

# AQUILA - Atlassian Confluence Integration through Oauth 2.0(3LO) and API Key(File Path)

## What are API Token Scopes?

Scopes define what actions an API token is allowed to perform in Atlassian apps such as Jira and Confluence. They enhance security by limiting permissions to only what's needed (e.g., read-only access to audit logs). Always use scoped tokens for AQUILA integrations—unscoped tokens are deprecated for most apps and may not support fine-grained access. For audit logs (events like user actions, config changes, or security incidents), use the specific scopes listed below. Broader scopes may be needed for other integrations (e.g., content indexing) but stick to these for basic monitoring to minimize risk.

## Prerequisites

1. Atlassian Admin Email Account
2. Atleast have a read rights linux privilege
3. Broswer (firefox,chrome)

---

## Creating an API Token with Scopes

1. Log in to <https://id.atlassian.com/manage-profile/security/api-tokens>.
2. Select "Create API token with scopes".
3. Enter a descriptive name for the token (e.g., "AQUILA- Audit Logs Monitoring").

Choose an expiration date for the token (between 1 and 365 days; consider shorter for security).

1. Select the application Atlassian Confluence.
  2. Select "Create API token with scopes".
  3. Select the scopes or permissions the token should have:
    - For Atlassian Confluence (audit logs): `read:audit-log:confluence` to access `/wiki/rest/api/audit`.
  4. Click "Create".
  5. Copy the token and save it securely. You cannot view it again after this step. If lost, generate a new one. Share only with trusted integrations like AQUILA—revoke if compromised.
-

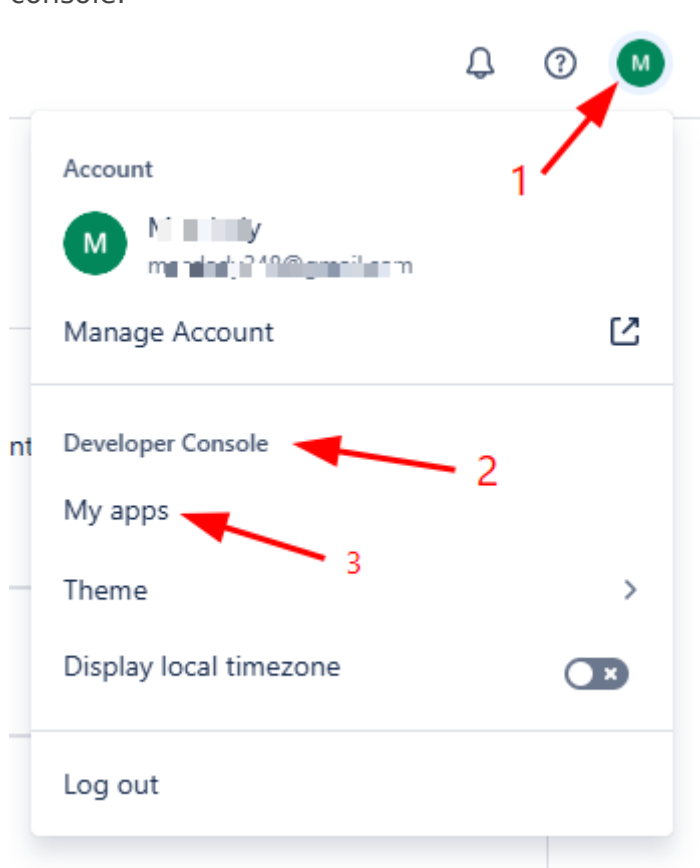
## OAuth 2.0(3LO)apps

OAuth 2.0 (3LO) (also known as "three-legged OAuth" or "authorization code grants") apps. OAuth 2.0 (3LO) allows external applications

and services to access Atlassian product APIs on a user's behalf. OAuth 2.0 (3LO) apps are created and managed in the [developer console](#).

### Enabling OAuth 2.0(3LO)

1. Select your profile icon in the top-right corner, and from the dropdown, select Developer console.



2. Select your app from the list (or create one if you don't already have one).

Show



Create ▾

Get started with Forge

OAuth 2.0 integration

Type

OAuth 2.0



#### Rotating refresh tokens are enabled

New OAuth 2.0 integrations must use rotating refresh tokens. Rotating refresh tokens improve security by limiting the validity of the refresh token and enabling automatic detection of refresh token reuse.

