An IBM Document from

IBM Z DevOps Acceleration Program

January, 2022 - Release 1.0

Migrating from IBM Engineering Workflow Management (EWM) to GIT

Gil Parent
gparent@fr.ibm.com

Abstract

How to migrate application source files from EWM to Git.

Table of Contents

1 Introduction

A migration project can be a very challenging and time-consuming task. It can also cause grief in the future if some files are corrupted. Because these problems could potentially be discovered months or years after the migration, it could become a nightmare to correct them. Therefore, the migration must be secured to get the assurance that all source files are correctly transferred and if not, we should at least have a report listing of inconsistent files.

Developing a set of "one off" scripts and procedures for a one-time migration is time consuming when there is already an existing set of scripts and procedures.

This document proposes a recipe to address this scenario.

2 Which tools do we have ?

The Dependency Based Build (DBB) product proposes a migration process to move files from PDSEs to a local Git workspace. This migration process will check for the non-roundtripable characters (more information at https://www.ibm.com/docs/en/adfz/dbb/1.0.0?topic=migrating-data-sets-git) and for the non-printable characters you may have in your files in PDSEs. It will also create the .gitattributes file containing all the necessary code page translations. In GIT, source code is stored in the UTF-8 format.

EWM can move files from the EWM server to the mainframe PDSEs by using the z/OS Dependency Based Build functionality. The default code page in EWM is also UTF-8, but files can also be stored with other code pages. This information is attached to the file as a property so when EWM moves the files from EWM server to mainframe PDSEs, it will be able to take care of the code page translation for each file.

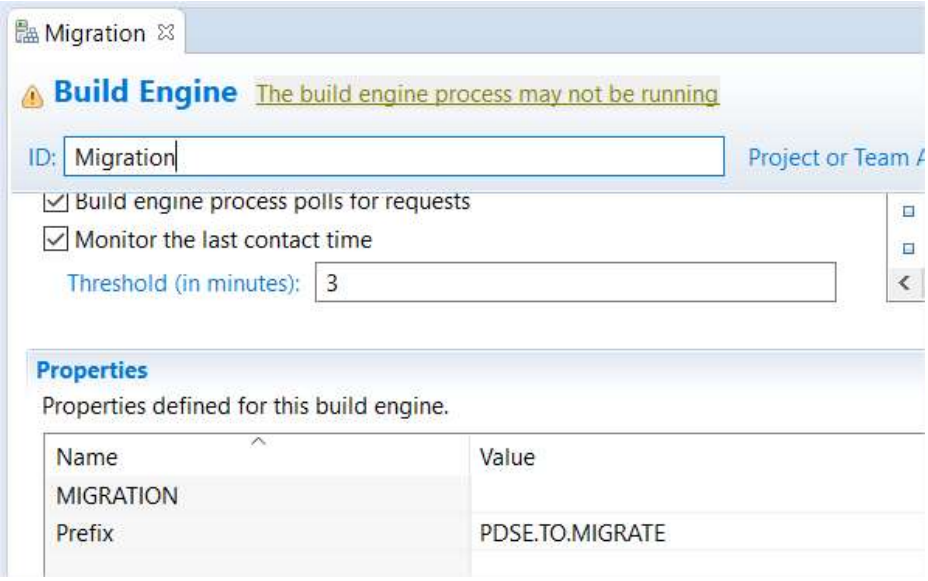
Another issue you may encounter is even though all the files in EWM have the code page UTF-8, the migration may not correctly check for hexadecimal characters (i.e., non-roundtripable/non-printable characters) in the source files, which will result in file corruption.

If you are sure all the files in EWM are code page UTF-8 and none of the files have non-roundtripable characters and/or non-printable characters, then you can continue to section 2.4 of this document.

