The Guide for AzDO Migration

Last updated by | Danny Garber | Feb 23, 2020 at 12:40 AM EST

Azure DevOps Migration Tools

The Azure DevOps Migration Tools allow you to bulk edit and migrate data between AzDO Projects on both Microsoft Team Foundation Server (TFS) and Azure DevOps Services. Take a look at the documentation to find out how. This project is published as code on GitHub as well as a Azure DevOps Migration Tools on Chocolatey.

What can you do with this tool?

- Migrate Work Items from one AzDO Project to another AzDO project and Sync changes after a migration
- Merge many AzDO Projects into a single AzDO Project
- Split one AzDO Project into many AzDO Projects
- Assistance in changing Process Templates
- Bulk edit of Work Items
- Migration of Test Suits & Test Plans

Quick Links

- Getting Started
- Documentation

The Migration Guidance for Kroger DASH AzDO project

In order to use this tool to migrate the existing DASH AzDO project to another AzDO project in other organization (or the same), the following steps should be taken.

Install and Configuration

1. Install the Azure DevOps Migration Tool following the links in the above section.
2. Create a new empty AzDO project with Kanban inheritance project type, which is inherited from Agile type.
3. Create or obtain the Personal Access Token (PAT) for both, the source and destination projects.

> Note: if both projects belong to the same organization, it's sufficient to have just one PAT for both source and destination projects.

4. Create a custom field `MigrationSourceWorkItemId` in the AzDO organization level following this instructions
5. Update the `configuration-workitems.json` file (you can find it here) with your project's PAT values and the custom field names. For example, if we want to migrate from `DASH` project to `DASH2` under the `csedevs` organization, your configuration sections for `Source` and `Target` projects may look like this:

```
    "Source": {
        "Collection": "https://dev.azure.com/csedevs/",
        "Project": "Dash",
        "ReflectedWorkItemIDFieldName": "Custom.MigrationSourceWorkItemId",
        "AllowCrossProjectLinking": false,
        "PersonalAccessToken": "<insert here your PAT>"
    },
    "Target": {
        "Collection": "https://dev.azure.com/csedevs/",
        "Project": "Dash2",
        "ReflectedWorkItemIDFieldName": "Custom.MigrationSourceWorkItemId",
        "AllowCrossProjectLinking": false,
        "PersonalAccessToken": "<insert here your PAT>"
    },
```

Project Iterations Migration

First, we need to copy all existing project iterations (e.g. Milestone 1, Milestone 2, ...) into a target project. In order to do that,

1. Open `configuration-workitems.json` in your preferrable editor. Find the `Processors` configuration section, and then look for the object type of `NodeStructuresMigrationConfig` .
2. Under this section, update `Enable` property to `true` , while making sure all other processors are disabled (their `Enable` property should be set to `false`). Here's a sample excrept of this processor's section:

```
{
  "ObjectType": "VstsSyncMigrator.Engine.Configuration.Processing.NodeStructuresMigrationConfig",
  "Enabled": true,
  "PrefixProjectToNodes": false,
  "BasePaths": []
},
```

3. Navigate to the folder where you installed the DevOps Migration tools (e.g. `c:\tools\VSTSSyncMigration`), and from the command line run this command:

```
migration.exe execute -c <path to your configuration file>
```

After the tools finishes running, you may see on your screen something similar to this screenshot:

```
migration.exe Information: 0 :  Access granted
--CreateNode: \Dash2\Area\Dash Administrators...found
--CreateNode: \Dash2\Iteration\Sprint 0, start date: 9/3/2019 12:00:00 AM, finish date: 10/25/2019 12:00:00 AM
--CreateNode: \Dash2\Iteration\OneWeek Training (MSFT Out), start date: 11/4/2019 12:00:00 AM, finish date: 11
--CreateNode: \Dash2\Iteration\Thanksgiving Holidays, start date: 11/25/2019 12:00:00 AM, finish date: 11/29/2
--CreateNode: \Dash2\Iteration\Winter Holiday Break 1, start date: 12/23/2019 12:00:00 AM, finish date: 12/27/
--CreateNode: \Dash2\Iteration\Milestone 1, start date: 10/13/2019 12:00:00 AM, finish date: 11/30/2019 12:00:
--CreateNode: \Dash2\Iteration\Milestone 2, start date: 12/9/2019 12:00:00 AM, finish date: 1/24/2020 12:00:00
--CreateNode: \Dash2\Iteration\Milestone 3, start date: 1/26/2020 12:00:00 AM, finish date: 2/28/2020 12:00:00
--CreateNode: \Dash2\Iteration\Winter Holiday Break 2, start date: 12/30/2019 12:00:00 AM, finish date: 1/3/20
--CreateNode: \Dash2\Iteration\Milestone 2 (Sprint 0), start date: 12/2/2019 12:00:00 AM, finish date: 12/6/20
--CreateNode: \Dash2\Iteration\Roadmap, start date: 3/2/2020 12:00:00 AM, finish date: 12/31/2020 12:00:00 AM.
migration.exe Information: 0 :  Migration Context Complete NodeStructuresMigrationContext
[Info]: Run complete...
[Info]: ------------------------------END-----------------------------
[Info]: Duration: 00:00:03.1967217
[Info]: End Time: 2/22/2020 11:34:11 PM
```

Now, continue to the next steps.

Project Iterations Migration

After the iteration pathes have been migrated, it's time to migrate all the existing teams (e.g. Dash Administrators).