[Learn more](#) · [Dismiss](#)

### Create a new OAuth 2.0 (3LO) integration

An app provides API credentials for Atlassian products and services, as well as features such as OAuth 2.0 (3LO).

Name\*

████████████████████

Name your app according to its purpose, for example, Dropbox integration or Timesheets for Jira.

I agree to be bound by Atlassian's developer terms.

Create

Cancel

3. Select Permissions in the left menu.
4. Next to the API you want to add, select Configure or Add.

ATLASSIAN Developer

Elastic Integration  
OAuth 2.0 App

- Overview
- Distribution
- Permissions**
- Contributors NEW
- Authorization
- Settings

Console / All apps / Elastic Integration / Permissions

### Permissions

Add and configure your app's API scopes. See OAuth 2.0 (3LO) for apps.

Scopes Used  
2

**Scopes**  
We recommend that you don't add more than 50 scopes to your app. Use classic scopes to minimize the number of scopes you need. [Learn more](#)

[Add Marketplace or custom app](#)

API name	Scopes used	Action	
<b>Personal data reporting API</b> Report user accounts that an app is storing personal data for.	0	Add	Documentation
<b>User identity API</b> Get the profile details for the currently logged-in user, such as the Atlassian account ID and email.	0	Add	Documentation
<b>Confluence API</b> Get, create, update, and delete content, spaces, and more.	1	<b>Configure</b>	Documentation
<b>Jira API</b> Get, create, update, and delete issues, projects, fields, and more.	1	Configure	Documentation
<b>Compass GraphQL API</b> Access to the Compass GraphQL API	0	Add	Documentation
<b>BRIE API</b> Create, cancel, and read backup and restore, retrieve and publish cloud details.	0	Add	Documentation

### Edit Jira platform REST API

To edit your app's Jira platform REST API, make changes to the list below. [Learn more about scopes for OAuth 2.0 integrations](#)

**Scopes**  
We recommend that you don't add more than 50 scopes to your app. Use classic scopes to minimize the number of scopes you need. [Learn more](#) · [Dismiss](#)

1 selected

<input type="checkbox"/>	<b>View Jira issue data</b> Read Jira project and issue data, search for issues, and objects associated with issues like attachments and worklogs.	read:jira-work
<input type="checkbox"/>	<b>Manage project settings</b> Create and edit project settings and create new project-level objects (e.g. versions and components).	manage:jira-project
<input checked="" type="checkbox"/>	<b>Manage Jira global settings</b> Take Jira administration actions (e.g. create projects and custom fields, view workflows, manage issue link types).	manage:jira-configuration
<input type="checkbox"/>	<b>View user profiles</b> View user information in Jira that the user has access to, including usernames, email addresses, and avatars.	read:jira-user
<input type="checkbox"/>	<b>Create and manage issues</b> Create and edit issues in Jira, post comments as the user, create worklogs, and delete issues.	write:jira-work
<input type="checkbox"/>	<b>Manage Jira webhooks</b> Fetch, register, refresh, and delete dynamically declared Jira webhooks.	manage:jira-webhook

This change will add 0 new scopes and remove 0 scopes [Cancel](#) [Save](#)

**Edit Scopes**

1

**Edit Scopes**

**Edit Scopes**

- Select Authorization in the left menu.
- Next to OAuth 2.0 (3LO), select Configure or Add.

ATLASSIAN Developer

Elastic Integration  
OAuth 2.0 App

- Overview
- Distribution
- Permissions
- Contributors NEW
- Authorization**
- Settings

Console / All apps / Elastic Integration / Authorization

### Authorization

View the authorization configured for your app

Authorization type	Action	
OAuth 2.0 (3LO) Allows your app to access APIs for Atlassian apps and services on a user's behalf.	<b>Configure</b>	Documentation

7. Enter the Callback URL. Set this to any URL that is accessible by the app. When you implement OAuth 2.0 (3LO) in your app (see next section). in this example "base URL /callback"

**The redirect\_uri must match this URL.**

ATLASSIAN Developer

Elastic Integration  
OAuth 2.0 App

Overview  
Distribution  
Permissions  
Contributors NEW  
Authorization  
Settings

