# SituationReport

## get_data

### Benutzerhandbuch

Jira-Daten für SituationReport exportieren

---

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

Fuer Agile Coaches und PI Manager -- Version 0.19.0

# 1 Daten aus Jira Cloud exportieren

> Hinweis: Das Modul get_data ist noch in Entwicklung. Dieses Handbuch beschreibt den manuellen Export von Jira-Daten über die Jira REST API. Der Exportprozess ist derselbe, den get_data später automatisiert — die exportierten JSON-Dateien sind direkt mit transform_data verarbeitbar.

Dieser Abschnitt erklärt, wie Sie echte Issue-Daten aus Jira Cloud für SituationReport exportieren. Der Export liefert dieselbe JSON-Struktur wie der Testdata Generator — transform_data verarbeitet beide identisch.

## 1.1 Was Sie benötigen

| Voraussetzung | Details |
|---|---|
| Jira Cloud Zugang | Ein Konto mit Lesezugriff auf das gewünschte Projekt |
| API-Token | Persönliches Token von id.atlassian.com — einmalig erstellen |
| Projekt-Key | Kürzel des Jira-Projekts, z. B. 'ART_A' oder 'SCRUM' |
| curl oder Browser | curl für automatisierte Abfragen; Browser für einfache Tests |

## 1.2 API-Token erstellen

Das API-Token ersetzt bei API-Abfragen Ihr Passwort und wird einmalig erstellt:

| Schritt | Aktion |
|---|---|
| 1 | https://id.atlassian.com im Browser aufrufen (in Jira eingeloggt sein) |
| 2 | Security → API tokens → Create API token |
| 3 | Einen Namen eingeben, z. B. 'SituationReport', auf 'Create' klicken |
| 4 | Token kopieren und sicher speichern — er wird nur einmal angezeigt! |

*Sicherheitshinweis: Behandeln Sie den API-Token wie ein Passwort. Speichern Sie ihn in einem Passwort-Manager und teilen Sie ihn nicht mit anderen.*

## 1.3 Die Jira REST API abfragen

Jira Cloud bietet zwei API-Varianten. Für neue Skripte empfiehlt sich die aktuelle v3-API; bestehende v2-Skripte funktionieren weiterhin. `expand=changelog` ist in beiden Varianten Pflicht — ohne ihn fehlen die Statusübergänge, die transform_data benötigt.

### *Aktuelle API (v3) – empfohlen*

Endpoint: `POST /rest/api/3/search/jql`
Parameter werden als JSON im Request-Body übergeben. Die Antwort enthält zusätzlich `isLast` und `nextPageToken` für die Paginierung (Abschnitt 1.5).

```
curl -X POST \
  "https://firma.atlassian.net/rest/api/3/search/jql" \
  -u "name@firma.de:IhrAPIToken" \
  -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 ist bei der v3-API auf 100 Issues pro Anfrage begrenzt. Verwenden Sie nextPageToken für Projekte mit mehr als 100 Issues (Abschnitt 1.5).*

### *Ältere API (v2) – weiterhin unterstützt*

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

Parameter als URL-Query-String. Unterstützt bis zu 1.000 Issues pro Anfrage und Offset-Paginierung mit `startAt`.

```
curl -u "name@firma.de:IhrAPIToken" \
  "https://firma.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
```

*Tipp: Für neue Skripte empfiehlt sich die v3-API. Bestehende v2-Skripte müssen nicht migriert werden.*

## 1.4 Welche Felder transform_data benötigt

| Jira-Feld | Pflicht | Verwendung |
|---|---|---|
| key | □ | Issue-Kennung (z. B. ART_A-001) |
| fields.issuetype.name | □ | Issue-Typ für Filterung (Feature, Bug, ...) |
| fields.created | □ | Erstellungsdatum des Issues |
| fields.status.name | □ | Aktueller Status |
| changelog (expand=changelog) | □ | Statusübergänge mit Zeitstempeln — MUSS im URL-Parameter gesetzt sein |
| fields.resolution | – | Optional — Auflösungstyp (z. B. 'Done', 'Won't Fix') |
| fields.summary | – | Optional — Issue-Titel |

## 1.5 Paginierung – mehr als 100 Issues

### *API v3 – Cursor-basierte Paginierung (nextPageToken)*

Die v3-API liefert maximal 100 Issues pro Anfrage. Jede Antwort enthält entweder einen `nextPageToken` für die nächste Seite oder `"isLast": true`, wenn alle Issues abgerufen wurden. Seiten müssen sequenziell abgerufen werden — das Token aus der aktuellen Antwort wird im nächsten Request eingesetzt.

```
# Seite 1 – ohne nextPageToken
curl -X POST \
  "https://firma.atlassian.net/rest/api/3/search/jql" \
  -u "name@firma.de: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

# nextPageToken aus Seite 1 auslesen:
#   cat ART_A_page1.json | grep nextPageToken
#   (oder mit jq: jq -r '.nextPageToken' ART_A_page1.json)

# Seite 2 – nextPageToken einfuegen
curl -X POST \
  "https://firma.atlassian.net/rest/api/3/search/jql" \
  -u "name@firma.de: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 aus Seite 1>"}' \
  -o ART_A_page2.json

# Wiederholen bis isLast:true in der Antwort

# Alle Seiten zusammenfuehren
python -m helper ART_A_page1.json ART_A_page2.json ... \
  --output ART_A_merged.json
```

Wiederholen Sie den Vorgang, bis die Antwort `"isLast": true` enthält. Der Helper führt alle Seiten zusammen und entfernt Duplikate automatisch.

### *API v2 – Offset-basierte Paginierung (startAt) – Legacy*

Die v2-API unterstützt bis zu 1.000 Issues pro Anfrage. Mit `startAt` kann jede Seite direkt angesprungen werden. Wichtig: `startAt` ist nur mit dem v2-Endpunkt kompatibel — nicht mit dem neuen v3-Endpunkt.

