Jira Azure DevOps migrator PRO

Product overview

Welcome to the Jira to Azure DevOps Migrator PRO documentation! This tool is designed to help you easily migrate your Jira issues to Azure DevOps, while providing additional PRO features atop of the community edition:

- concatenate issue fields
- migrate releases and the FixVersion field
- Select any property for object- and array-type fields for mapping

Notice about the Bootstrapper Utility

This offering of **Jira Azure DevOps migrator PRO** includes a handy bootstrapper application for generating config files tailored to your needs. We recommend using this utility to get started as quickly and easily as possible with migrating Jira issues to Azure DevOps.

You can find the tool at `migration-jira-ado/jira-migrator-bootstrapper/`. You will also find documentation for this tool at `migration-jira-ado/jira-migrator-bootstrapper.pdf`.

Getting Started

To get started with the Jira to Azure DevOps Migrator PRO, please refer to the documentation of the **Jira Azure DevOps migrator Community Edition**. The documentation provides step-by-step instructions to help you configure and run the migrator tool on your local machine.

You can find the online documentation here: https://github.com/solidify/jira-azuredevops-migrator/tree/master/docs (https://github.com/solidify/jira-azuredevops-migrator/tree/master/docs)

Specifically, see the following pages:

- Overview: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/overview.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/overview.md)
- How to configure the tool: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/config.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/config.md)
- How to run the **Jira Exporter**: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/jira-export.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/jira-export.md)
- How to run the **Azure DevOps importer**: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/wi-import.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/wi-import.md)
- FAQ: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/faq.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/faq.md)

Once you have read through the open documentation and feel comfortable configuring the tool, proceed with this document to learn about the PRO features.

Usage examples

```
.\jira-export.exe -u alexander.hjelm@solidify.dev -p [Jira API token] --url
https://solidifydemo.atlassian.net/ --config C:\dev\config-agile.json --force --license-file-path
C:\temp\xxx.atlassian.net.2023-01-24.json
```

```
.\wi-import.exe --token [ADO PAT] --url https://dev.azure.com/solidifydemo --config C:\dev\config-
agile.json --force
```

More details:

- Jira export tool, usage: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/jira-export.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/jira-export.md).
- ADO import tool, usage: https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/wi-import.md (https://github.com/solidify/jira-azuredevops-migrator/blob/master/docs/wi-import.md).

PRO Features

1. Consolidating Issue Fields

One of the key features of the Jira to Azure DevOps Migrator PRO is the ability to consolidate issue fields. This allows you to merge multiple fields from your Jira issues into a single field in Azure DevOps based on a configurable text pattern.

To consolidate issue fields, simply add a field mapping with the `MapFieldsComposite` mapper to your `config.json` file, like this:

```
"field-map": {
    "field": [
    ...
        {
        "source": "${summary}-${description}",
        "target": "System.Description",
        "mapper": "MapFieldsComposite"
        },
    ...
    ]
}
```

This example will concatenate the `summary` and `description` fields and output the result to the `System.Description` field in Azure DevOps. Within the `source` field, you may specify any pattern you wish, but any `${}` clause will be substituted by the value of the corresponding field.

Notice about missing fields in the MapFieldsComposite source pattern

If you have source fields in your pattern which are missing data in your Jira Issues, the data will migrate, but you may experience missing fields in the pattern. Alternatively, you can set the **ignore-mapping-composite-fields-if-missing-values** boolean property in your **config.json** to True to ignore migrating composite patterns with missing field data. Then, if any of the fields (demarcated by ${}) in the pattern are missing from the source issue, no data will be migrated at all for that particular issue and field. This will affect all MapFieldsComposite field maps in your configuration.

2. Migrating Releases

Another feature of the Jira to Azure DevOps Migrator PRO is the ability to migrate releases. This feature allows you to migrate all the information related to a release, including associated issues and the `FixVersion` field. To migrate releases, you will need to make some modifications to the ADO WIT Process of the target project, as well as some modifications to the config file.

Note: if you used the **jira-migrator-bootstrapper** utility to generate your config file, this file will already be pre-filled with everything you need to migrate releases, as long as you set `ExportReleases true`.

Follow the steps as outlined below:

1. Create the `Release` issue type (or the equivalent issue type with the name of your choice) in the ADO WIT Process of the target project.
2. Add the `export-releases`, `source-project-key` and `releases-folder` properties to your `config.json`. This will let the jira exporter know that you wish to export releases, from which project and where on the disk to put them:

   ```
   "export-releases": true,
   "releases-folder": "Releases",
   "source-project-key": "AGILEDEMO",
   ```

3. Add the `JiraRelease` type to the `type-map` in your `config.json`. This will let the jira exporter know what Work Item Type the releases should be created as in Azure DevOps. The following sample will create the releases as the work item type `Release` (you may need to edit the Azure DevOps process model and create the desired work item type first):

   ```
   "type-map": {
       "type": [
       {
           "source": "JiraRelease",
           "target": "Release"
       },
       ...
       ]
   },
   ```

4. Add a status mapping for the `JiraRelease` type to the `field-map` in your `config.json`. This will let the jira exporter know how to map the possible states that the releases can be in. Here is an example:

```
"field-map": {
    "field": [
        {
            "source": "status",
            "target": "System.State",
            "for": "JiraRelease",
            "mapping": {
            "values": [
                {
                "source": "Released",
                "target": "1"
                },
                {
                "source": "Unreleased",
                "target": "2"
                },
                {
                "source": "Archived",
                "target": "3"
                },
                {
                "source": "Overdue",
                "target": "4"
                }
            ]
            }
        },
    ]
}
```

5. Ensure that your config file has a field mapping entry for the `fixVersions` field, for example:

```
{
    "source": "fixVersions",
    "source-type": "id",
    "target": "Custom.Fixversions"
},
```

3. Selecting any property for object- and array-type fields

In order to select a certain property, create a **field map** in your configuration and use the `custom-key` field to select a JSON property.