Console / My apps / Elastic Integration / Authorization

### Authorization

OAuth 2.0 authorization code grants (3LO) for apps

Configure OAuth 2.0 authorization code grants to allow your app to access data (within specific scopes)

Callback URLs\*

https://elastic-integration.atlassian.net/callback

Enter up to 30 redirect URLs, one per line

Save changes Discard changes

8. Click Save changes.
9. In **Authorization URL generator** Copy and Paste in the browser.

ATLASSIAN Developer

Console / My apps / Elastic Integration / Authorization

### Authorization

OAuth 2.0 authorization code grants (3LO) for apps

Configure OAuth 2.0 authorization code grants to allow your app to access data (within specific scopes) from Atlassian APIs on the user's behalf. Learn more about OAuth 2.0 authorization code grants for [Jira Cloud](#) and [Confluence Cloud](#).

Callback URLs\*

https://elastic-integration.atlassian.net/callback

Enter up to 30 redirect URLs, one per line

Save changes

#### Authorization URL generator

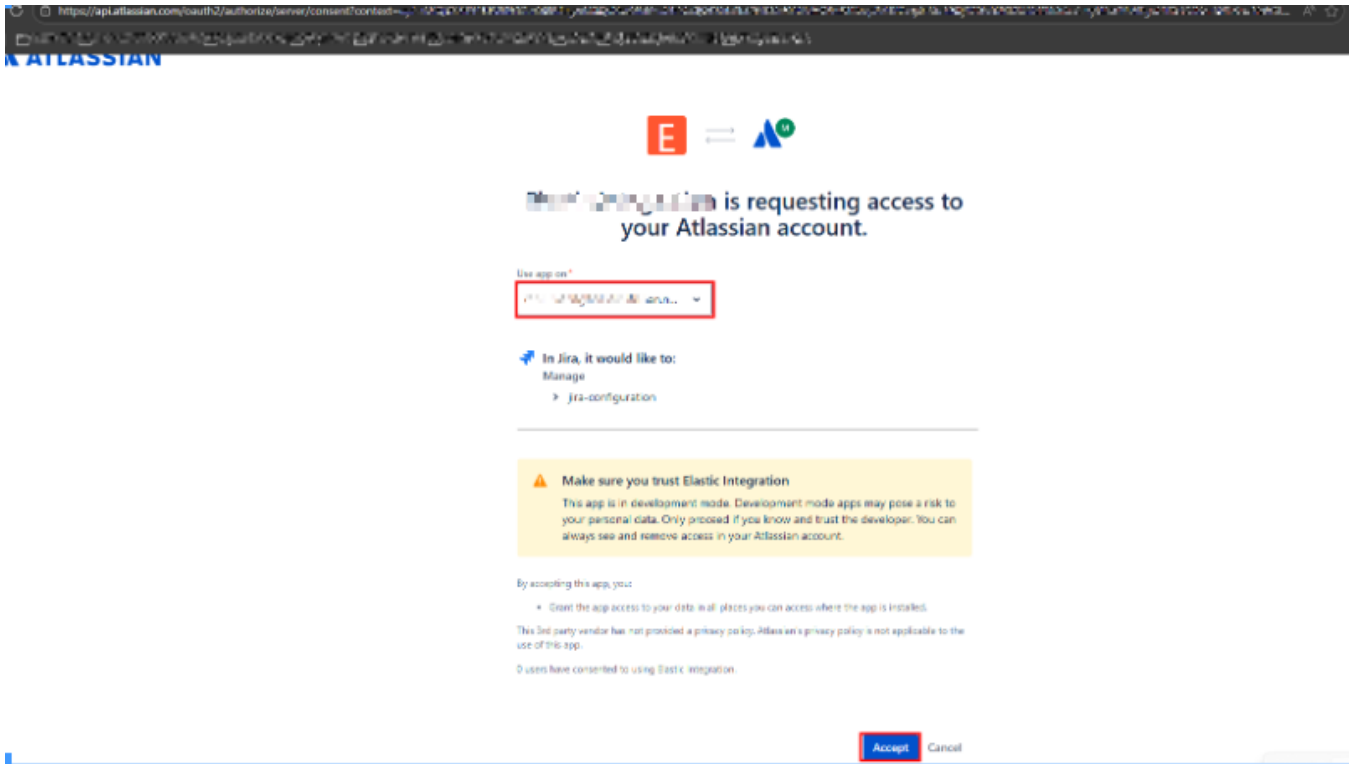
Use this URL to install your 3LO app.

