SituationReport

get_data

User Manual

Export Jira data for SituationReport

---

situation-report -- github.com/Jaegerfeld/situation-report

For Agile Coaches and PI Managers -- Version 0.19.0

1 Exporting Data from Jira Cloud

> Note: The get_data module is still under development. This manual describes the manual export of Jira data via the Jira REST API. This is the same process that get_data will automate — the exported JSON files can be processed directly by transform_data.

This chapter explains how to export real issue data from Jira Cloud for SituationReport. The export produces the same JSON structure as the Testdata Generator — transform_data processes both identically.

1.1 What you need

Requirement	Details
Jira Cloud account	An account with read access to the desired project
API token	Personal token from id.atlassian.com — created once
Project key	The Jira project shortcode, e.g. 'ART_A' or 'SCRUM'
curl or browser	curl for automated queries; browser for quick tests

1.2 Creating an API token

The API token replaces your password for API calls and is created once:

Step	Action
1	Open https://id.atlassian.com in your browser (while logged into Jira)
2	Security → API tokens → Create API token
3	Enter a label, e.g. 'SituationReport', click 'Create'
4	Copy the token and store it safely — it is only shown once!

Security note: Treat the API token like a password. Store it in a password manager and do not share it.

1.3 Querying the Jira REST API

Jira Cloud offers two API variants. Use the current v3 API for new scripts; existing v2 scripts continue to work. `expand=changelog` is required in both variants — without it, status transitions are missing and transform_data cannot compute time-in-stage values.

Current API (v3) – recommended

Endpoint: `POST /rest/api/3/search/jql`

Parameters are passed as a JSON request body. The response additionally includes `isLast` and `nextPageToken` for pagination (Section 1.5).

```
curl -X POST \
  "https://company.atlassian.net/rest/api/3/search/jql" \
```

```
  -u "name@company.com:YourAPIToken" \
  -H "Content-Type: application/json" \
  -d '{"jql":"project=ART_A ORDER BY created ASC",
       "maxResults":100,
       "expand":["changelog"],
       "fields":["issuetype","created",
                 "status","summary"]}' \
  -o ART_A_page1.json
```

maxResults is limited to 100 issues per request with the v3 API. Use nextPageToken for projects with more than 100 issues (Section 1.5).

Older API (v2) – still supported

Endpoint: `GET /rest/api/2/search`

Parameters as URL query string. Supports up to 1,000 issues per request and offset-based pagination with `startAt`.

```
curl -u "name@company.com:YourAPIToken" \
  "https://company.atlassian.net/rest/api/2/search?\
jql=project=ART_A+ORDER+BY+created+ASC\
&expand=changelog\
&maxResults=1000\
&fields=issuetype,created,status,project,summary,resolution" \
  -o ART_A_page1.json
```

Tip: For new scripts, the v3 API is preferred. Existing v2 scripts do not need to be migrated.

1.4 Fields required by transform_data

Jira field	Required	Purpose
key	□	Issue identifier (e.g. ART_A-001)
fields.issuetype.name	□	Issue type for filtering (Feature, Bug, ...)
fields.created	□	Issue creation date
fields.status.name	□	Current status
changelog (expand=changelog)	□	Status transitions with timestamps — MUST be set in the URL
fields.resolution	–	Optional — resolution type (e.g. 'Done', 'Won't Fix')
fields.summary	–	Optional — issue title

1.5 Pagination – more than 100 issues

API v3 – Cursor-based pagination (nextPageToken)

The v3 API returns at most 100 issues per request. Each response contains either a `nextPageToken` for the next page, or `"isLast": true` when all issues have been fetched. Pages must be retrieved sequentially — the token from the current response is used in the next request.

```
# Page 1 – no nextPageToken
curl -X POST \
  "https://company.atlassian.net/rest/api/3/search/jql" \
  -u "name@company.com:Token" \
  -H "Content-Type: application/json" \
  -d '{"jql":"project=ART_A ORDER BY created ASC",
       "maxResults":100,"expand":["changelog"],
       "fields":["issuetype","created",
                 "status","summary"]}' \
  -o ART_A_page1.json

# Read nextPageToken from page 1:
#   cat ART_A_page1.json | grep nextPageToken
#   (or with jq: jq -r '.nextPageToken' ART_A_page1.json)

# Page 2 – insert nextPageToken
curl -X POST \
  "https://company.atlassian.net/rest/api/3/search/jql" \
  -u "name@company.com:Token" \
  -H "Content-Type: application/json" \
  -d '{"jql":"project=ART_A ORDER BY created ASC",
       "maxResults":100,"expand":["changelog"],
       "fields":["issuetype","created",
                 "status","summary"],
       "nextPageToken":"<token from page 1>"}' \
  -o ART_A_page2.json

# Repeat until isLast:true appears in the response

# Merge all pages
python -m helper ART_A_page1.json ART_A_page2.json ... \
  --output ART_A_merged.json
```

