Metadata Management Tutorial

Data Mapping Specification Using Meta Integration® Metadata Management (MIMM)

TABLE OF CONTENTS

TABLE OF FIGURES

1 Introduction

The need for more sophisticated and precise metadata management is a growing concern for most large organizations. Nearly all components that comprise modern information technology, from CASE tools, ETL engines, Warehouses, BI, EAI environments, as well as metadata repositories, contain, and often derive their processing from, metadata. The metadata for these environments is distributed and duplicated, often times active, and generally represented in a variety of methodologies, depending upon the underlying technology they represent.

Meta Integration® Metadata Management (MIMM), provide strikingly expansive set of capabilities in many facets of metadata management, including:

- Business Glossary
- Data Governance
- Metadata comparison, integration, and mapping
- Version and configuration management
- Data life cycle related metadata management
- Lineage and impact analysis
- Enterprise architecture development, management and deployment.
- Data documentation
- Data Mapping Specifications, design and forward-engineering

This document is the culmination of more than fifteen years of experience in supporting the enterprise metadata management and integration requirements of numerous clients. It presents in detail and with supporting tutorials metadata management process and methods, best practices, as well as, many strategic scenarios that leverage the Meta Integration® Metadata Management (MIMM) suite. These examples are comprehensive and directed at real-world examples tied to business-oriented goals and return on investment. In all, anyone who completes the exercises in this document should find it straight-forward to implement and deploy an effective and comprehensive metadata management environment.

Disclaimer

Some of the features detailed in this document may not apply and/or be available for the particular Meta Integration® Metadata Management (MIMM) edition you may have.

1.1 How to use this document

It is certainly possible to skip through the tutorials, and thus simply glean an "management-level understanding" of Meta Integration® Metadata Management (MIMM) and its use within a metadata management environment. However, it is not recommended that one try to skip parts of the tutorials and then try to go through later parts. When following through the tutorial sections, it is very important to respect the order of the steps (and the order sections/labs within each section). The results of preceding tutorials are re-used and built upon in each successive lesson.

In addition, it is important to ensure complete understanding of the conceptual background provided in the sections leading up to and supporting the tutorial material. Thus, one should not simply jump into the tutorials with carefully reviewing the concepts presented in that section.

As this document include hand-on tutorials, a great deal of specificity is required. This detail includes specifying particular CASE, ETL, BI, etc., vendor's tools. While Meta Integration® Metadata Management (MIMM) environment itself is capable of working with over 100 different versions of third-party tools (see http://www.metaintegration.net/Products/MIMB/SupportedTools.html), it is necessary for the clarity conciseness of the tutorials to limit the cadre of tools that will be referred to. Please note that it is not necessary to have these tools on-hand to get the full benefit of the tutorials. Remember also, though you may intend to use Meta Integration® Metadata Management (MIMM) suite of tools with many of the supported third-party tools not specified in the tutorial, it is still quite valuable to learn the processes, methods and best practices presented here. Then one may reuse what one has learned and apply that knowledge and skill to the particular set of tools that are critical to one's own enterprise.

1.2 Conventions used in the tutorial

The following font conventions will be used throughout the tutorial.

- User Interface item – New
- Submenu item – New › Folder
- Terminology item – *model content* item
- Name or label reference – Accounts Payable

2 Data Mapping Specifications

Some data flow processes are not harvestable using the bridges provided as a part of the Meta Integration® Model Bridge (MIMB) tool suite. Of course, if these processed are not modeled in Meta Integration® Metadata Management (MIMM), it will leave gaps in the lineage and impact analysis answers and provide an incomplete picture of the physical architecture of your systems.

In order to address these gaps and produce proper lineage and impact results, Meta Integration® Metadata Management (MIMM) has a Data Mapping Specifications editing and management toolset. Data Mapping Specifications are essentially simply high-level logical (or notional) definitions of the way data "flows" from some number of source models into elements of a target model. These mappings are specified using a simple web based drag and drop type mapping specification editor and are defined using descriptive text and one may also define pseudo operations using an operation editor.

In this chapter we will learn how to take advantage of these tools and capabilities.

2.1 Constructing a data mapping specification

The Data Mapping Specifications form a content within Meta Integration® Metadata Management (MIMM) which defines logical mappings between source and target data stores. The data stores themselves are modeled separately as model contents within Meta Integration® Metadata Management (MIMM) but external to the Data Mapping Specifications, as shown in this diagram:

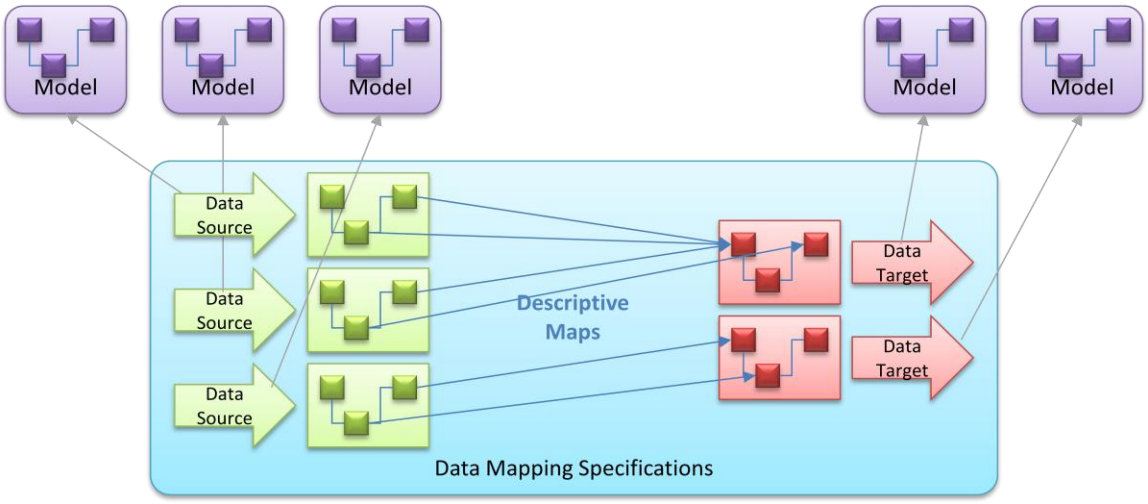

Figure 1 - Mapping Specification Construction

Each data store model content is referenced as either a source, a target, or can be both a source and target. One may then drag and drop objects (e.g., schemas, tables or columns) in a data source onto objects in a data target, thus creating *maps*. These maps may then be documented in *descriptions* or with more explicit *operations*.

2.2 Simple Load

Quite often, the gap in the lineage for metadata analysis is simply a direct load of data from one datastore to another. In this case, it is fairly straight-forward to create a data mapping specification to represent this data movement in Meta Integration® Metadata Management (MIMM), thus providing full lineage reporting through that mapping.

Let's get started.

We have a new data mart which we wish to add to the system. Let's import the model.

Navigate to the repository path:
…1 - Data Sources - Data Stores/2 - Data Warehouse/Vendor Mart

Create a new model named Vendor Mart. It is an ERwin 9.6 Data Modeler model. Import the model already provided at:

…\Models\MetadataManagement\Finance\CaErwin9Xml\VendorMart.xml

Note, we have a nicely modeled data store:

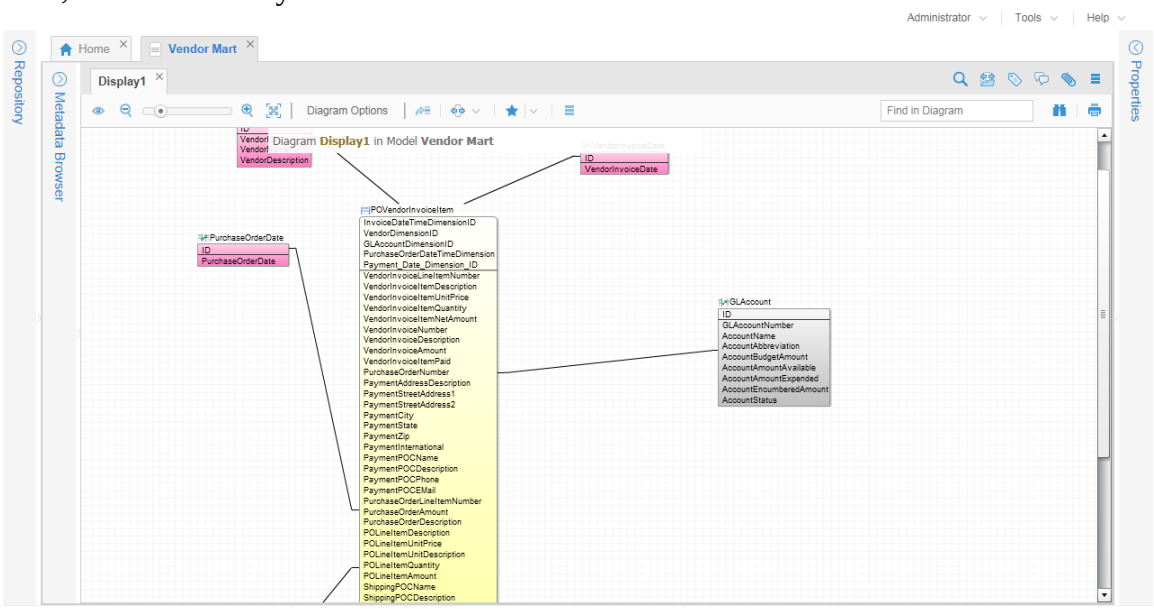

Figure 2 - Diagram in Adjustments model

This model looks like a close match to the Dimensional DW model. In fact, according to the designers it should be, as they say it is a simple move (using some javascript based move scripts) from the Dimensional DW database. So, let's see if we are lucky and simple schema level map will work.
Navigate to the Repository path:
… /Finance System/2 - Data Warehouse
And right-click on the [folder] and select New➔ Data Mapping Specifications:

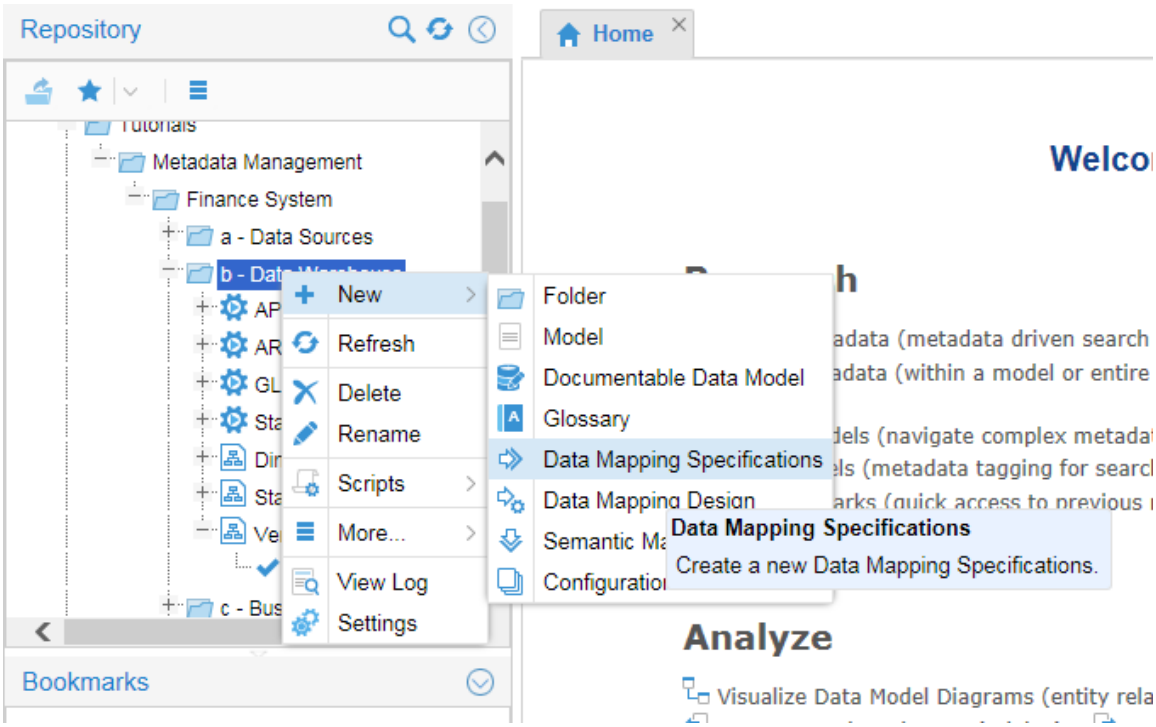

Figure 3 - Create new Data Mapping Specifications

Name the Data Mapping Specifications as "Dimensional DW to Vendor Mart" and click on the Create button. You now have a data mapping specification opened in the work panel. We begin by dragging the source model (Dimensional DW) into the source side of the Data Mapping Specifications:

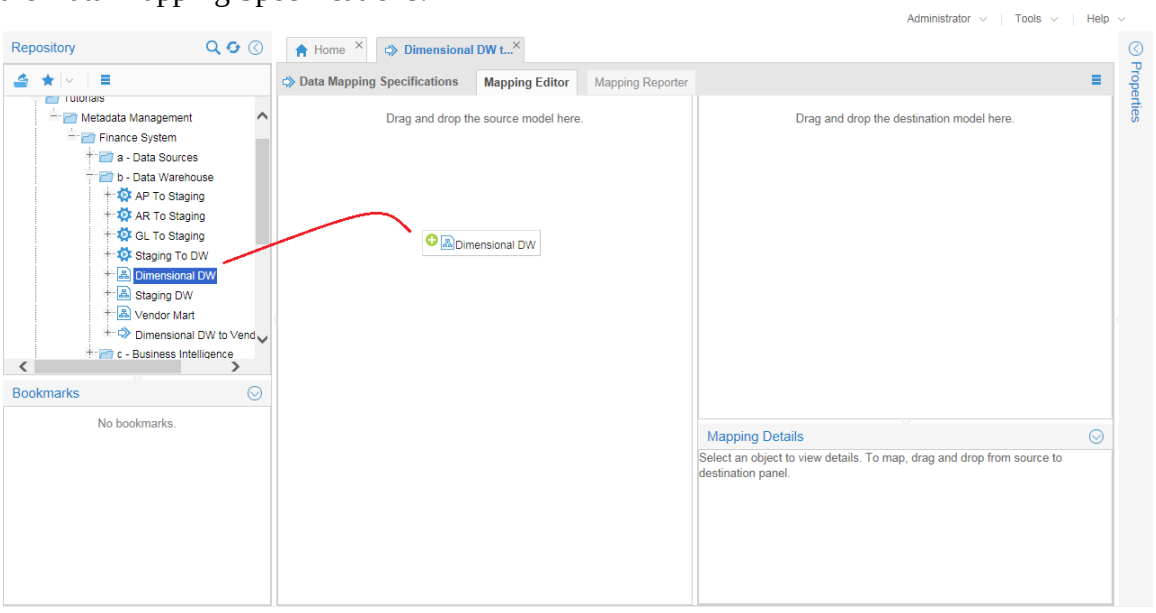

Figure 4 - Drag source into data mapping specification

Drag the Vendor Mart model on to the target panel:

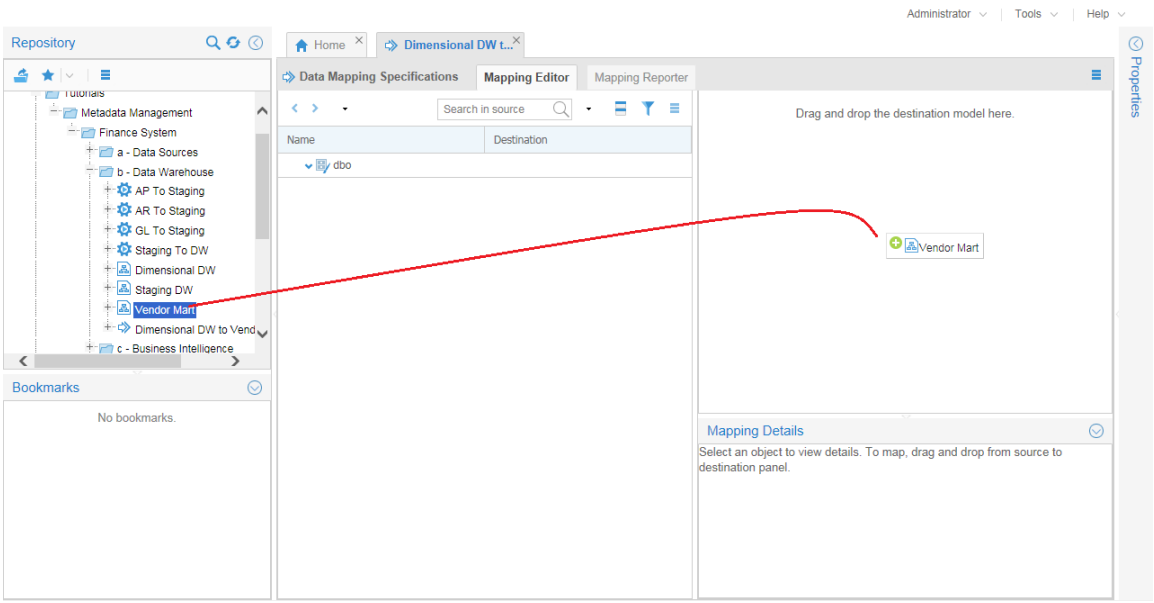

Figure 5 - Adding target to mapping

Now drag the dbo schema in the source panel into the _Default Schema_ schema in the target panel. Now, expand the _Default Schema_ and note that all the target tables have mappings:

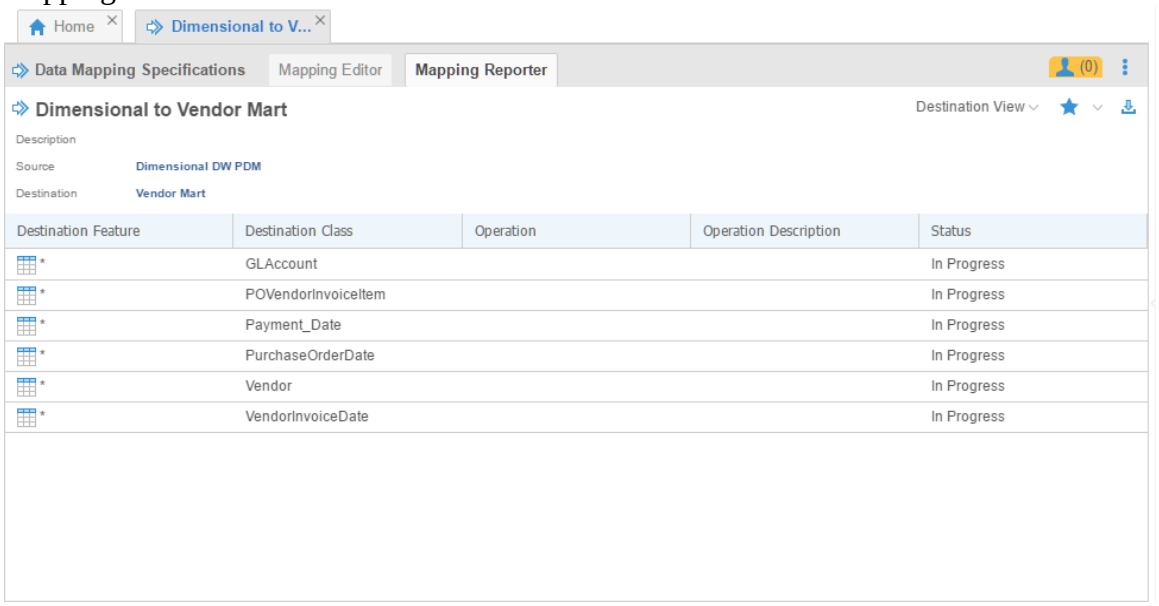

Figure 6 - Target tables mapped

Now, expand the dbo schema in the source panel and click on the GLAccount table in the target panel. Note, the GLAccount table in the source panel is outlined in Green. This is because it sources from the same named table in the source model.

Now, expand the GLAccount table in the source and target panels, and click on the GLAccountNumber column in the target panel. Again, note that the mapping is defined at the column level as well.

So, the mapping is complete with one motion.

Let's add the model and mapping to our configuration. Drag the Vendor Mart model into the Warehouse [folder] in the Finance configuration. Also include the new mapping (Dimensional DW to Vendor Mart) in the configuration.

Build the configuration. Now go to the Architecture Diagram tab and edit the layout of the diagram to move the mapping specification and new model as shown here:

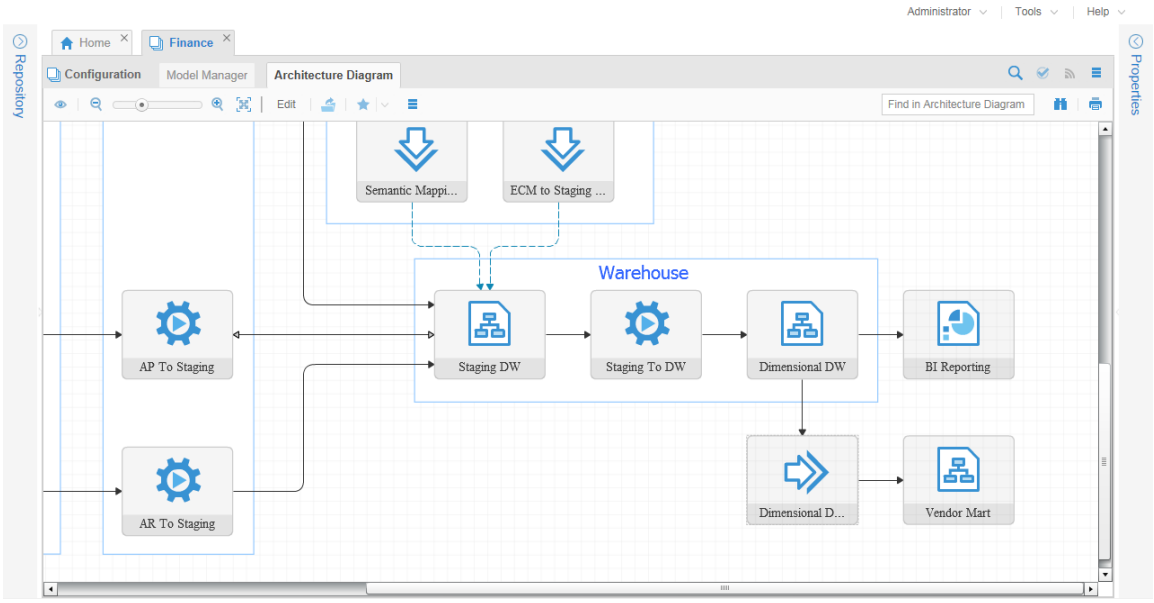

Figure 7 - New Mart and mapping

We now have a good configuration, showing the mapping specification to the Vendor Mart. So, now when we trace lineage, say from the Vendor table in the Accounts Payable model, we now see this model as part of the impact trace:

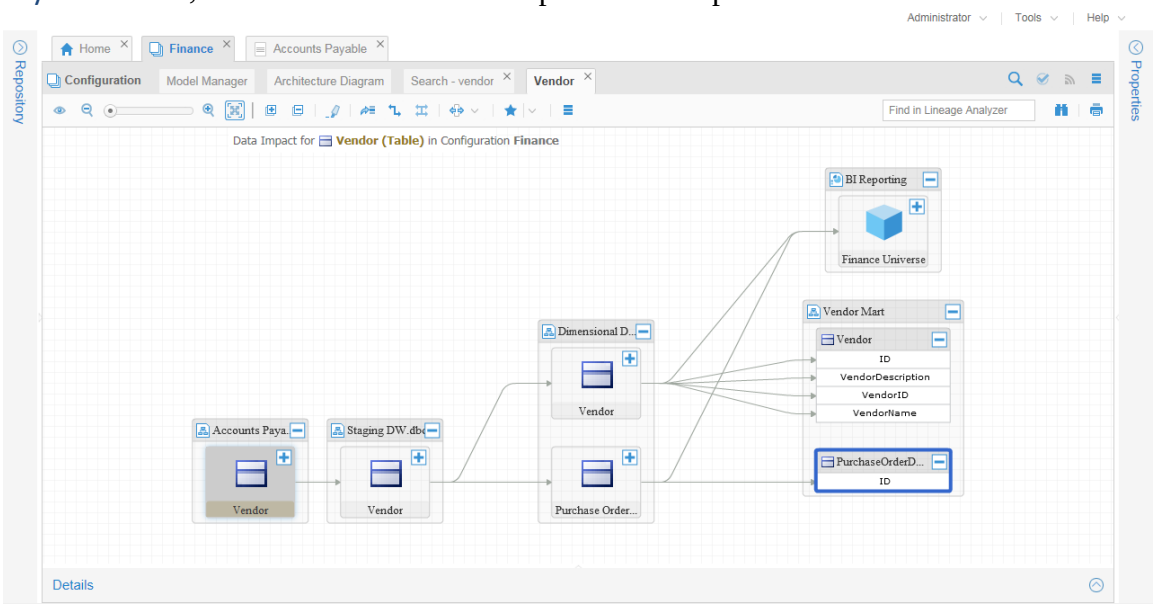

Figure 8 - Lineage impact trace for Vendor

2.3 Detailed Data Mapping Specifications

In this scenario, we are presented with a data source which while not new is new to the Finance configuration we have so far worked with. This new data store is a relational database, and we have a good data model of it, so it is well documented.