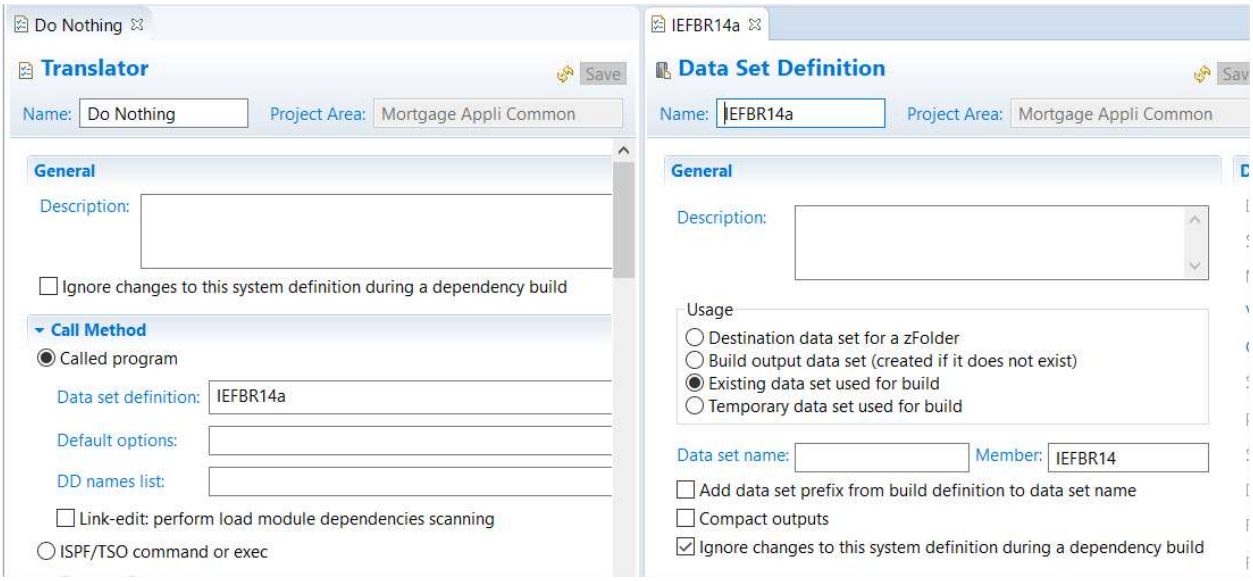
The idea is to leverage the EWM builds for the file transfer from the EWM SCM to the target mainframe PDSEs without performing the typical build steps like compiling, link editing, binding and any other custom steps that are added to the builds. Following this approach, the members are stored with the correct codepage on the mainframe.

2.1 Adaptation in EWM

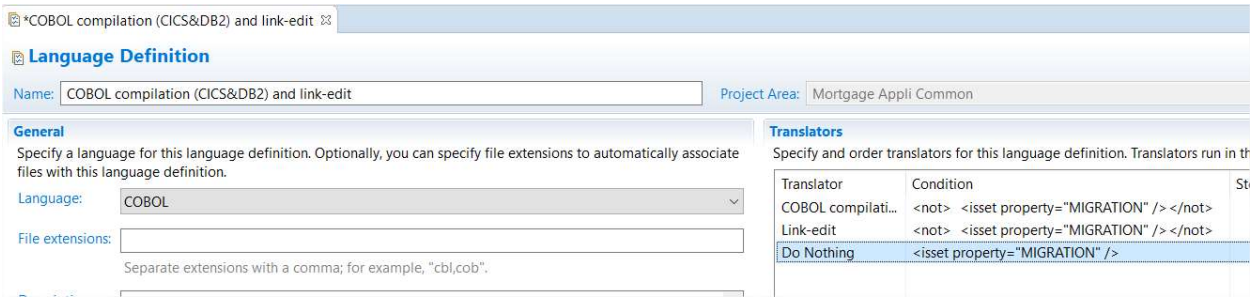
To keep the existing EWM builds working for the developers, the idea is to add a new translator in EWM and to add conditions to all translators for the necessary language definitions. To accomplish this, a new property named MIGRATION can be added to the build engines or, even better, a new build engine dedicated to the migration with this property can be created.



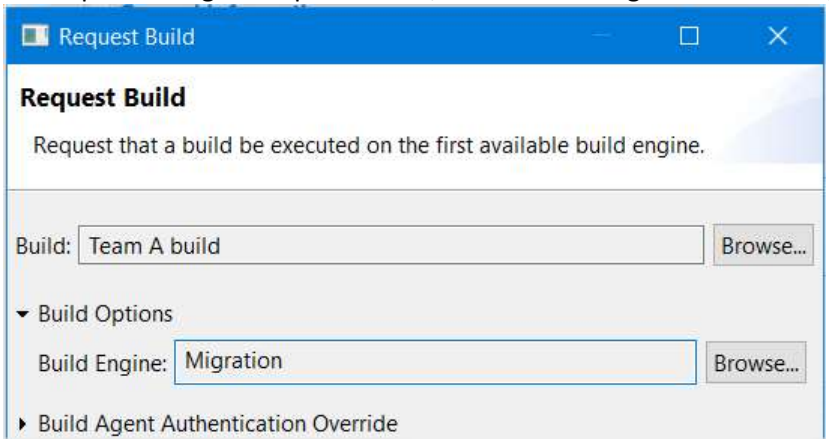
After that, you can create a translator “Do Nothing” as follows :



Next, modify the language definition by adding the new translator and associated conditions.



