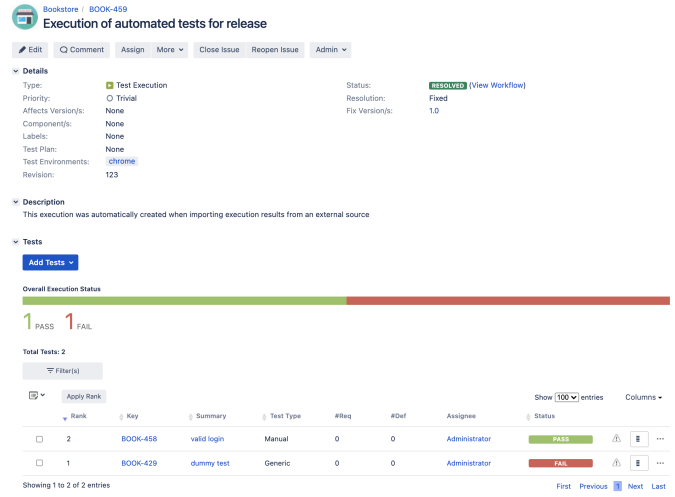
example of a API request with "curl" using basic authentication

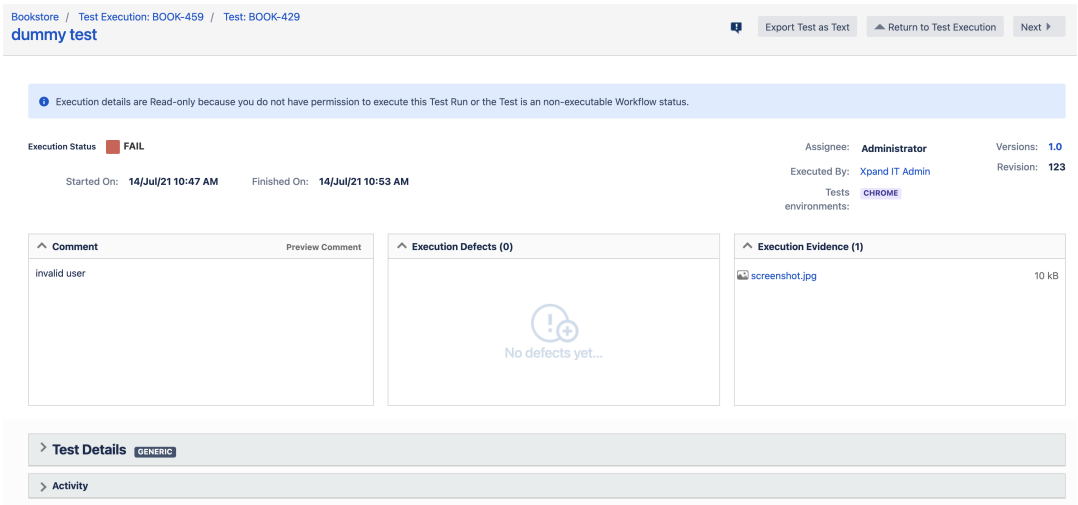
```
curl -H "Content-Type: application/json" -X POST -u jira_username:jira_password --data @payload.json
https://jiraserver.example.com/rest/raven/2.0/import/execution
```

JSON body payload

```
{
    "info": {
        "summary": "Execution of automated tests for release",
        "description": "This execution was automatically created when importing execution results from an
external source",
        "project": "BOOK",
        "version": "1.0",
        "revision": "123",
        "startDate": "2021-07-14T11:47:35+01:00",
        "finishDate": "2021-07-14T11:53:00+01:00",
        "testEnvironments": [
            "chrome"
        ]
    },
    "tests": [
        {
            "testKey": "BOOK-429",
                "start": "2021-07-14T11:47:35+01:00",
                "finish": "2021-07-14T11:53:00+01:00",
            "comment": "invalid user",
            "status": "FAIL",
            "evidences" : [
              {
               "data": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkG...."
               "filename": "screenshot.jpg",
               "contentType": "image/jpeg"
               }
            ]
        },
        {
            "testKey": "BOOK-458",
                "start": "2021-07-14T11:47:35+01:00",
                "finish": "2021-07-14T11:53:00+01:00",
            "status": "PASS"
        }
    ]
}
```

After importing results, a Test Execution would be created containing the results for the two tests; the details of the execution screen show the data of the Test Run, including its status, comment, timings, among others.





