SituationReport

get_data

Manuel d'utilisation

Exporter des données Jira pour SituationReport

---

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

Pour les Agile Coaches et les PI Managers -- Version 0.19.0

1 Exporter des données depuis Jira Cloud

> Remarque : Le module get_data est encore en cours de développement. Ce manuel décrit l'export manuel de données Jira via l'API REST Jira. C'est le même processus que get_data automatisera — les fichiers JSON exportés peuvent être traités directement par transform_data.

Ce chapitre explique comment exporter des données réelles d'issues depuis Jira Cloud pour SituationReport. L'export produit la même structure JSON que le Générateur de données de test — transform_data traite les deux de manière identique.

1.1 Ce dont vous avez besoin

Prérequis	Détails
Compte Jira Cloud	Un compte avec accès en lecture au projet souhaité
Token d'API	Token personnel depuis id.atlassian.com — créé une seule fois
Clé du projet	Le code abrégé du projet Jira, ex. 'ART_A' ou 'SCRUM'
curl ou navigateur	curl pour les requêtes automatisées ; navigateur pour les tests rapides

1.2 Créer un token d'API

Le token d'API remplace votre mot de passe pour les appels API et est créé une seule fois :

Étape	Action
1	Ouvrez https://id.atlassian.com dans votre navigateur (connecté à Jira)
2	Security → API tokens → Create API token
3	Saisissez un libellé, ex. 'SituationReport', cliquez sur 'Create'
4	Copiez le token et conservez-le en sécurité — il n'est affiché qu'une seule fois !

Note de sécurité : Traitez le token d'API comme un mot de passe. Conservez-le dans un gestionnaire de mots de passe et ne le partagez pas.

1.3 Interroger l'API REST Jira

Jira Cloud propose deux variantes d'API. Utilisez l'API v3 actuelle pour les nouveaux scripts ; les scripts v2 existants continuent de fonctionner. `expand=changelog` est obligatoire dans les deux variantes — sans lui, les transitions de statut sont absentes et transform_data ne peut pas calculer les valeurs de temps par étape.

API actuelle (v3) – recommandée

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

Les paramètres sont transmis dans le corps JSON de la requête. La réponse inclut également `isLast` et `nextPageToken` pour la 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 est limité à 100 issues par requête avec l'API v3. Utilisez nextPageToken pour les projets avec plus de 100 issues (Section 1.5).

API plus ancienne (v2) – toujours supportée

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

Paramètres sous forme de chaîne de requête URL. Supporte jusqu'à 1 000 issues par requête et la pagination basée sur offset avec `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
```

Conseil : Pour les nouveaux scripts, l'API v3 est préférée. Les scripts v2 existants n'ont pas besoin d'être migrés.

1.4 Champs requis par transform_data

Champ Jira	Obligatoire	Rôle
key	□	Identifiant de l'issue (ex. ART_A-001)
fields.issuetype.name	□	Type d'issue pour le filtrage (Feature, Bug, ...)
fields.created	□	Date de création de l'issue
fields.status.name	□	Statut actuel
changelog (expand=changelog)	□	Transitions de statut avec horodatages — DOIT être défini dans l'URL
fields.resolution	–	Optionnel — type de résolution (ex. 'Done', 'Won't Fix')
fields.summary	–	Optionnel — titre de l'issue

1.5 Pagination – plus de 100 issues

API v3 – Pagination basée sur curseur (nextPageToken)

L'API v3 renvoie au maximum 100 issues par requête. Chaque réponse contient soit un `nextPageToken` pour la page suivante, soit `"isLast": true` quand tous les issues ont été récupérés. Les pages doivent être récupérées séquentiellement — le token de la réponse courante est utilisé dans la requête suivante.