When performing a “Request Build”, choose the “Migration” build engine.

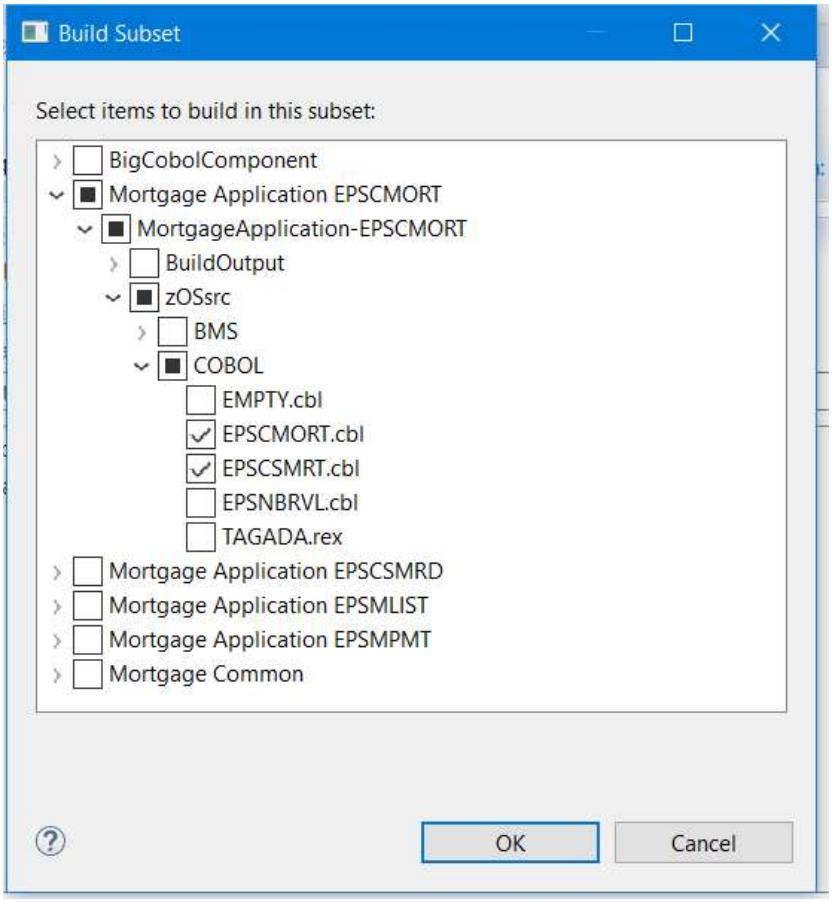


Because the MIGRATION property was set-up in the build engine, only the translator “Do Nothing” will be executed and only the files are uploaded to the configuration build libraries (PDSEs)

To be more selective on the list of files to process, you can use the subset artifact to select which files should be built.

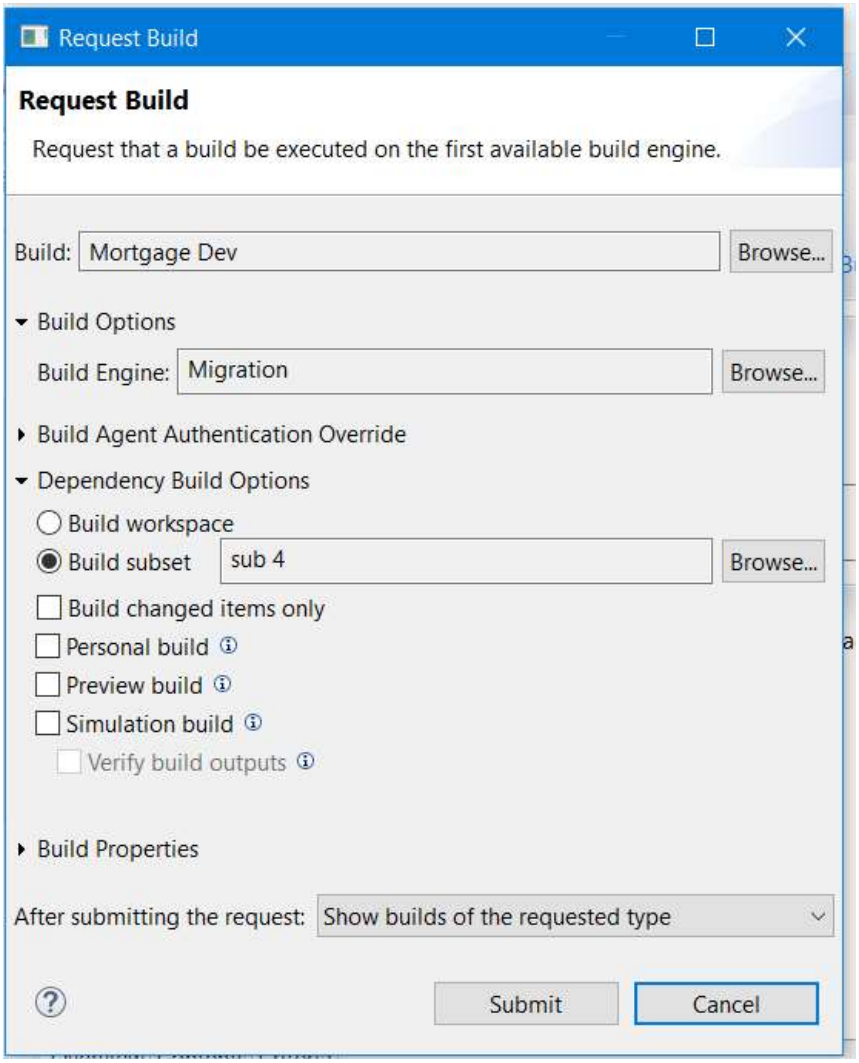
Create a subset. In the “Contents” tab, you have several ways to select the files you want :