Xray Cloud

The first thing to do would be to obtain a token, using the Client Id and Client Secret from the corresponding API key on Xray.

This would require performing an HTTP request to the authentication REST API endpoint from Xray Cloud.

Then a second request could be made to upload the test automation results; in the following example, the Xray JSON is stored on a payload.json file.

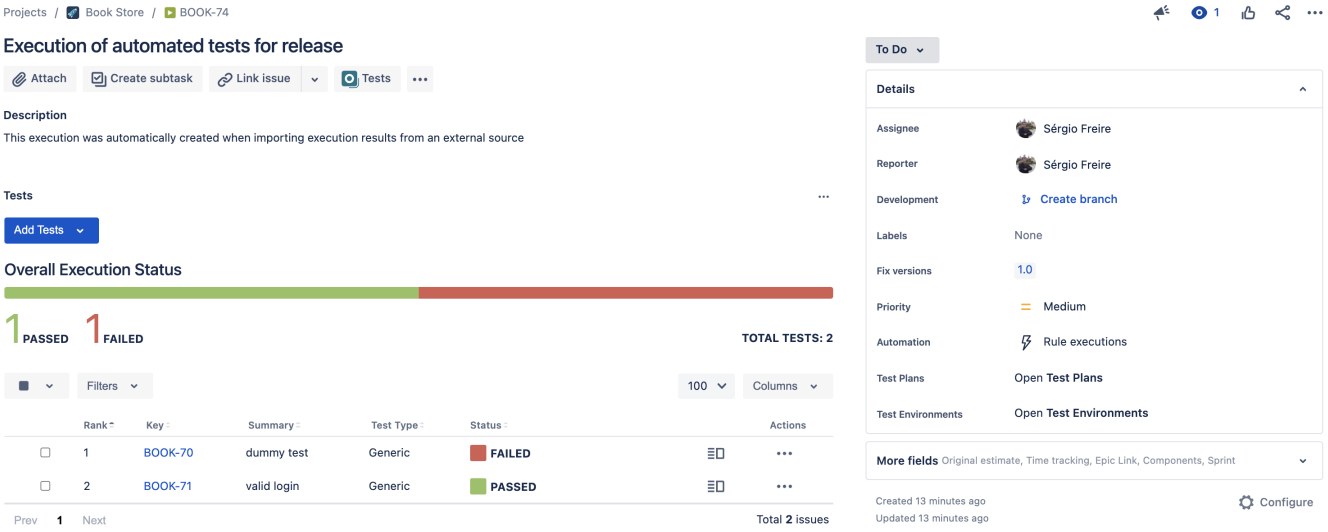
example of a API request with "curl" using basic authentication

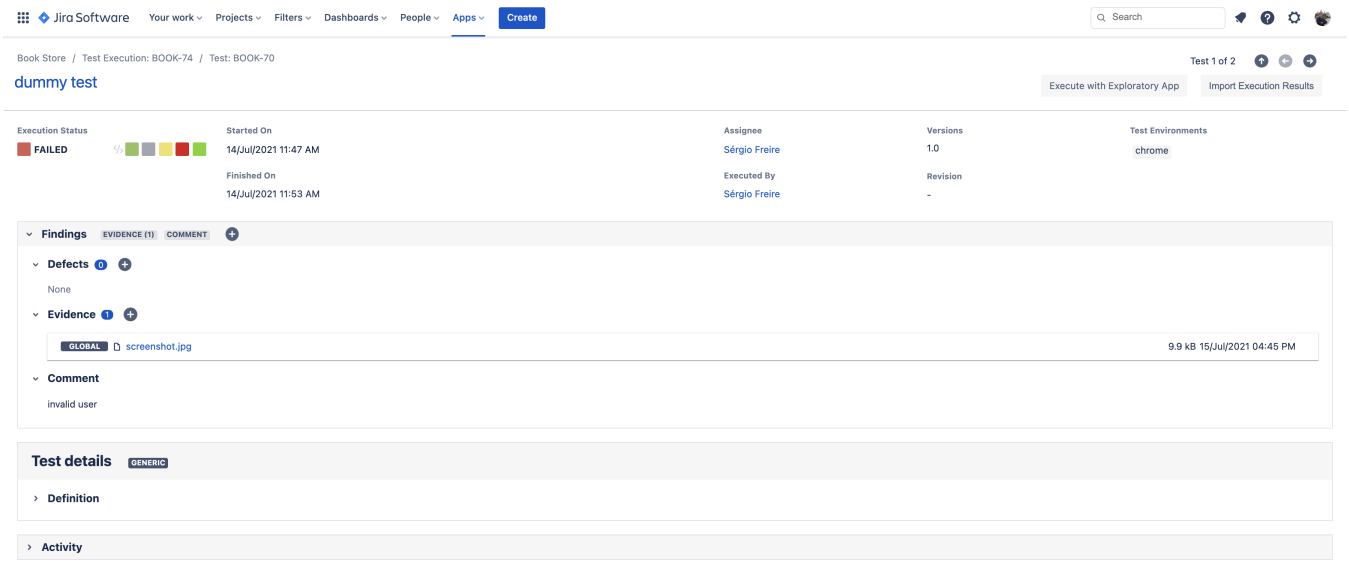
```
token=$(curl -H "Content-Type: application/json" -X POST --data '{ "client_id":
"32A27E69B0AC4E539C14016437000000","client_secret":
"d62f81eb9ed859e11e54356dd8a00e4a5f0d0c2a2b52340776f6c7d6d7000000" }'  https://xray.cloud.getxray.app/api/v2
/authenticate)

curl -H "Content-Type: application/json" -X POST -H "Authorization: Bearer $token"  --data @"payload.json"
"https://xray.cloud.getxray.app/api/v2/import/execution
```