Note: The generated URLs use the first redirect URI from your saved list. When making OAuth requests, ensure you use the appropriate redirect URI that matches your application flow.

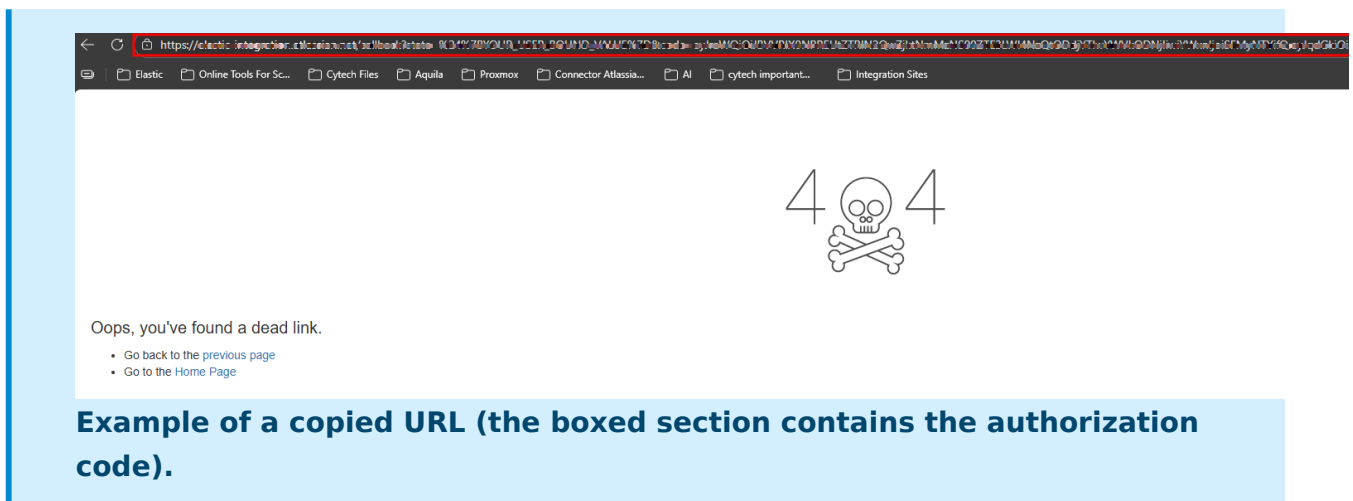
- The Jira API for your app doesn't have any granular scopes. Add granular scopes to your app.
- The Jira Service Management API for your app doesn't have any classic scopes. Add classic scopes to your app.

Classic Jira platform REST API authorization URL

https://auth.atlassian.com/authorize?audience=api.atlassian.com&client\_id=Ga4...&response\_type=code&prompt=consent&state=\${YOUR\_USER\_BOUND\_VALUE}&scope=manage%3Ajira-configuration&redirect\_uri=https%3A%2F%2Felastic-integration.atlassian.net%2Fcallback



10. Copy the URL and paste in text editor (notepad).



## Exchange authorization code for access token

Paste this Curl command into terminal.

```
curl --request POST \  
  --url 'https://auth.atlassian.com/oauth/token' \  
  --header 'Content-Type: application/json' \  
  --data '{"grant_type": "authorization_code", "client_id": "YOUR_CLIENT_ID", "client_secret":  
"YOUR_CLIENT_SECRET", "code": "YOUR_AUTHORIZATION_CODE", "redirect_uri":
```

```
"https://YOUR_APP_CALLBACK_URL"}'
```

Change all fields:

- `client_id`: (*required*) Set this to the **Client ID** for your app. Find this in **Settings** for your app in the [developer console](#).
- `client_secret`: (*required*) Set this to the **Secret** for your app. Find this in **Settings** for your app in the [developer console](#).
- `code`: (*required*) Set this to the authorization code received from the initial authorize call (described above).
- `redirect_uri`: (*required*) Set this to the callback URL configured for your app in the [developer console](#).

