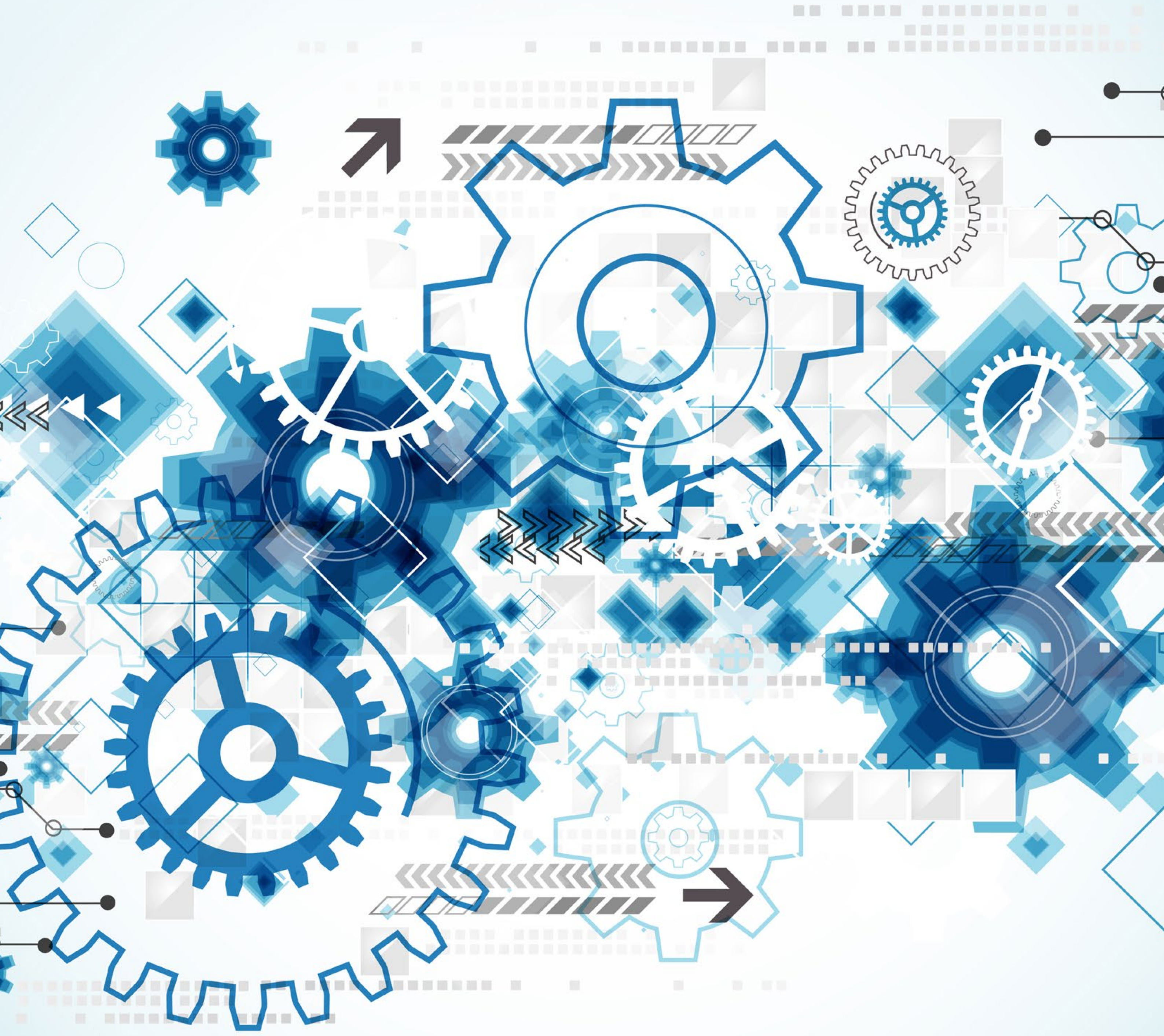
ULTIMATE SOFTWARE

WHITEPAPER

MICROSOFT DYNAMICS 365 FOR OPERATIONS

EDT MIGRATION TOOL



Contents

Introduction to Microsoft Dynamics 365 EDT migration tool

At Ultimate Software we help our customers to bring and keep their processes under control with Microsoft Dynamics 365 – in production, trade, maritime, or service companies. As we are experts in Microsoft Dynamics 365, we are always looking for the most suitable solutions for many different challenges. The moment we found out that there were no solutions for automatically refactoring EDT table relations into table references, we created an EDT migration tool for Microsoft Dynamics 365 for Operations. This tool will duplicate the functionality of the similar AX 2012 tool provided by Microsoft, but works with the updated, XML-based file-format of Microsoft Dynamics 365 for Operations.

Background

In Dynamics AX it is considered best practice to use Extended Data Types (EDTs) for every column of every table in the database, instead of basic datatypes like string or integer. This practice makes the code easier to read, the variables easier to understand and reduces work overall during development.
In AX 2009 and prior versions, the Extended Data Types (EDTs) had relations with other tables to mark that, any time this EDT was used on a child table, an implicit relationship was defined between the child table and the table defined in the Relation.