Let's import the model.

Navigate to the repository path:
…/Finance System/a –Data Sources - Data Stores/1 – Operational Data Stores/

Create a new model named Adjustments. It is an ERwin 9.6 Data Modeler model:

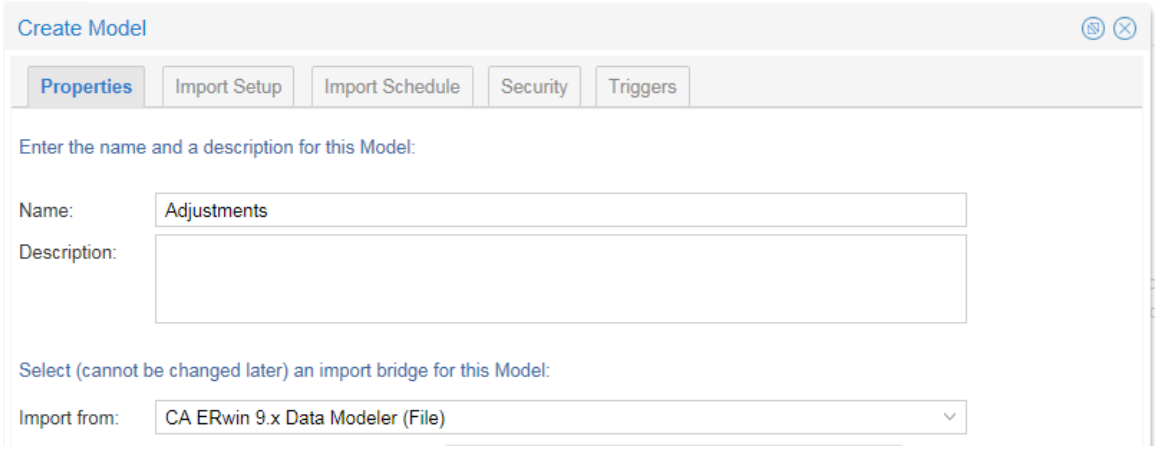

Figure 9 - Creating Adjustments model

Import the model already provided at:

C:\Temp\Models\MetadataManagement\Finance\CaErwin9Xml\Adjustments.xml

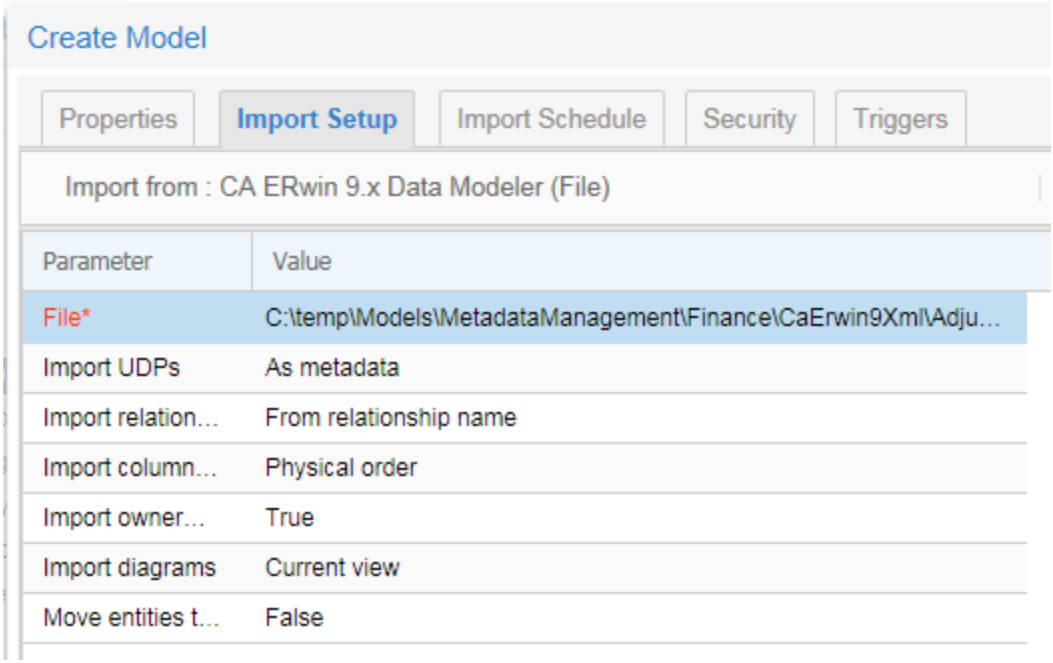

Figure 10 - Importing Adjustments model

Note, we have a nicely modeled data store for the logical model:

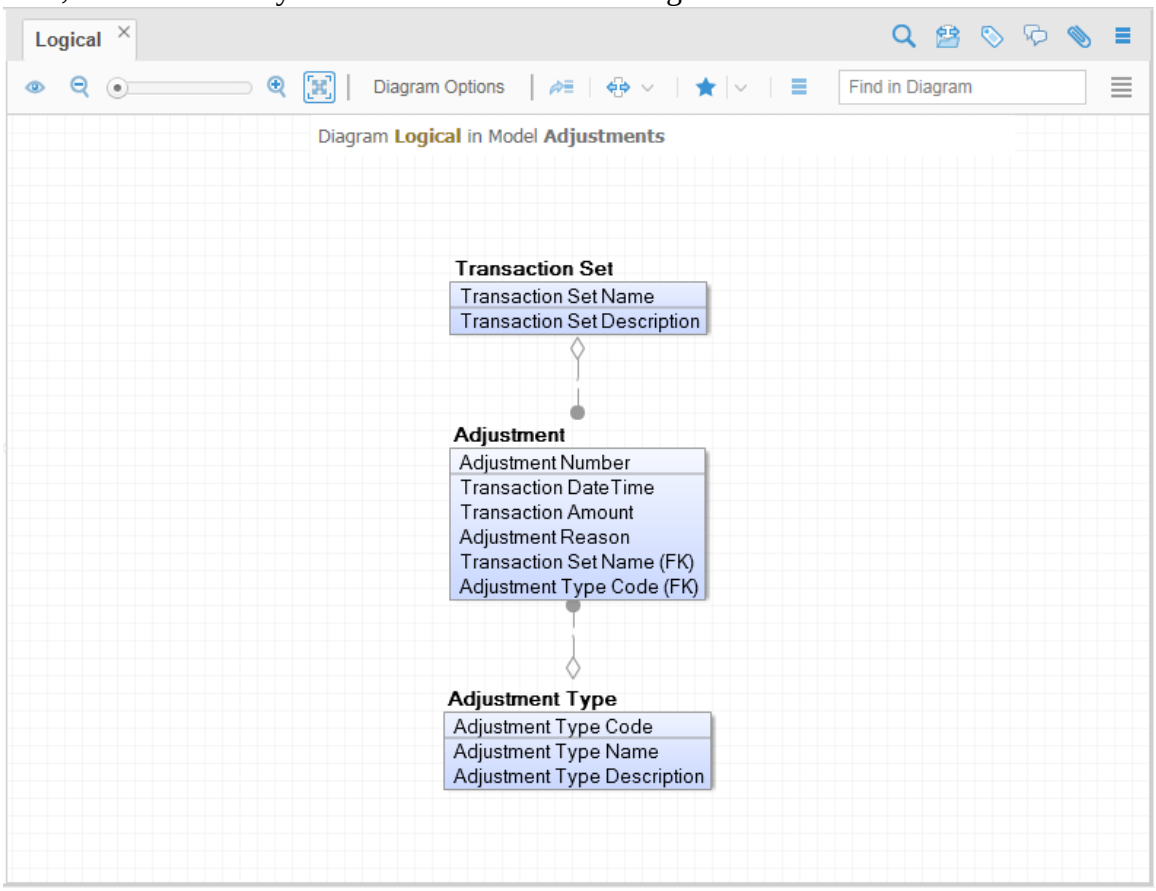

Figure 11 - Logical diagram in Adjustments model

While, the physical side is less obvious:

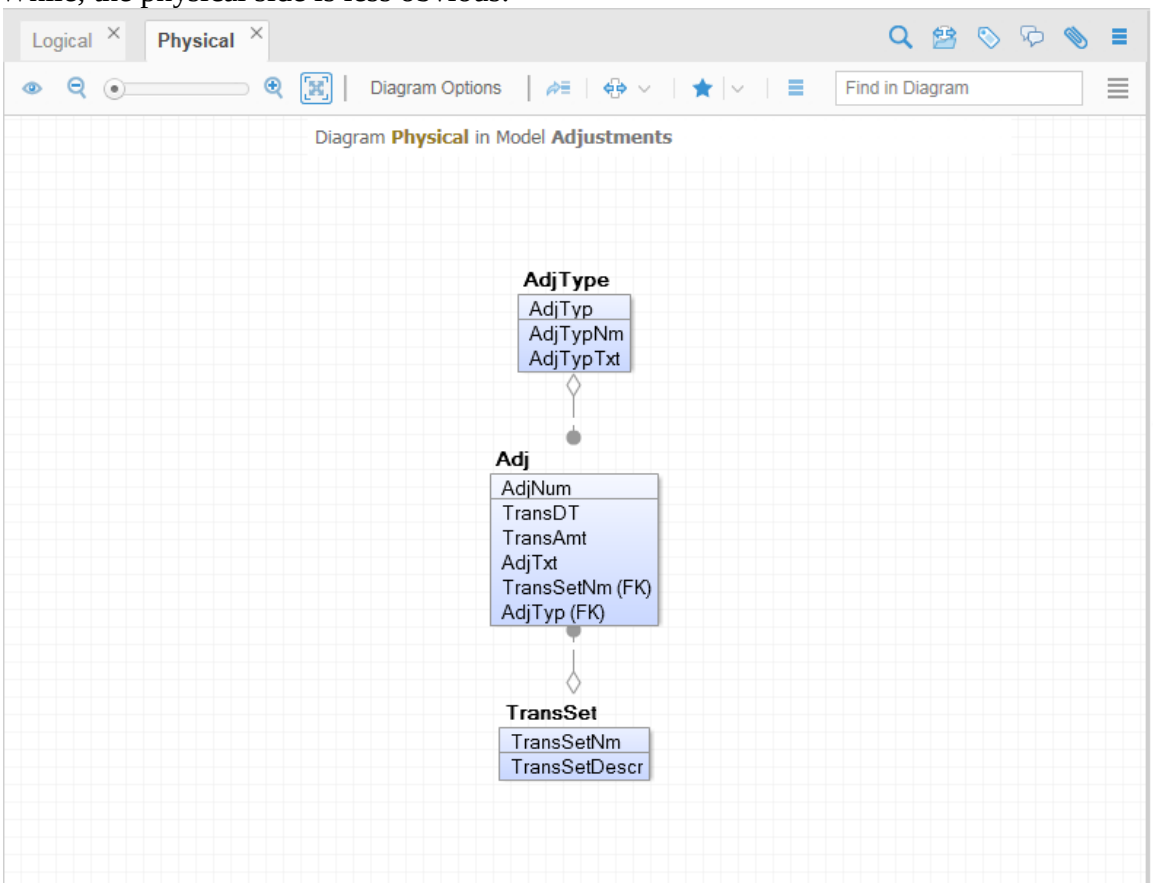

Figure 12 - Physical diagram

However, the load process is a code based an application, and there is no easy way to extract the mapping metadata from it.

There is some mapping documentation (text based) of the basic source-target relationships involved and even some descriptive transformation text. We will use all of this to define a logical representation of the mapping, called the Data Mapping Specifications.

Navigate to the Repository path:
…/Finance System/b - Data Warehouse
And right-click on the [folder] and select New➔ Data Mapping Specifications. Name the Data Mapping Specifications as "Adjustments to Staging DW".

You now have a data mapping specification opened in the work panel. We begin by dragging the source model into the source side of the Data Mapping Specifications. And now drag the destination model into the destination side of the Data Mapping Specifications Now, hide the Repository Panel and we will begin mapping sources to targets.

2.4 Data mapping specification workflow

Before we do so, however, it would be good to discuss mapping project management. In particular, when defining a task for mapping, it is often important to first define a scope for the mapping effort. In general, this scope is simply a specification of which source and target elements should be included in the resulting mapping.

Secondly, one would like to track progress as the effort progresses. It could take some time to collect the proper information to document and certify all of the mapping definitions (and potentially operations). Thus, one would like to track the status of each mapping and report on this.

Meta Integration® Metadata Management (MIMM) provides the basic features to help manage a data mapping specification workflow process, as show below:

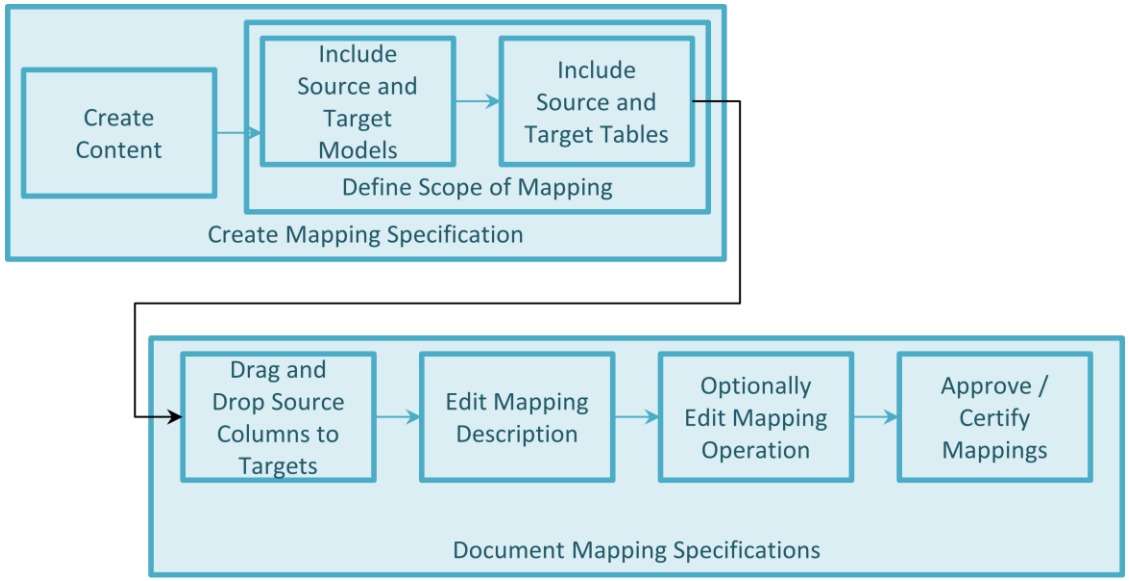

Figure 13 - Data Mapping specification workflow process

2.4.1 Include business names

As it will be convenient to refer to elements by their business names when working with a mapping specification (basically a business level object), please add the Business Name column to the source and target side:

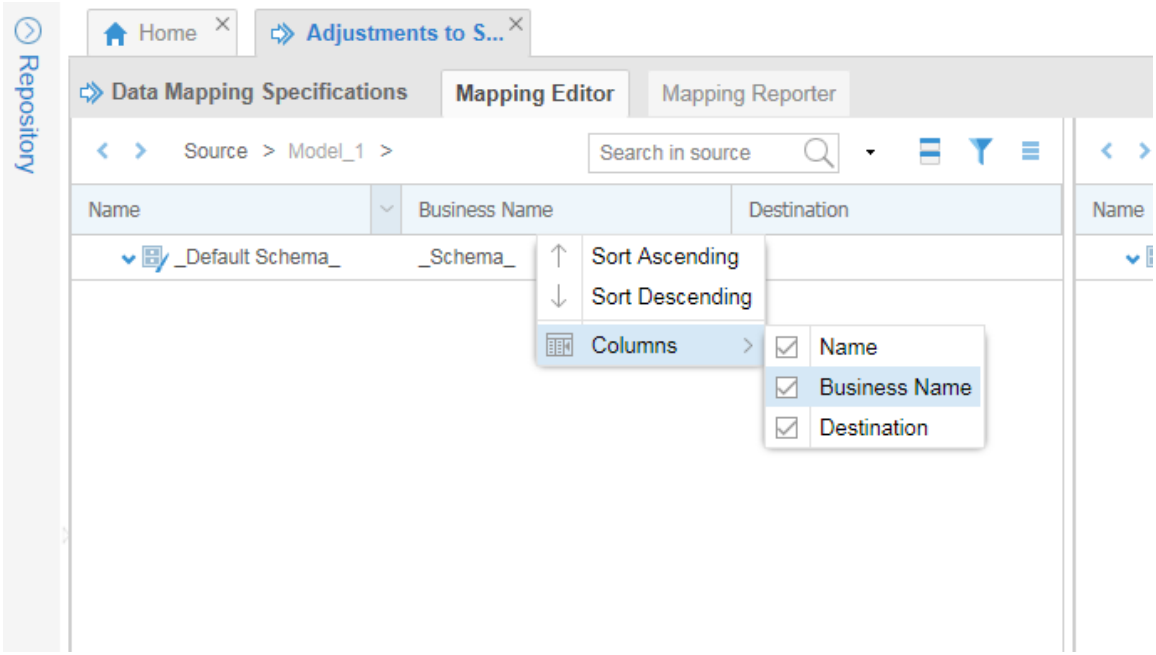

Figure 14 - Add business names

2.4.2 Define mapping scope

Let's begin with the example here. We are going to assign out the responsibilities for mapping to others who are more familiar with the details of the mapping. However, from the mapping documentation that is available (see Appendix I), one can see the tables which should be involved. In particular, we need to limit the presentation to what will be mapped, so as to scope the work to be done.

According to the mapping documentation, there is no need to include the Transaction Set table in the mapping. To ensure this, expand the dbo schema in the target side, right-click on the TransactionSet (User) table and select Exclude:

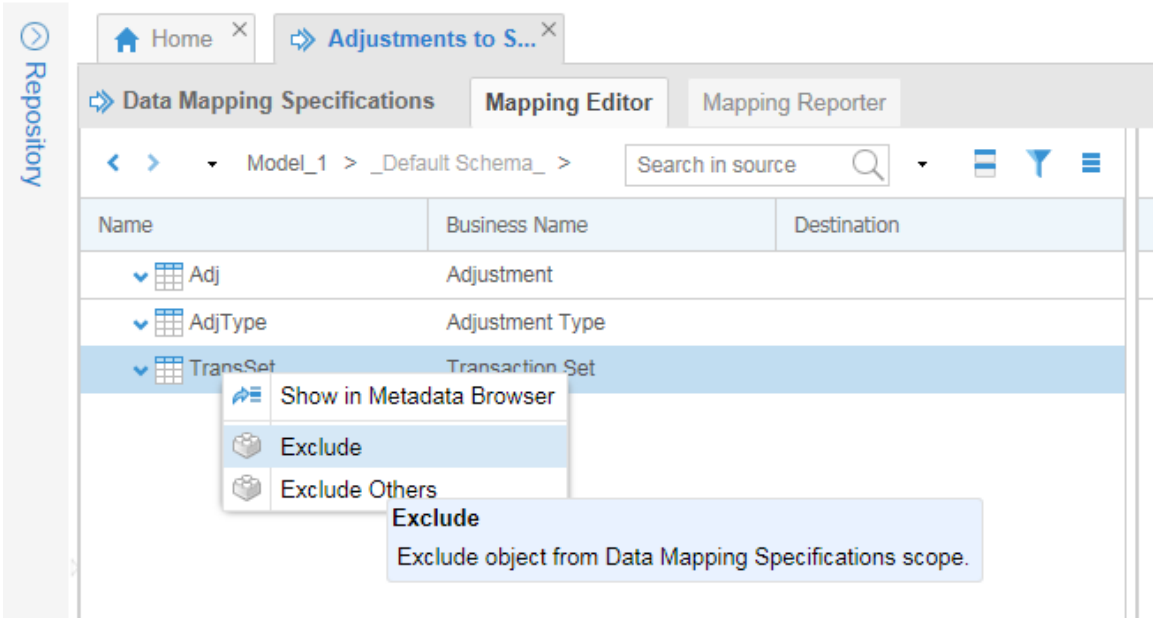

Figure 15 - Exclude the Transaction Set table

Note, the name of the Transaction Set table is now grayed out, indicating that only the Adj (Adjustment) and AdjTyp (Adjustment Type) source tables are to be included in the scope.

Now, for the target side, the mapping documentation says is to map only to the following three tables:

- CustomerPayment
- GLAccount
- VendorPayment

To do so, first right-click on the CustomerPayment table and select Exclude Others:

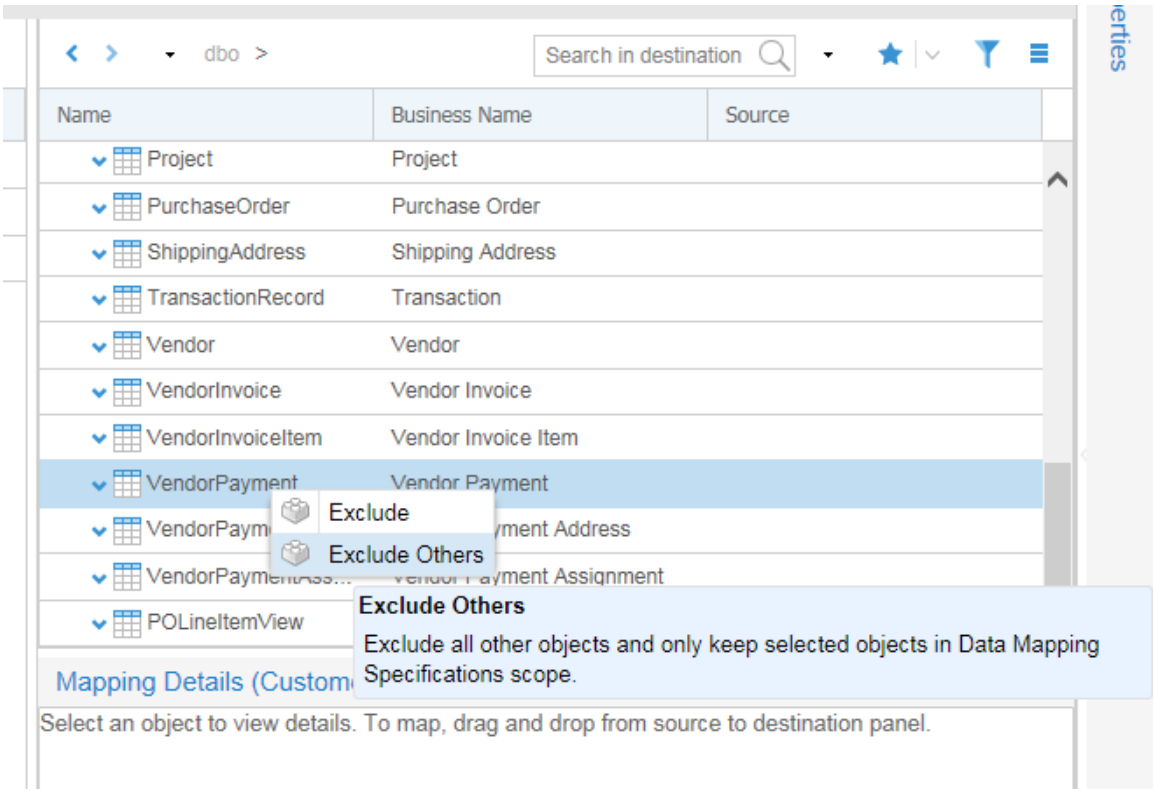

Figure 16 - Exclude Others

Note, Meta Integration® Metadata Management (MIMM) informs the user that this will exclude all other tables. Accept.

As one can see, multi-select (with control-click and shift-click) is enabled in the source and target sides of a data mapping specification.

2.4.3 Filtering

According to the mapping documentation:

> Adjustments are migrated as payments (either to a vendor or from a customer). The vendor and customer identification and the general ledger account number are hard coded. Adjustment based payments are hand resolved, as there are few each FY Quarter.

This is why only these three tables are involved in the target.