**If successful, this call returns an access token similar to this:**

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": <string>,
  "expires_in": <expiry time of access_token in second>,
  "scope": <string>
}
```

## Make calls to the API using the access token Get the `cloudid` for your site

Your app now has an access token that it can use to authorize requests to the APIs for the Atlassian site. To make requests, do the following:

1. **Get the `cloudid` for your site.**
2. Construct the request URL using the `cloudid`.
3. Call the API, using the access token and request URL.

## Get the `cloud_id` for your site

Make a GET request to <https://api.atlassian.com/oauth/token/accessible-resources> passing the access token as a bearer token in the header of the request. For example:

**Replace "ACCESS\_TOKEN" with your newly generated token.**

```
curl --request GET \  
  --url https://api.atlassian.com/oauth/token/accessible-resources \  
  --header 'Authorization: Bearer ACCESS_TOKEN' \  
  --header 'Accept: application/json'
```

This will retrieve the sites that have scopes granted by the token (see [Check site access for the app](#) below for details). Find your site in the response and copy the `id`. This is the `cloud_id` for your site.

### sample output: a Atlassian Confluence site:

```
[  
  {  
    "id": "1324a887-45db-1bf4-1e99-ef0ff456d421",  
    "name": "Site name",  
    "url": "https://your-domain.atlassian.net",  
    "scopes": [  
      "write:confluence-content",  
      "read:confluence-content.all",  
      "manage:confluence-configuration"  
    ],  
    "avatarUrl": "https://site-admin-avatar-cdn.prod.public.atl-  
paas.net/avatars/240/flag.png"  
  }  
]
```

## Creating the Bash Wrapper Script:

**The wrapper script manages event collection, logging, and ensures only one instance runs at a time.**

### 1. Create required directories for data, logs:

Create a directory named `confluence` (for this example):

```
mkdir confluence
```

### 2. Create the wrapper script file:

```
sudo nano /usr/local/bin/confluence_audit.sh
```

### 3. Add the following content to the file:

Replace the file path `FLAT_FILE` as well as `CLOUD_ID`, `USER_EMAIL`, and `API_KEY`, with their actual values.

**Don't change "flattened.json" file name so that the script will work properly.**

```
#!/bin/bash

set -euo pipefail

# -----
# Config
# -----
FLAT_FILE="/home/<testing-confluence>/confluence/flattened.json"
LOCK_FILE="/var/lock/confluence-events.lock"
CLOUD_ID=" "
USER_EMAIL=" "
API_KEY=" "

# Ensure only one instance runs at a time
exec 200>"$LOCK_FILE"
flock -n 200 || exit 1

# -----
# Call Atlassian Confluence API
# -----
RESPONSE=$(curl -s --request GET \
  --url "https://api.atlassian.com/ex/confluence/$CLOUD_ID/wiki/rest/api/audit" \
  --user "$USER_EMAIL:$API_KEY" \
  --header "Accept: application/json")

# -----
# Validate JSON
# -----
if ! echo "$RESPONSE" | jq . >/dev/null 2>&1; then
  echo "Error: API did not return valid JSON"
```

```
    exit 1
fi

# -----
# Flatten directly to file
# -----
echo "$RESPONSE" | jq -c '
  (.results // [])[]
' > "$FLAT_FILE"

# -----
# Count records
# -----
count=$(echo "$RESPONSE" | jq '(.results // []) | length')

echo "Flattened $count results into $FLAT_FILE"
```

#### 4. Make the script executable:

```
sudo chmod +x /usr/local/bin/confluence_audit.sh
```

#### 5. Set ownership to the dedicated user:

Replace the "user:group" in this example both `<testing-confluence:testing-confluence>`.

```
sudo chown testing-confluence:testing-confluence /usr/local/bin/confluence_audit.sh
```

## Creating the Systemd Service File:

The systemd service ensures the event collection runs continuously and restarts automatically on failure.

#### 1. Create the service file:

```
sudo nano /etc/systemd/system/confluence-audit.service
```

#### 2. Add the following content to the file:

## Replace WorkingDirectory file path.