```
# Seite 1 (Issues 1–1000)
curl -u "name@firma.de:Token" \
```

```
  "https://firma.atlassian.net/rest/api/2/search?\
jql=project=ART_A&expand=changelog&maxResults=1000&startAt=0" \
  -o ART_A_page1.json

# Seite 2 (Issues 1001–2000)
curl -u "name@firma.de:Token" \
  "https://firma.atlassian.net/rest/api/2/search?\
jql=project=ART_A&expand=changelog&maxResults=1000&startAt=1000" \
  -o ART_A_page2.json

# Dateien mit dem Helper zusammenfuehren
python -m helper ART_A_page1.json ART_A_page2.json \
  --output ART_A_merged.json
```

Erhöhen Sie `startAt` schrittweise um 1.000, bis das Ergebnis weniger als 1.000 Issues enthält.

## 1.6 Vom JSON-Export zur Workflow-Datei

Nach dem Export müssen Sie eine Workflow-Datei erstellen, die Ihre tatsächlichen Jira-Status abbildet. Das folgende Python-Skript extrahiert alle Status-Namen aus der JSON-Datei:

```
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))
```

Aus den gefundenen Status-Namen erstellen Sie eine `workflow.txt` — ordnen Sie die Status in logischer Reihenfolge an und setzen Sie `<First>` und `<Closed>`. Das Format der Workflow-Datei ist im Benutzerhandbuch von transform_data beschrieben.

# 2 Häufige Fragen (FAQ)

## Was passiert, wenn expand=changelog fehlt?

transform_data findet keine Statusübergänge und gibt für alle Issues leere Zeitwerte aus. Die Excel-Ausgaben enthalten keine nutzbaren Daten. Stellen Sie sicher, dass `expand=changelog` immer in der URL steht.

## Ich erhalte den Fehler 'Unauthorized' beim curl-Aufruf.

Prüfen Sie: E-Mail-Adresse und API-Token korrekt? Sind sie durch einen Doppelpunkt getrennt? Hat Ihr Konto Lesezugriff auf das Projekt? Tipp: Einen neuen API-Token generieren, falls der alte abgelaufen ist oder verloren wurde.

## Kann ich mehrere Jira-Projekte zusammenführen?

Ja — entweder über JQL (`jql=project in (ART_A, ART_B)`) für einen einzigen Export, oder mit dem Helper mehrere separate Exporte zusammenführen. Achten Sie darauf, dass die Workflows beider Projekte kompatibel sind.

## Wie groß kann die JSON-Datei werden?

Faustregel: ca. 2–5 KB pro Issue (abhängig von der Changelog-Länge). 1.000 Issues entsprechen ca. 2–5 MB. Bei sehr großen Projekten (10.000+ Issues) empfiehlt sich Paginierung und der Einsatz des Helpers.

## Soll ich Testdaten oder echte Daten verwenden?

Für Demos, Schulungen und Installationstests sind synthetische Testdaten aus dem Testdata Generator die bessere Wahl — keine Datenschutzbedenken, reproduzierbar und kontrolliert. Für echte Flow-Analysen und Entscheidungen spiegeln echte Jira-Daten den tatsächlichen Workflow-Zustand wider.

# 3 Glossar

| Begriff | Erklärung |
|---|---|
| ART | Agile Release Train — ein Team aus mehreren Scrum-Teams in SAFe |
| API | Application Programming Interface — Programmierschnittstelle |
| API-Token | Sicherheitsschlüssel für den Zugang zur Jira-REST-API; ersetzt das Passwort bei curl-Abfragen |
| Changelog | Jira-Protokoll aller Statusänderungen eines Issues (expand=changelog) |
| CFD | Cumulative Flow Diagram — zeigt den Issuebestand je Stage kumuliert über die Zeit |
| Closed Stage | Die Stage, die ein Issue als abgeschlossen markiert (workflow.txt: <Closed>) |
| Cycle Time | Zeit von der ersten aktiven Stage (<First>) bis zur Closed-Stage |
| First Stage | Erste aktive Stage, ab der die Cycle Time zählt (workflow.txt: <First>) |
| Helper | SituationReport-Modul zum Zusammenführen mehrerer Jira-JSON-Dateien |
| Issue | Arbeitselement in Jira (Feature, Bug, Enabler, Story, …) |
| JQL | Jira Query Language — Abfragesprache für Jira-Suchen, z. B. project=ART_A |
| JSON | JavaScript Object Notation — Format der Jira-API-Exporte |
| Stage | Ein Schritt im Workflow (entspricht einer Jira-Spalte oder einem Status-Namen) |
| Workflow | Geordnete Abfolge von Stages, die ein Issue durchläuft |