```
# Page 1 – sans 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

# Lire nextPageToken depuis la page 1 :
#   cat ART_A_page1.json | grep nextPageToken
#   (ou avec jq: jq -r '.nextPageToken' ART_A_page1.json)

# Page 2 – inserer 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 de la page 1>"}' \
```

```
  -o ART_A_page2.json

# Repeter jusqu'a ce que isLast:true apparaisse

# Fusionner toutes les pages
python -m helper ART_A_page1.json ART_A_page2.json ... \
  --output ART_A_merged.json
```

Répétez jusqu'à ce que la réponse contienne `"isLast": true`. Le Helper fusionne toutes les pages et supprime les doublons automatiquement.

API v2 – Pagination basée sur offset (startAt) – Ancien

L'API v2 supporte jusqu'à 1 000 issues par requête. Avec `startAt`, n'importe quelle page peut être accédée directement. Important : `startAt` n'est compatible qu'avec l'endpoint v2 — pas avec le nouvel endpoint v3.

```
# 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

# Fusionner les fichiers avec le Helper
python -m helper ART_A_page1.json ART_A_page2.json \
  --output ART_A_merged.json
```

Incrémentez `startAt` de 1 000 jusqu'à ce que la réponse contienne moins de 1 000 issues.

1.6 De l'export JSON au fichier de flux

Après l'export, créez un fichier de flux qui associe vos statuts Jira réels. Le fragment Python suivant extrait tous les noms de statut de l'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))
```

À partir des noms de statut trouvés, créez un `workflow.txt` — organisez les statuts dans un ordre logique et définissez les marqueurs `<First>` et `<Closed>`. Le format du fichier de flux est décrit dans

le Manuel d'utilisation de transform_data.

2 Foire aux Questions (FAQ)

Que se passe-t-il si expand=changelog est absent ?

transform_data ne trouve aucune transition de statut et produit des valeurs de temps vides pour tous les issues. Les fichiers Excel ne contiennent pas de données exploitables. Incluez toujours `expand=changelog` dans l'URL.

J'obtiens une erreur 'Unauthorized' lors de l'appel curl.

Vérifiez : L'adresse e-mail est-elle correcte ? Le token d'API est-il correct ? Sont-ils séparés par deux-points ? Votre compte dispose-t-il d'un accès en lecture au projet ? Conseil : Générez un nouveau token d'API si l'ancien a été perdu ou a expiré.

Puis-je fusionner plusieurs projets Jira ?

Oui — soit via JQL (`jql=project in (ART_A, ART_B)`) pour un seul export, soit en fusionnant deux exports séparés avec le Helper. Assurez-vous que les flux de travail des deux projets sont compatibles.

Quelle taille peut atteindre le fichier JSON ?

Règle générale : environ 2–5 Ko par issue (selon la longueur du changelog). 1 000 issues ≈ 2–5 Mo. Pour les très grands projets (10 000+ issues), utilisez la pagination et le Helper.

Dois-je utiliser des données de test ou des données réelles ?

Pour les démonstrations, la formation et les tests d'installation, les données de test synthétiques du Testdata Generator sont le meilleur choix — sans problèmes de confidentialité, reproductibles et entièrement contrôlées. Pour les analyses de flux réelles et les décisions, les données Jira réelles reflètent l'état réel du flux de travail.

3 Glossaire

Terme	Explication
ART	Agile Release Train — une structure d'équipes dans SAFe
API	Application Programming Interface — interface de programmation
Token d'API	Clé de sécurité pour accéder à l'API REST Jira ; remplace le mot de passe dans les appels curl
Changelog	Journal Jira de tous les changements de statut d'un issue (nécessite expand=changelog)
CFD	Cumulative Flow Diagram — montre le nombre cumulatif d'issues par étape dans le temps
Étape Closed	L'étape qui marque un issue comme terminé (workflow.txt : <Closed>)
Temps de cycle	Temps entre la première étape active (<First>) et l'étape closed
Étape First	Première étape active où commence la mesure du temps de cycle (workflow.txt : <First>)
Helper	Module SituationReport pour fusionner plusieurs fichiers JSON Jira
Issue	Un élément de travail dans Jira (Feature, Bug, Enabler, Story, …)
JQL	Jira Query Language — langage de requête pour les recherches Jira, ex. project=ART_A
JSON	JavaScript Object Notation — format des exports de l'API Jira
Étape	Une étape dans le flux de travail (correspond à une colonne ou un nom de statut Jira)
Flux de travail	Séquence ordonnée d'étapes par lesquelles passe un issue