In AX 2012 this practice was no longer recommended, and instead, defining Table Relations was now considered the best practice. Microsoft also supplied an EDT migration tool that automatically created Table Relations instead of EDT relations.
This AX 2012 tool also introduced Table References on the EDTs, which are very similar in functionality to the EDT Relations.

The EDT migration tool

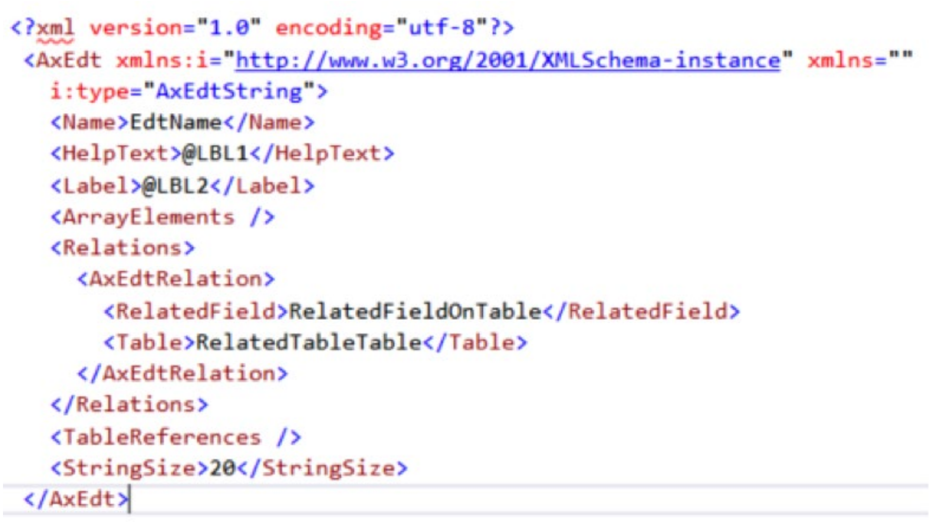
In Microsoft Dynamics 365 for Operations (formerly known as AX7) the EDT migration tool is not available anymore.
However if a team skipped running in the AX 2012 version, or is upgrading from AX 2009 to Operations 365 with as few steps as possible, there might still be EDTs in the source code with relations.
This simple tool helps with refactoring the still-existing EDT relations into Table References on the EDTs and Table Relations onto the tables where the EDTs are used. The source code of the tool is available on GitHub, in this repository: https://github.com/balog-gergely/EdtMigrator.

Limitations and licensing

This EDT Migration Tool is published under the Microsoft Public License (https://msdn.microsoft.com/en-us/library/ff647676.aspx), under the copyright of Ultimate Software B.V., The Netherlands (http://www.ultimatesoftware.nl/).
The tool is as-is. It is not feature-complete; we took several shortcuts while writing it. There are several rare, complex scenarios that we choose to log instead of covering every possibility in code. It migrated over 90% of our 700+ EDTs successfully, leaving only the remaining 10% for manual migration.
The tool will not migrate an EDT if it has overlays, has fixed-field relations or array elements.