You can either select a top-level field directly or select a field at a path by providing the full path with a dot (.) separated notation.

This feature works for both Object-type fields (map the value directly) or for arrays (using the MapArray mapper).

Specifying the property to map for Object-type fields

Example scenario: We want to extract the `emailAddress` property of the below custom field and use it in our mapper:

```
"customfield_10012": {
    "self": "https://jira.myorg.com/rest/api/2/user?username=databaseadmins",
    "name": "databaseadmins",
    "key": "databaseadmins",
    "emailAddress": "DatabaseServices@myorg.com",
    "avatarUrls": {
        "48x48": "https://jira.myorg.com/secure/useravatar?avatarId=10232",
        "24x24": "https://jira.myorg.com/secure/useravatar?size=small&avatarId=10232",
        "16x16": "https://jira.myorg.com/secure/useravatar?size=xsmall&avatarId=10232",
        "32x32": "https://jira.myorg.com/secure/useravatar?size=medium&avatarId=10232"
    },
    "displayName": "Database Admins",
    "active": true,
    "timeZone": "America/Los_Angeles"
},
```

Solution: create the following field map:

```
{
"source": "customfield_10037",
"target": "Custom.DatabaseAdmins",
"custom-key": "emailAddress"
},
```

Specifying the property to map for Array-type fields

Example scenario: We want to extract the `id` properties of the below custom field and use it in a **MapArray** mapper:

```
"customfield_10037": [
  {
    "self": "https://solidifydemo.atlassian.net/rest/api/2/customFieldOption/10010",
    "value": "All",
    "id": "10010"
  },
  {
    "self": "https://solidifydemo.atlassian.net/rest/api/2/customFieldOption/10011",
    "value": "None",
    "id": "10011"
  }
],
```

Solution: create the following field map:

```
{
  "source": "customfield_10037",
  "target": "Custom.MultiSelect",
  "mapper": "MapArray",
  "custom-key": "id"
},
```

4. Correct embedded links to Jira Issues in text fields

The **Jira Azure DevOps Migrator PRO** tool contains an additional utility for correcting hyperlinks to other Jira Issues in your source project inside html text fields, such as Description, Repro Steps and comments. The updated fields will contain links that point to the correct Work Item in Azure DevOps instead of JIRA.

In order to enable this feature, you will need to do some edits to the Work Item form in Azure DevOps. For each work item type, you must add the field `Custom.Reference` (or however you choose to name the field). The purpose of this field is to hold the `key` of the source issue.

Now, in your `config.json` file, add the following flag (Replace `Custom.Reference` with the field name you chose in the previous step):

```
"jira-key-reference-field": "Custom.Reference",
```

Finally, in your `config.json` file, add a field mapping for the `key field` (Again, replace `Custom.Reference` with the field name you chose in the previous step):

```
{
  "source": "key",
  "target": "Custom.Reference"
}
```

Now the `wi-import.exe` tool is correctly configured, and will automatically update any hyperlinks to other Jira issues in your source project!

5. Migrate RemoteLinks/WebLinks to Hyperlinks

Remote Links (Web links) are automatically migrated with the Jira AzureDevOps Migrator PRO. No additional configuration is needed.

6. Migrate Sprint dates

Sprint dates are automatically migrated with the Jira AzureDevOps Migrator PRO.

In your `config.json` file, add a field mapping for the `Sprint`, like this:

```
{
  "source": "Sprint",
  "source-type": "name",
  "target": "System.IterationPath",
  "mapper": "MapSprint"
}
```

No further configuration is needed.

7. Support for state transition dates for workflows with custom states.

In the Jira Azure DevOps Migrator PRO, the following ADO fields are supported for WIT processes with custom Proposed states and Completed states:

- ActivatedDate
- ClosedDate
- ActivatedBy
- ClosedBy

By default, these fields are only supported for the states **New**, **Closed** and **Done**, across all work item types.

This feature is enabled by default, and no further configuration is needed.

8. Migrate the Work Log field (Time Spent, Remaining Estimate)

```
{
  "source": "timetracking",
  "target": "Custom.RemainingEstimateHours",
  "custom-key": "remainingEstimate"
},
{
  "source": "timetracking",
  "target": "Custom.TimeSpentHours",
  "custom-key": "timeSpent"
},
{
  "source": "timetracking",
  "target": "Custom.RemainingEstimateSeconds",
  "custom-key": "remainingEstimateSeconds"
},
{
  "source": "timetracking",
  "target": "Custom.TimeSpentSeconds",
  "custom-key": "timeSpentSeconds"
}
```