1. Open `configuration-workitems.json` in your preferrable editor. Find the `Processors` configuration section, and then look for the object type of `TeamMigrationConfig` .
2. Under this section, update `Enable` property to `true` , while making sure all other processors are disabled (their `Enable` property should be set to `false`). Here's a sample excrept of this processor's section:

```
{
  "ObjectType": "VstsSyncMigrator.Engine.Configuration.Processing.TeamMigrationConfig",
  "Enabled": true,
  "EnableTeamSettingsMigration": true,
  "PrefixProjectToNodes": false
},
```

3. Navigate to the folder where you installed the DevOps Migration tools (e.g. `c:\tools\VSTSSyncMigration`), and from the command line run this command:

```
migration.exe execute -c <path to your configuration file>
```

You should see the completion of this step at the end of the migration tool run.

And now, please continue to the next steps.

Work Items Query Migration

This step describes how to migration the existing Queries.

1. Open `configuration-workitems.json` in your preferrable editor. Find the `Processors` configuration section, and then look for the object type of `WorkItemQueryMigrationConfig` .

2. Under this section, update `Enable` property to `true`, while making sure all other processors are disabled (their `Enable` property should be set to `false`). Here's a sample excrept of this processor's section:

```
{
  "ObjectType": "VstsSyncMigrator.Engine.Configuration.Processing.WorkItemQueryMigrationConfig",
  "Enabled": true,
  "EnableTeamSettingsMigration": true,
  "PrefixProjectToNodes": false,
  "SharedFolderName": "Shared Queries"
},
```

> Note: if your queries stored in a different shared folder in your source project, then update the `SharedFolderName` property value with the correct Queries folder location.

3. Navigate to the folder where you installed the DevOps Migration tools (e.g. `c:\tools\VSTSSyncMigration`), and from the command line run this command:

```
migration.exe execute -c <path to your configuration file>
```

You should see the completion of this step at the end of the migration tool run.

And now, please continue to the next steps.

Git Repo Migration

Before we run the migration tools to copy all existing Work Items, we should first migrate all existing Git repos from the source project in order to allow the migration tool to restore all existing links to the repo items.
To copy all Git repos from one project to another, we'll use the `git commands`.

As an example, we will show here how to migrate a single repo `dash-dmp` from the source project `Dash` to the target project `Dash2`. You should repeat these steps for all Git repos you want to migrate to your target project.

1. Clone the existing repo into your local machine:

   ```
   git clone --mirror https://csedevs@dev.azure.com/csedevs/Dash/_git/dash-dmp
   ```

   After the clone command finishes, you should see a new folder created in the same location from where you run the `git` command named `dash-dmp.git`. Navigate into it.

2. Before we push the cloned repo into our target project, we must first create a new empty repo with the same name in that project. So, navigate to your target project in the Browser, and go to the Repos. Then choose `+ New repository` from the menu and enter the name for your new repo, e.g. `dash-dmp`.

3. Now, we can push this repo into the target project.

   ```
   git push --mirror https://csedevs@dev.azure.com/csedevs/Dash2/_git/dash-dmp
   ```

   You can now navigate directly into this new repo in your Azure AzDO project in browser to see that master, and all branches with their history have been successfully migrated to your target project.

Repeat 1-3 steps for all the remaining Git repos.

> Note: in some cases for some repos, for example, when outstanding PRs are still being reviewed, the `git --mirror push` command may fail. As workaround, that repo can be migrated via AzDO `Import repository` menu item from the online AzDO interface.

Work Items Migration

This step describes how to migration all existing work items such as user stories, epics, features, bugs, HTML and image attachments and more.

1. Open `configuration-workitems.json` in your preferrable editor. Find the `Processors` configuration section, and then look for the object type of `WorkItemMigrationConfig` .
2. Under this section, update `Enable` property to `true` , while making sure all other processors are disabled (their `Enable` property should be set to `false`). Here's a sample excrept of this processor's section:

```
{
  "ObjectType": "VstsSyncMigrator.Engine.Configuration.Processing.WorkItemMigrationConfig",
  "Enabled": false,
  "ReplayRevisions": true,
  "PrefixProjectToNodes": false,
  "UpdateCreatedDate": true,
  "UpdateCreatedBy": true,
  "UpdateSourceReflectedId": false,
  "BuildFieldTable": true,
  "AppendMigrationToolSignatureFooter": false,
  "QueryBit": "",
  "OrderBit": "[System.ChangedDate] desc",
  "LinkMigration": true,
  "AttachmentMigration": true,
  "AttachmentWorkingPath": "c:\\temp\\WorkItemAttachmentWorkingFolder\\",
  "FixHtmlAttachmentLinks": true,
  "SkipToFinalRevisedWorkItemType": false,
  "WorkItemCreateRetryLimit": 5,
  "FilterWorkItemsThatAlreadyExistInTarget": false,
  "PauseAfterEachWorkItem": false,
  "AttachmentMazSize": 480000000
}
```

> Note: if you want to filter migrating the work items that match specific criteria, you may use the `QueryBit` property to insert the `WHERE` clause to filter your source items. For example, in order to include all items only from the `Milestone 3` and `RoadMap` iterations, modify `QueryBit` like this:
> `"QueryBit": "AND [System.IterationPath] IN ('Dash\\Milestone 3', 'Dash\\Roadmap')",`

3. Navigate to the folder where you installed the DevOps Migration tools (e.g. `c:\tools\VSTSSyncMigration`), and from the command line run this command:

```
migration.exe execute -c <path to your configuration file>
```

Depending on the total number of migrating items, this command may take from several minutes to several hours.
So, it's always a good time to take a break and go and have some coffee while this migration tools does its job.

You should see the completion of this step at the end of the migration tool run.

What is remaining?

Unfortunately, due to the lack of automated tool to migrate some of the components of your AzDO project, the following remaining items of the project must be migrated manually:

- Build pipelines
- Variable Groups
- Artifacts