```
[Unit]
Description=Atlassian Confluence Audit Log Fetcher
After=network.target
Wants=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/confluence_audit.sh

# Run as your user
User=testing-confluence
WorkingDirectory=/home/testing-confluence/confluence

# Logging
StandardOutput=journal
StandardError=journal

# Security (optional but recommended)
NoNewPrivileges=true

[Install]
WantedBy=multi-user.target
```

## Create a Systemd Timer to handle looping and run automatically in the background:

A systemd timer is a feature of systemd used to schedule tasks to run automatically at specific times or intervals.

### 1. Create the timer file:

```
sudo nano /etc/systemd/system/confluence-audit.timer
```

### 2. Add the following content to the file:

```
[Unit]
Description=Run Atlassian Confluence Audit every 10 minutes

[Timer]
OnBootSec=60
OnUnitActiveSec=600
Persistent=true
Unit=confluence-audit.service

[Install]
WantedBy=timers.target
```

## Configuring Log Rotation

To manage log file sizes and prevent disk space issues, configure log rotation for Atlassian Confluence Audit Logs.

### Step 1: Create a logrotate configuration file:

```
sudo nano /etc/logrotate.d/confluence_audit
```

### Step 2: Add the following configuration to the file:

**Replace `/home/your_user/` with your actual path, and update `user_owner` and `user_group` to match the correct user and group.**

```
/home/your_user/confluence/flattened.json {
    daily
    rotate 7
    missingok
    notifempty

    copytruncate

    compress
    compressoptions -1
    delaycompress

    dateext
    dateformat -%Y%m%d-%H%M%S
```

```
create 0640 user_owner user_group
}
```

## Enabling and Starting the Service

### Step 1: Reload systemd to recognize the new service file:

```
sudo systemctl daemon-reload
```

### Step 2: Enable the service to start automatically on boot:

```
sudo systemctl enable --now confluence-audit.timer
```

### Step 3: Verify the service is running:

```
sudo systemctl list-timers
```

You should see something like this.

```

NEXT                                LEFT LAST                                PASSED UNIT
Sat 2026-04-11 06:03:57 PST          1min 59s Sat 2026-04-11 05:53:57 PST 8min ago jira-audit.timer
Sat 2026-04-11 06:10:00 PST           8min Sat 2026-04-11 06:00:08 PST 1min 49s ago sysstat-collect.time
Sat 2026-04-11 06:19:54 PST          17min Fri 2026-04-10 06:25:08 PST - apt-daily-upgrade.ti
Sat 2026-04-11 06:30:47 PST          28min Sat 2026-04-11 05:26:41 PST - fwupd-refresh.timer
Sat 2026-04-11 07:00:00 PST          58min Sat 2026-04-11 06:00:08 PST 1min 49s ago logrotate.timer
Sat 2026-04-11 07:31:07 PST          1h 29min Fri 2026-04-10 23:33:13 PST - anacron.timer
Sat 2026-04-11 11:49:02 PST          5h 47min Sat 2026-04-11 01:10:47 PST - apt-daily.timer
Sat 2026-04-11 14:22:01 PST           8h Sat 2026-04-11 03:37:00 PST - motd-news.timer
Sun 2026-04-12 00:00:00 PST          17h Sat 2026-04-11 00:00:00 PST - dpkg-db-backup.timer
Sun 2026-04-12 00:07:00 PST          18h - - sysstat-summary.time
Sun 2026-04-12 00:12:38 PST          18h Sat 2026-04-11 03:37:00 PST - man-db.timer
Sun 2026-04-12 03:10:56 PST          21h Sun 2026-04-05 03:10:35 PST - e2scrub_all.timer
```

## Monitoring Live Logs:

To view real-time logs from the confluence-audit service:

```
journalctl -u confluence-audit -f
```

Please provide the following information to CyTech.

- Flattened.json file path example `"/home/testing-confluence/confluence/flattened.json"`

---

***If you need further assistance, kindly contact our support at [support@cytechint.com](mailto:support@cytechint.com) for prompt assistance and guidance.***

---

Revision #6

Created 19 April 2026 16:41:19 by Benjie Janlay Jr.

Updated 21 April 2026 23:18:51 by Benjie Janlay Jr.