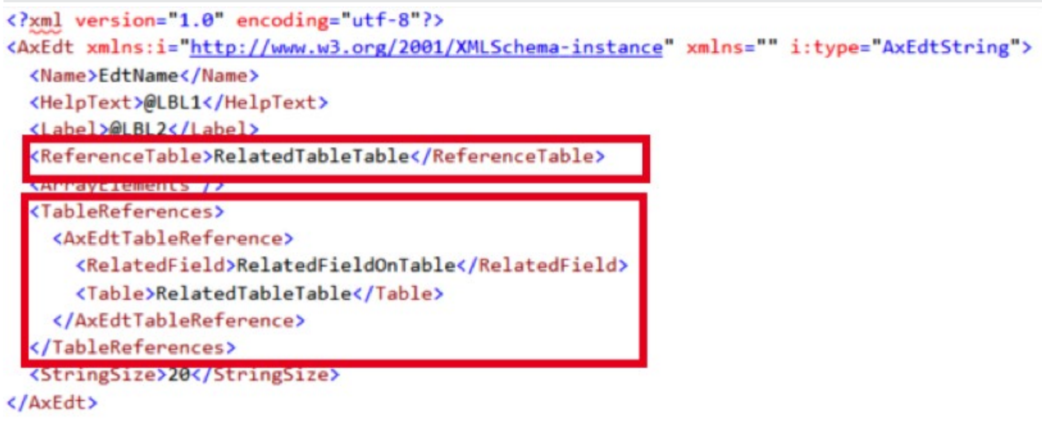
XML structure of EDTs

The following is an example of an EDT with an EDT relation (pre migration). It is referencing the RelatedTable. RelatedFieldOnTable.



```
<?xml version="1.0" encoding="utf-8"?>
 <AxEdt xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns=""
   i:type="AxEdtString">
   <Name>EdtName</Name>
   <HelpText>@LBL1</HelpText>
   <Label>@LBL2</Label>
   <ArrayElements />
   <Relations>
     <AxEdtRelation>
       <RelatedField>RelatedFieldOnTable</RelatedField>
       <Table>RelatedTableTable</Table>
     </AxEdtRelation>
   </Relations>
   <TableReferences />
   <StringSize>20</StringSize>
 </AxEdt>
```

Two things need to be changed for this to be up to the current standards: the ReferenceTable node needs to be filled, and the contents of the AxEdtRelations need to be moved to the TableReferences node.



```
<?xml version="1.0" encoding="utf-8"?>
<AxEdt xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="" i:type="AxEdtString">
  <Name>EdtName</Name>
  <HelpText>@LBL1</HelpText>
  <Label>@LBL2</Label>
  <ReferenceTable>RelatedTableTable</ReferenceTable>
  <ArrayElements />
  <TableReferences>
    <AxEdtTableReference>
      <RelatedField>RelatedFieldOnTable</RelatedField>
      <Table>RelatedTableTable</Table>
    </AxEdtTableReference>
  </TableReferences>
  <StringSize>20</StringSize>
</AxEdt>
```

There are also EDTs with multiple relations and EDTs with fixed field relations, making a further restriction on the contents of the field, for example only allowing the use of CargoCodes where the CargoType==1 on the record. Fixed field relations became Filters on the Table Reference node of the EDT. Fixed field EDT relations also need to be duplicated onto the tables themselves as fixed field table relations.
However in our project there were so few EDTs with fixed field relations that we didn't automate this part of the migration, opting instead to log such occurrences and deal with them manually.

There can also be inheritance relations between EDTs, for example the SquareId can extend the ShapeId fictional EDT. In this case the relations were usually stored on the parent in the inheritance hierarchy (in our case the ShapeId), so we must move up on the inheritance tree until we find an EDT that has table relations or table references defined, or we reach a root node in the tree. In the latter case, we can be certain that this EDT does not refer to a table.

Because our custom EDTs can extend baseline data types, we also have to check baseline directories. Naturally, we don't want to update these EDTs, only read them. There are also EDTs with Array Elements and this tool also skips those.