- all the file of the stream
- all the files of components
- all files of folders
- file by file.



To force the build to process all the selected files, when you request a build, choose the subset and uncheck the “build changed items only”.

The build will run and move the selected files to the targeted PDSEs.



2.2 How to use the DBB migration script

A DBB migration script is shipped with the DBB toolkit and is used to copy members from PDSEs to the local Git workspace in USS.

This script works with a mapping file in which you list all the files to copy. Please have a read here: https://www.ibm.com/docs/en/adfz/dbb/1.1.0?topic=migrating-data-sets-git

The migration script is located in the migration subdirectory of your DBB toolkit installation, in /usr/lpp/dbb/v1r0/migration

The script will check for specific characters in hexadecimal (i.e., non-roundtripable/non-printable characters) which can cause issues after the migration when you try to build the files from Git using the DBB scripts. However, it is expected that you have addressed that already when you migrated to EWM SCM.

A very interesting solution is to combine the use of rules and the preview option, which will generate the mapping files for you. More information can be found on the DBB migration website https://www.ibm.com/docs/en/adfz/dbb/1.1.0?topic=migrating-data-sets-git. After that, you can run the script again with the mapping file to copy the files from the PDSEs to the folders of your local Git workspace in USS.

After the files have been migrated to USS, use the Git commands to ADD and then COMMIT and PUSH the files to the Git server.

2.3 How to migrate several versions of the files.

The organization in EWM is probably made of several streams representing different versions of your applications. Several versions of the files must be migrated, and you will start with the oldest version.

The process described above can be used by migrating the same files from the different streams to the same Git repositories. This will create a history of the changes.

If you want more history, you can also use the snapshots done in EWM to get more versions from EWM.

2.4 Other solutions

- ### Zload

You can use the EWM scm command zload (find more information at https://www.ibm.com/docs/en/elm/7.0.2?topic=reference-zload) to move files from EWM to the mainframe and then use the DBB migration script. With this solution you will be able to select a list of files after which you will use the DBB migration script.

The zLoad command just loads the files, so you are going to lose all RTC file properties. This is something that needs to be considered.

- ### Files copy/paste

After having loaded the local EWM workspace, use the Windows copy/paste function to create the files in the local Git workspace and then use the Git Commit command to push the files to the Git Server.

Potential issues:

- How to check the non-printable and the non-roundtripable characters contained in the files.
- UTF-8 is the default code page in EWM, but files can be stored with another code page. In Git, the code page is UTF-8 so you will have potential issue with non-translatable characters. This will probably not occur in Git but instead in MVS when these files are transferred from Git to MVS.
- If you want to keep track of the changes made in EWM then you will have to redo the same process for every version that you want to store in Git.

- ### Solution proposed on the web

Another solution is proposed on the web site https://jazz.net/forum/questions/261294/rtc-scm-to-git-migration. This solution is just mentioned here but has not been evaluated.

3 Conclusion

Several solutions are available to migrate applications from EWM to Git. They all have pros and cons and will depend on how you first migrate and/or work with EWM.

A pain point could be if you are using the file's properties to store compilation options, linkedit options or any other values used during the build in EWM. In Git, there is no feature of having properties attached to a file so you will have to extract and then store these properties in a specific file which will be used as an input file during the build process in Git.