How to Test the Confluence V2 Mappings

For the connectathon, there is a complete installation of Eclipse and TBX on a USB drive, which you can copy onto your machine, and so avoid the jobs of installing, setting up an Eclipse workspace, etc.

This version of TBX already has some projects which test the mapping tables on Confluence, but you can set up and run your own test by going through the following steps, with help from me.

The outline steps are:

* Use a V2 example message from the Confluence examples page or the Github repository.
* Work out which V2 segments and FHIR resources are involved
* Import the required segment mappings from the Confluence tables into TBX
* Apply the mappings to the V2 example, to create a FHIR bundle
* This bundle will be incomplete because of limitations in the import process and the mappings.
* Edit the FHIR bundle to make it more complete; generate and run a transform
* Inspect the resulting mappings, transform in FHIR mapping language, etc.

This helps to show where the V2 mappings are incomplete, and to make them more complete; and to generate transforms from the mappings.

More detailed steps follow.

The V2\_TBX folder has these sub-folders:



* **'Confluence Mappings'** contains XML extracts from the Confluence segment mapping tables
* **'eclipse'** holds the eclipse icon used to run TBX
* **'Extraction Program'** contains the Java program used to make XML downloads of the Confluence mapping tables (you will not need this)
* **'MWB V2 definitions'** contains definitions of the V2 structure of the examples, made with the Messaging Workbench, and used by TBX
* **'V2 Examples'** contains the V2 examples from the Confluence examples page
* **'V2\_workspace'** is the eclipse workspace used by TBX.

To create a new test of the mapping tables, proceed as follows:

1. Choose a V2 example message containing the segments whose segment mapping tables you want to test.
2. Work out what FHIR resources are needed in the bundle resulting from the transform.
3. In the eclipse folder, use the eclipse icon  to start TBX.
4. You will see a number of TBX projects testing the mappings. To create a new test:
5. Create a new Mapper project (**File > New > Other > Mapper Project** )
6. Copy the V2 example message into the 'Instances' folder of that project
7. Copy an empty FHIR bundle from the 'Instances' folder of the project 'FHIR\_R4' to the Instances folder of the new project
8. Copy the MWB structure definition of the V2 message into the 'Structures' folder of the project
9. Copy the folder of Confluence Mappings into the 'Mappings' folder of the project
10. In the Structures folder, by editing an exisiting '.prof' file, make a '.prof' file defining the FHIR resources needed in the bundle. This is a simple text file, looking like (the first row defines where FHIR resource definitions are found):

project FHIR\_R4

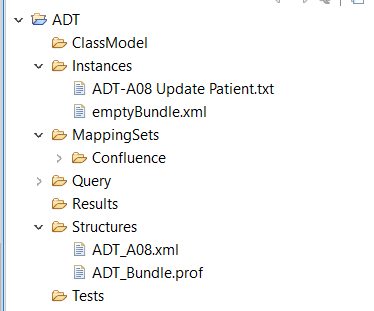
core Bundle

core MessageHeader

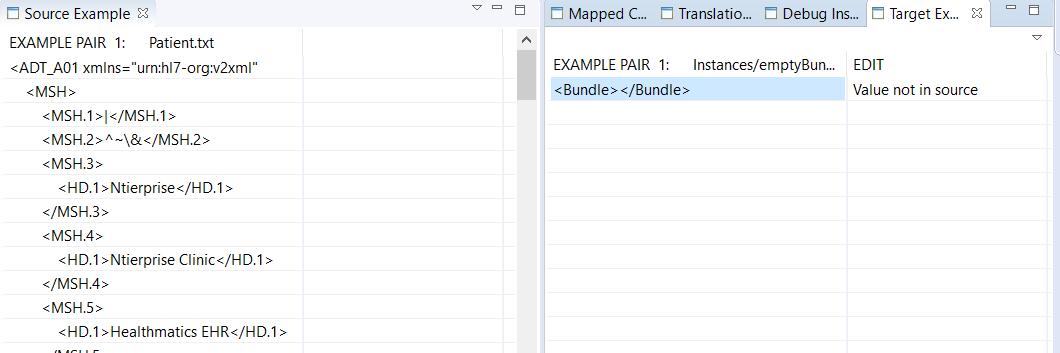
core Patient

core RelatedPerson

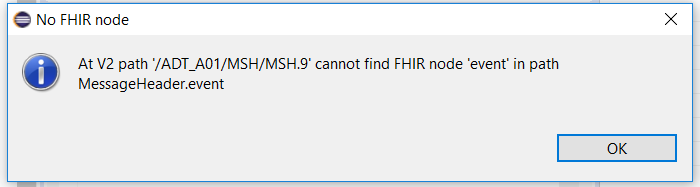
1. Now the folders in the new project should look approximately like:



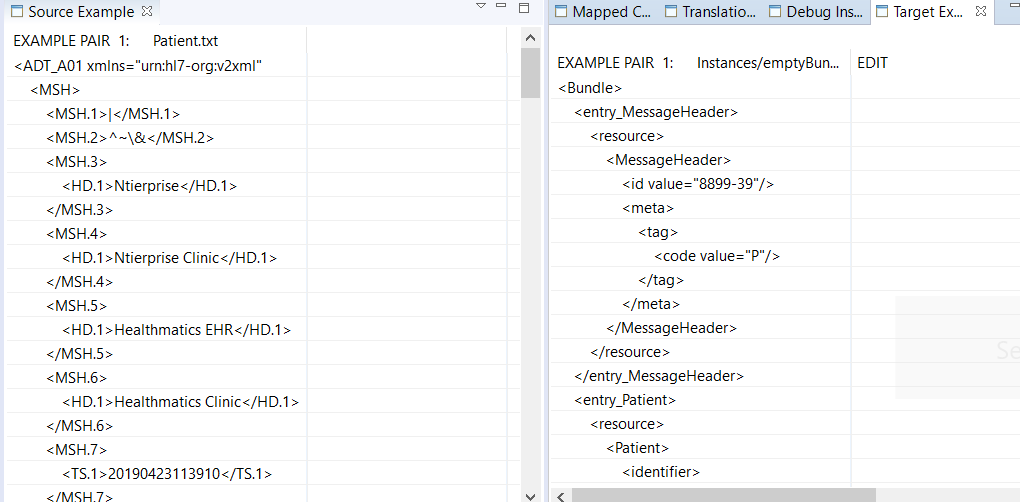
1. In the Source Editor window, choose the dropdown menu item 'New Transform'.
2. This will ask you to make 5 file choices - selecting the 4 files shown above, and defining one new file:
   1. for the source structure, choose the MWB-created XML file defining the V2 message
   2. for the target structure, choose the .prof file defining the FHIR bundle
   3. For the source instance, choose the V2 example message
   4. For the target instance, choose the empty FHIR bundle
   5. choose a name and location to save a '.gen' file, which defines the transform (holding the choices you just made)
3. You should then see the source and target examples in the source editor view and the target editor view:



1. To import the Confluence mappings, choose the dropdown menu item 'Define Tabular Mappings' in the source editor view
2. This will ask you to choose xml mappings files, imported from the Confluence tables, until you choose cancel (after you have chosen all the segment mappings you want to use)
3. You next apply these mappings, to extend the target example. Choose the dropdown menu item 'Apply Tabular Mappings' in the target editor view.
4. This will give you a series of warning messages to do with the mappings (because they are currently quite limited), such as



1. Just keep clicking 'OK' to these warnings. You can get rid of them next time round, by removing the offending lines from the XML extracts of the Confluence mappings.
2. Then TBX applies all the mappings it can apply, to add structure to the target example, using data values taken from the V2 source example:



1. These additions to the target instance are quite limited, because we cannot yet capture all the mappings from the Confluence tables (and there are no profiles or extensions in the target)
2. You can use the target editor to add to the target bundle the extra fields which you think should be there. This is easier than using a general editor.
3. The target editor only allows you to add structures which are allowed by the FHIR resource definitions.
4. Add a few of these, then save all changes.(and it is best to make a backup copy of the FHIR target - because if you happen to do 'Apply Tabular Mappings' again by mistake, it will overwrite your changes)
5. If you have added parts to the target which TBX cannot work out how to derive from the source, it will warn you. These typically come from:
   1. fixed values in the target
   2. code conversions from source to target
   3. data format conversions from source to target
6. There are ways to fix all these issues, described in the TBX Help files.
7. If you choose the target editor menu item 'Generate Transform' , TBX will generate a V2 to FHIR transform and tell you of any problems it comes across when doing so.
8. When generating a transform, you can save it as HAPI Java code or as FHIR Mapping Language - see the TBX help files for how to do this.
9. You can test the resulting transform by the menu option **Test > Native Transform**. This is currently a bit slow, but shows you any gaps in the transform result, compared to the example pair you have built.