XML structure of tables

During migration we try to faithfully recreate the relations on the tables where the EDT was used. For example if there is a CargoTypeId EDT that references the TypeId field of the fictional CargoType table, then on every child table containing a field with the CargoTypeId EDT we will need to create a foreign key relation to the CargoType.CargoTypeId field.

```
<AxTableRelation xmlns="" i:type="AxTableRelationForeignKey">
  <Name>CargoType</Name>
  <RelatedTable>CargoType</RelatedTable>
  <Constraints>
    <AxTableRelationConstraint xmlns="" i:type="AxTableRelationConstraintField">
      <Name>CargoTypeId</Name>
      <Field>CargoTypeId</Field>
      <RelatedField>TypeId</RelatedField>
    </AxTableRelationConstraint>
  </Constraints>
</AxTableRelation>
```

If the same EDT appears multiple times on a table, then multiple foreign keys need to be created. In that case we must fill the Role and Related Table Role fields of the AxTableRelation to avoid compile errors. When creating these table relations several types of conflicts can occur. The tool logs all of these potential problems for later manual resolution.

- Foreign keys might already exist that have additional filters. For example it is possible that there is already a foreign key relation on the CargoTypeId field that also has an extra constraint on the CargoCategory field. In this case the migration tool will mark this relation for manual checking.
- EDTs that have fixed field relations or multiple table relations are not migrated by the tool.
- It is possible that there are already multiple pre-existing foreign keys on the table all using a single field as their source. The tool reports this as an anomaly.

Tool setup

Before running the tool the following variables have to be set in the code:

- MetadataFolder: the root of your AOS directory
- TargetModels: the list of models that you want the tool to process, either because they contain EDTs or they contain tables using those EDTs. The names are interpreted as subfolders under the MetadataFolder, for example "ApplicationSuite\ApplicationSuite.Contoso"
- ReadOnlyModels: the list of models that can contain parent EDTs. It is safe to list all baseline models here.

Tool output

The tool creates a folder next to the executable that contains all the updated xml files in exactly the same folder structure as the original. The contents of this folder can simply be copied to the MetadataFolder to overwrite the existing state before checking in the changes into source control. Several log files are also created.
The following items need attention after the migration:

- MultiRelationEdt.txt contains the EDTs that had multiple relations. All of these must be manually processed, and the relations manually created on the tables using these EDTs.
- AnomalyDetected.txt contains pre-existing foreign keys that don't completely align with the table relation on the EDT. This is usually because there is an extra field in the foreign key that we didn't expect. These items have to be manually checked one-by-one, making a case-by-case decision whether to change the relation or not.
- EdtsWithMultiRelationParents.txt contains the EDTs that have parents that we are not equipped to read/parse.
- ParentProblems.txt contains child EDTs for which we could not find the parents.
- TableReferenceWithoutRelatedNodes.txt contains the EDTs that we encounter which have the Table Reference property filled out, but do not have any Table Relations defined.

The following are only reports:

- PrimaryIdDetected.txt is a list of the cases where a field on a table was actually the field referenced by the EDT relation, so no foreign key had to be created. No further action is necessary on these items.
- ForeignKeyCreated.txt contains the list of all table or table extension foreign keys that the tool created.
- ExistingForeignKeyDetected.txt contains the list of foreign keys that are already perfectly compatible with the migrated EDT relation. No further action is necessary on these items.

In Conclusion

After developing this migration tool for O365, we, at Ultimate Software, felt that it was an excellent tool to share with you, as it is entirely implementation independent and solves a problem that currently has no solutions.
If you would like to develop this tool further, contributions are welcome to the repository (https://github.com/balog-gergely/EdtMigrator)

Please follow our company web page for news and inspiration.

Ultimate Software
Maanlander 33
3824 MN Amersfoort
The Netherlands
Tel: +31 (0)88 - 42 42 424

Author: Gergely Balog
gergely.balog@UltimateSoftware.nl

www.ultimatesoftware.nl
www.ultimateshippingsolutions.com
sales@ultimatesoftware.nl