JSON body payload

```
{
    "info": {
        "summary": "Execution of automated tests for release",
        "description": "This execution was automatically created when importing execution results from an
external source",
        "project": "BOOK",
        "version": "1.0",
        "revision": "123",
        "startDate": "2021-07-14T11:47:35+01:00",
        "finishDate": "2021-07-14T11:53:00+01:00",
        "testEnvironments": [
            "chrome"
        ]
    },
    "tests": [
        {
            "testKey": "BOOK-70",
                "start": "2021-07-14T11:47:35+01:00",
                "finish": "2021-07-14T11:53:00+01:00",
            "comment": "invalid user",
            "status": "FAILED",
            "evidences" : [
              {
               "data": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkG...."
               "filename": "screenshot.jpg",
               "contentType": "image/jpeg"
                       }
                   ]
        },
        {
            "testKey": "BOOK-71",
                "start": "2021-07-14T11:47:35+01:00",
                "finish": "2021-07-14T11:53:00+01:00",
            "status": "PASSED"
        }
    ]
}
```

After importing results, a Test Execution would be created containing the results for the two tests; the details of the execution screen show the data of the Test Run, including its status, comment, timings, among others.





Importing test automation results to Xray using other formats (non-Xray JSON)

Besides the Xray JSON format as described in the previous scenario, Xray also can import and process test automation results from reports stored in different formats (e.g., JUnit XML, NUnit XML, Cucumber JSON, etc).

This will make use of Xray Cloud REST API or Xray server/DC REST API.

In this case, what's important to have in mind is that:

- the supported test automation report formats may be different between Xray Cloud and Xray Server/DC
- the authentication mechanism is different in Xray Cloud and on Xray Server/DC
- HTTP POST request for the REST API call is slightly different for Xray Cloud and Xray Server/DC

Therefore, make sure you are using the proper documentation of the REST API, either Xray Cloud REST API or the Xray server/DC REST API one.

There's an open-source repository on GitHub with some code samples, in different languages, that you can check out 🙂

Note: the import of Cucumber or Behave JSON reports follows a different flow than the standard one. More info on that here.

References

- Xray server/DC and Jira server/DC
  - Xray server/DC main documentation site
  - APIs
    - Jira server/DC REST API
    - Xray server/DC REST API
- Xray Cloud and Jira Cloud
  - Xray Cloud main documentation site
  - API's
    - Jira Cloud REST API
    - Xray Cloud REST API
    - Xray Cloud GraphQL API
- Xray Code Snippets (open-source project with code samples, using different languages)
- Xray Postman collections (Postman collections showcasing usage of APIs both for Xray server/DC and Xray Cloud, including REST and GraphQL examples)
- Other useful resources
  - RestMan (a Chrome extension very useful to try out REST API call)
  - Postman (API client with good support for REST APIs)
  - Insomnia (API client that provides good support for GraphQL)