Thus, we now have only three source and three target tables in our scope. To simplify the presentation, then, use the Filter action icon:

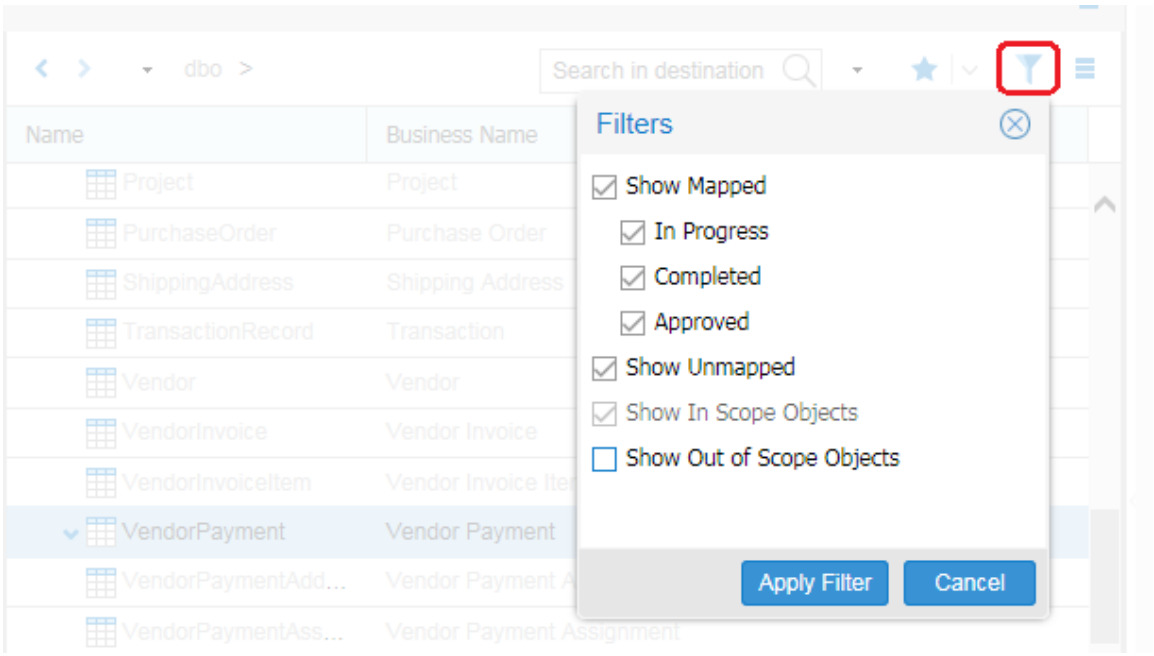

Figure 17 - Filter target objects

Select the Show Unmapped checkbox and click on the Apply FIlter button.

And we have a more manageable list:

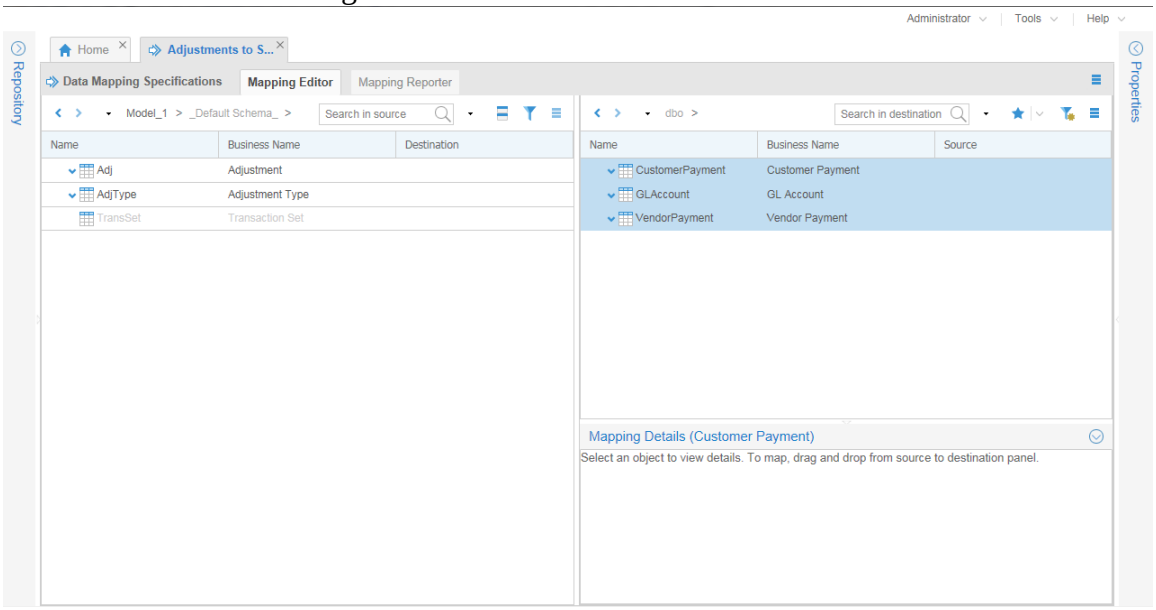

Figure 18 - Filtering

Now, do the same on the source side:

2.4.4 Creating a map

Ok, now, from the mapping documentation, Let's see what the mapping documentation says:

> Customer payments are created for each adjustment which is >0.

Thus, one could try draging the Adjustment table to the Customer Payment table. However, if you were to do that, you would have to remove the mapping as there are no matching column names. So we will have to map the columns manually.

At the column level, we see that Adjustment.TransactionAmount is used to populate Customer Payment.PaymentAmount, Vendor Payment.PaymentAmount and GL Account. AmountAvailable. Expand the Adjustment source table and Customer Payment target table and drag the TransactionAmount column to the PaymentAmount column, creating a map:

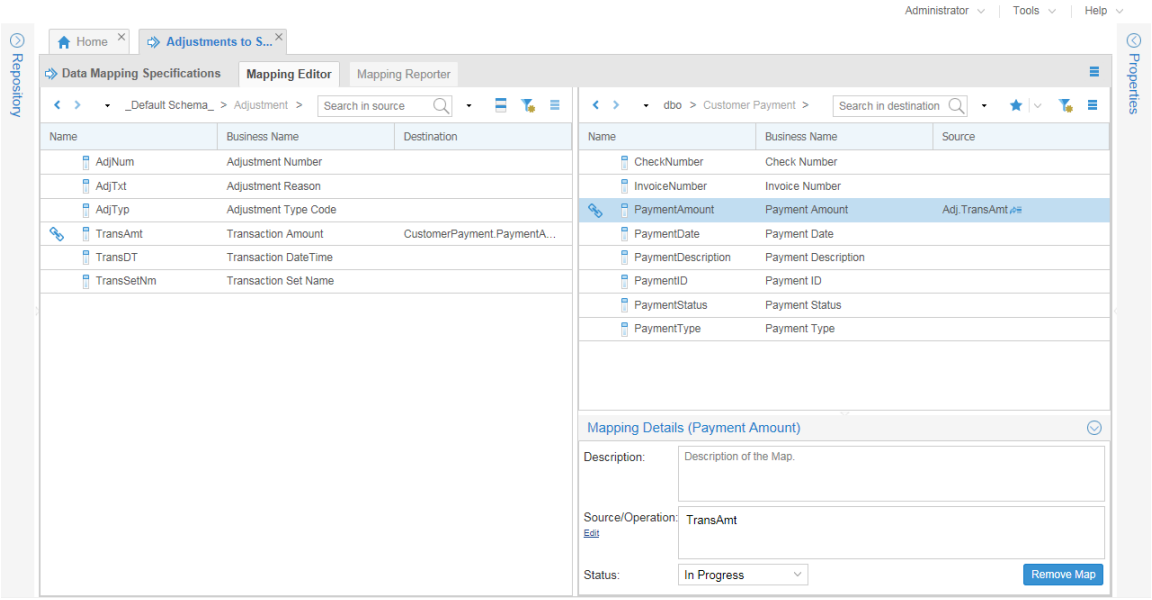

Figure 19 - Map from Contact name to Customer Name

Column level maps are seen as target defined, meaning that each column mapped in the target defines an individual map which may have references to one or more source column. In this way, this map just created is referred to as the Customer Payment.PaymentAmount map, and any description or operation is thus identified.

Note, the operation is already populated with the source column physical name (TransAmt). One can simply drag and drop into the Operation to include additional columns.
Let's see what the mapping documentation says:

> Customer pay amount is derived from all adj amount

Thus, we need to make an entry to the Description for this mapping, copying in the line of text from the mapping documentation except above.