Repeat until the response contains `"isLast": true`. The Helper merges all pages and removes duplicates automatically.

API v2 – Offset-based pagination (startAt) – Legacy

The v2 API supports up to 1,000 issues per request. With `startAt`, any page can be accessed directly. Important: `startAt` is only compatible with the v2 endpoint — not with the new v3 endpoint.

```
# Page 1 (issues 1–1,000)
curl -u "name@company.com:Token" \
  "https://company.atlassian.net/rest/api/2/search?\
jql=project=ART_A&expand=changelog&maxResults=1000&startAt=0" \
```

```
  -o ART_A_page1.json

# Page 2 (issues 1,001–2,000)
curl -u "name@company.com:Token" \
  "https://company.atlassian.net/rest/api/2/search?\
jql=project=ART_A&expand=changelog&maxResults=1000&startAt=1000" \
  -o ART_A_page2.json

# Merge files with the Helper
python -m helper ART_A_page1.json ART_A_page2.json \
  --output ART_A_merged.json
```

Increment `startAt` by 1,000 until the response contains fewer than 1,000 issues.

1.6 From JSON export to workflow file

After exporting, create a workflow file that maps your actual Jira statuses. The following Python snippet extracts all status names from the export:

```
import json
data = json.loads(open('ART_A_page1.json').read())
statuses = set()
for issue in data['issues']:
    for h in issue['changelog']['histories']:
        for item in h['items']:
            if item['field'] == 'status':
                statuses.add(item['toString'])
print(sorted(statuses))
```

From the status names found, create a `workflow.txt` — arrange the statuses in logical order and set the `<First>` and `<Closed>` markers. The workflow file format is described in the transform_data User Manual.

2 Frequently Asked Questions (FAQ)

What happens if expand=changelog is missing?

transform_data finds no status transitions and outputs empty time values for all issues. The Excel files contain no usable data. Always include `expand=changelog` in the URL.

I get an 'Unauthorized' error when calling curl.

Check: Is the email address correct? Is the API token correct? Are they separated by a colon? Does your account have read access to the project? Tip: Generate a new API token if the old one was lost or expired.

Can I merge multiple Jira projects?

Yes — either via JQL (`jql=project in (ART_A, ART_B)`) for a single export, or merge two separate exports with the Helper. Make sure the workflows of both projects are compatible.

How large can the JSON file get?

Rule of thumb: approximately 2–5 KB per issue (depending on changelog length). 1,000 issues ≈ 2–5 MB. For very large projects (10,000+ issues), use pagination and the Helper.

Should I use test data or real data?

For demos, training, and installation tests, synthetic test data from the Testdata Generator is the better choice — no privacy concerns, reproducible, and fully controlled. For real flow analyses and decisions, actual Jira data reflects the true workflow state.

3 Glossary

Term	Explanation
ART	Agile Release Train — a team-of-teams structure in SAFe
API	Application Programming Interface — programming interface
API token	Security key for accessing the Jira REST API; replaces the password in curl calls
Changelog	Jira log of all status changes for an issue (requires expand=changelog)
CFD	Cumulative Flow Diagram — shows the cumulative issue count per stage over time
Closed stage	The stage that marks an issue as completed (workflow.txt: <Closed>)
Cycle time	Time from the first active stage (<First>) to the closed stage
First stage	First active stage where cycle time measurement begins (workflow.txt: <First>)
Helper	SituationReport module for merging multiple Jira JSON files
Issue	A work item in Jira (Feature, Bug, Enabler, Story, …)
JQL	Jira Query Language — query language for Jira searches, e.g. project=ART_A
JSON	JavaScript Object Notation — format of Jira API exports
Stage	One step in the workflow (corresponds to a Jira column or status name)
Workflow	Ordered sequence of stages that an issue passes through