How to use Backlog's migration tool

Move issues and wikis between different spaces or projects within the same space using Backlog's migration tool.

In this guide, we'll show you how to:

- Prepare for migration
- Migrate issues and wikis

Prepare for migration

To start, you'll need:

- Java 8 runtime environment to execute jar files
- Command-line access to execute simple commands
- Administrator access (space or project admin depending on what you're migrating and where)
- The ability to create query parameters using the Backlog Get Issue List API (optional)

Next, you'll need to:

1. **Create API keys** for both the migration origin space and the migration destination space.
2. **Set up a working directory** (referred to as "work") and move the jar file there.

Example commands:

Mac:

```
Unset
$ mkdir work
$ cd work
$ mv download-directory/backlog-migration-[latest version].jar .
```

Windows:

```
Unset
c:\> md work
c:\> cd work
```

```
c:\work> move download-folder\backlog-migration-[latest
version].jar .
```

3. **Configure proxy settings** (if applicable) with the following command:

```
Unset
java -Djdk.http.auth.tunneling.disabledSchemes= \
-Dhttps.proxyHost=[Proxy server hostname/IP address] \
-Dhttps.proxyPort=[Port number] \
-Dhttps.proxyUser=[Proxy username] \
-Dhttps.proxyPassword=[Proxy password] \
-jar backlog-migration-[latest version].jar
```

And finally, **download the migration tool** and verify the installation with the following command:

```
Unset
java -jar backlog-migration-<latest version>.jar
```

Migrate issues and wikis

Before migrating, keep in mind the following limitations:

- Specific project key changes and custom attributes may not migrate correctly.
- API rate limitations can affect migration, especially on Free plans.
- Some features such as stars on issues or wikis are not migrated.

To migrate your content, you need to:

1. **Create a project** in the destination space.
2. **Run the init command** to create a mapping file with the required parameters:
    - --src.key: API key for the origin space
    - --src.url: URL of the origin Backlog space
    - --dst.key: API key for the destination space
    - --dst.url: URL of the destination Backlog space
    - --projectKey: Project key(s) for the origin and destination spaces

```
Unset
java -jar backlog-migration-[latest version].jar init \
--src.key [Source API key] \
--src.url [https://Source-space-id.backlog.jp] \
--dst.key [Destination API key] \
--dst.url [https://Destination-space-id.backlog.jp] \
--projectKey [Source project key:Destination project key]
```

3. **Edit the users.csv file to map users** between the source and destination spaces. For deleted users, add placeholder users in the destination space.
4. **Run the execute command** to import issues and wikis. You can customize the command with:
    - --fitIssueKey: Match issue numbers closely
    - --importOnly: Only import from the previously outputted file
    - --exclude: Choose to exclude issues or wikis
    - --retryCount: Set the number of retries in case of errors

```
Unset
java -jar backlog-migration-[latest version].jar execute \
--src.key [Source API key] \
--src.url [https://Source-space-id.backlog.jp] \
--dst.key [Destination API key] \
--dst.url [https://Destination-space-id.backlog.jp] \
--projectKey [Source project key:Destination project key]
```

5. **Filter issues to migrate** similar to Backlog API v2 query parameters. For example:

```
Unset
--filter "issueTypeId[]=1&issueTypeId[]=2"
```

6. **Review duplicates** found automatically by the migration tool.

Filtering issues to be migrated

You can filter issues from the migration origin project using the `--filter` option. The filtering method is similar to using query parameters with the Backlog API v2 for retrieving issue lists. Note that filtering options do not support “sort”, “order”, “projectId[]”, or custom attributes (e.g., `customField_`). The `--filter` option can only be used with the `execute` command, not the `init` command.

Examples of filtering options:

Option Example	Type	Description
`--filter "issueTypeId[]=1&issueTypeId[]=2"`	Number	Filter by type ID. If you don't know the ID, go to Project Settings > Types in Backlog and hover over the type name to see the ID in the URL.
`--filter "categoryId[]=99&categoryId[]=98"`	Number	Filter by category ID. The category ID can be found by hovering over the category name in Project Settings > Category List in Backlog.
`--filter "id[]=123&id[]=456"`	Number	Filter by issue ID. This is useful for retrying specific issues that failed to migrate.
`--filter "offset=1000"`	Number	Issue offset. Use this to split large migrations into smaller batches. Combine with `--filter count`.
`--filter "count=100"`	Number	Set the maximum number of issues to be migrated. Combine with `--filter offset`.

Duplicate checks during re-migration

To prevent duplication during re-migration, the tool automatically checks:

- If the "User ID who registered the issue," "Issue registration time," and "Issue subject" all match, the issue will not be duplicated.
- If the "Wiki page name" matches, the wiki will not be duplicated.

Log files

The migration tool's directory structure is as follows, with log files stored in the `log` directory:

```
Unset
Working directory/
├ mapping/
│ ├ users.csv
│ └ users_list.csv
└ log/
  ├ backlog-migration-warn.log
  └ backlog-migration.log
```

Limitations

1. If the project key is changed after the parent issue has been changed, the source cannot reflect the changes to the parent issue.
2. Issue types that existed in the past are automatically created during migration. Delete them manually after migration if necessary.
3. Stars attached to issues and wikis are not migrated.
4. Units set in custom attributes of a number are not migrated in the comment's changelog.
5. Projects with the same source and destination cannot be migrated.
6. The `--fitIssueKey` and `--filter` options cannot be used simultaneously.
7. Custom attributes with the same name at the source and destination may cause inconsistencies in migrated data. Check these in advance.
8. Global search indexes are not migrated.
9. States added to a project cannot be migrated from ASP to Enterprise.
10. Free plan API rate limitations prevent migration.
11. Parallel execution of this tool may exceed API rate limits and is not guaranteed to work.
12. Wiki images using display dugs by ID will not display if moved to another space.
13. Projects with more than nine state changes cannot be migrated.
14. Names with leading/trailing spaces or commas in Issue Types, Categories, Status, Versions, Milestones, or Custom Fields may not migrate correctly. Remove these before migration.
15. Files, Subversion, and Git are not migrated.
16. Notification read status is not transferred.

Good to know

When migrating to a lower plan, be aware of plan-specific restrictions such as the number of projects, users, and file size limits. The Backlog Migration Tool cannot migrate beyond these limitations.

Have a question or need help?

If you have any questions or need assistance, please contact Nulab support.