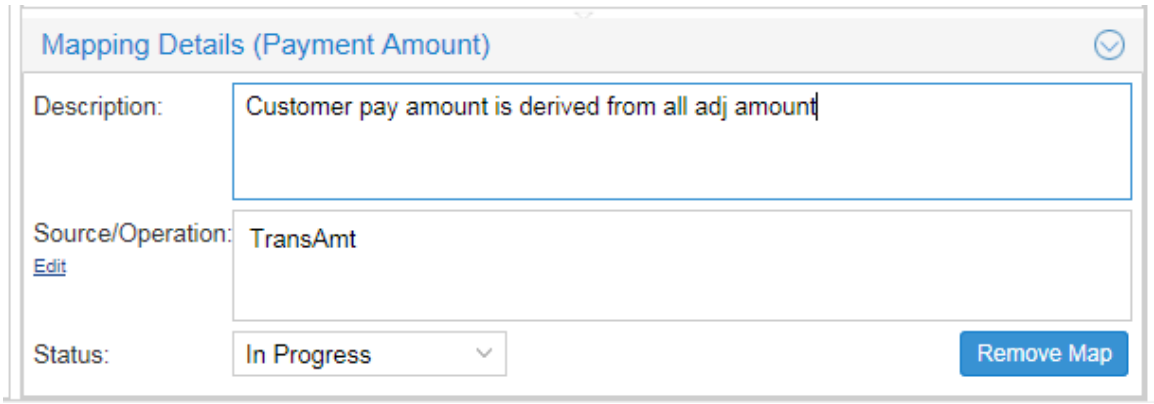

Figure 20 - Pasted mapping

Also, one may populate the text of the Description by drag and drop. Since the current description uses "adj amount" to refer to "TransAmt", we can add the correct name to the description by drag and drop. Drag the source column into the Mapping Details panel:

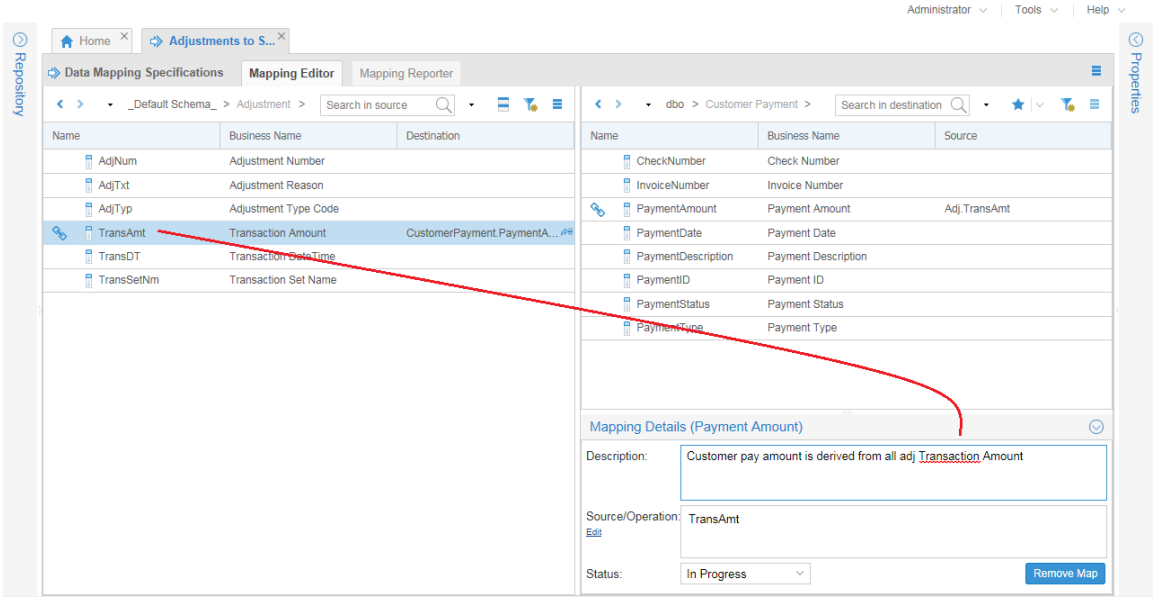

Figure 21 - Drag name into description

Note, the business or logical name (Transaction Amount) is placed in the mapping details. And then add a space and parentheses.

Now that we have completed this map, change the status to Completed:

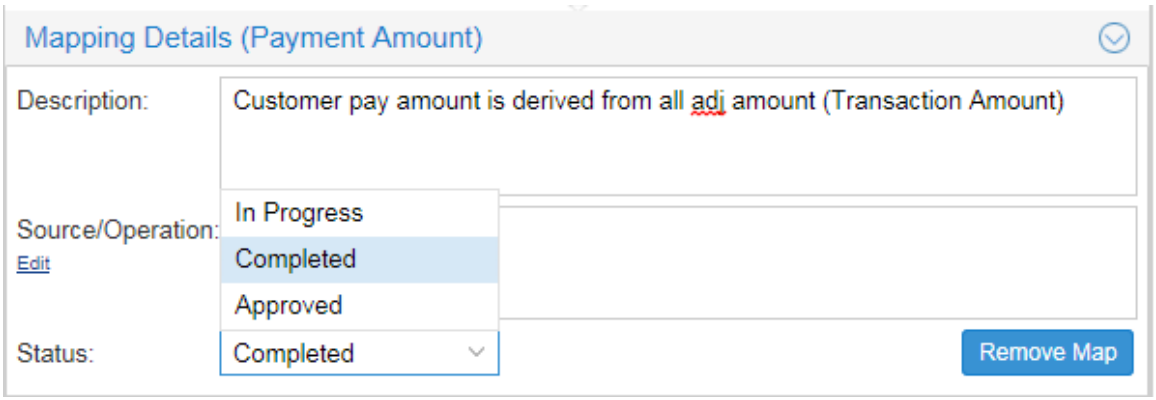

Figure 22 - Set mapping status

With that map completed, drag the following three source columns into the PaymentID target column:

- AdjNum (Adjustment Number)
- TransDT (Transaction Date)
- TransSetNm (Transaction Set Name)

In this case, the target column is to be composed from the three source columns identified. Enter the text "Based upon Adjustment Number, Transaction Date and Transaction Set Name" in the Description field, either through typing only or also using the drag and drop to populate the text:

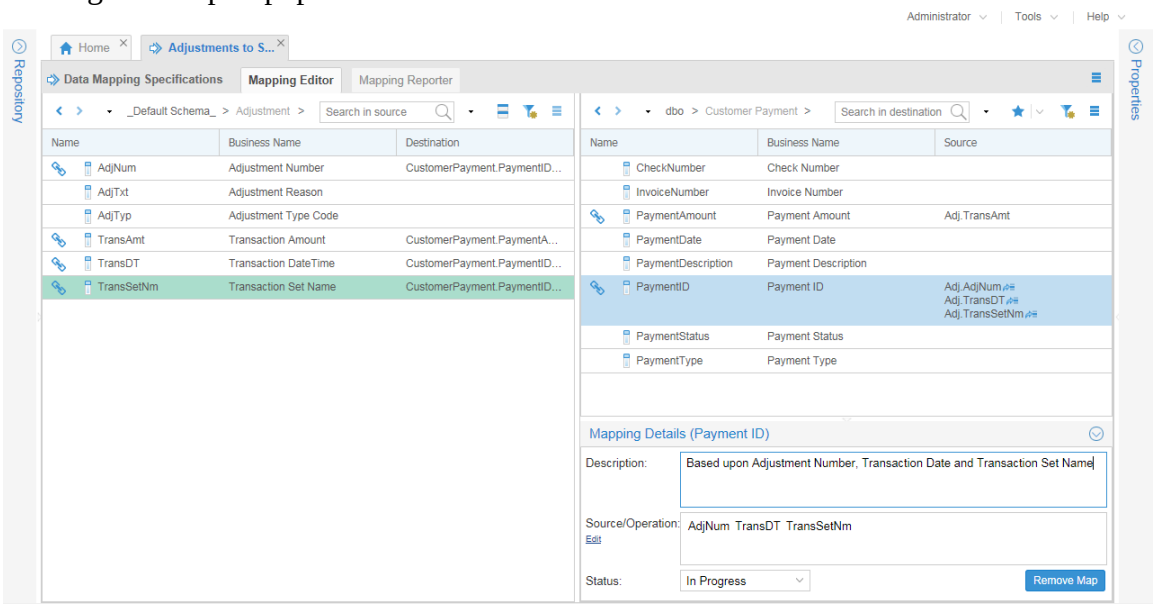

Figure 23 - Column map to Customer ID

With the mapping defined in this way, lineage will be shown but no meaningful operation will be presented. Instead, the description is where the details for the map are presented. Please also set the Status for the map to Completed.

Finally, according to the requirement for the last column map to the PaymentDescription, the column map should be:

Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name:
- if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here
Otherwise, the decoded value of Adjustment Type Code should be included in the description text.

Thus, drag the Adjustment Reason and Adjustment Type Code into the PaymentDescription:

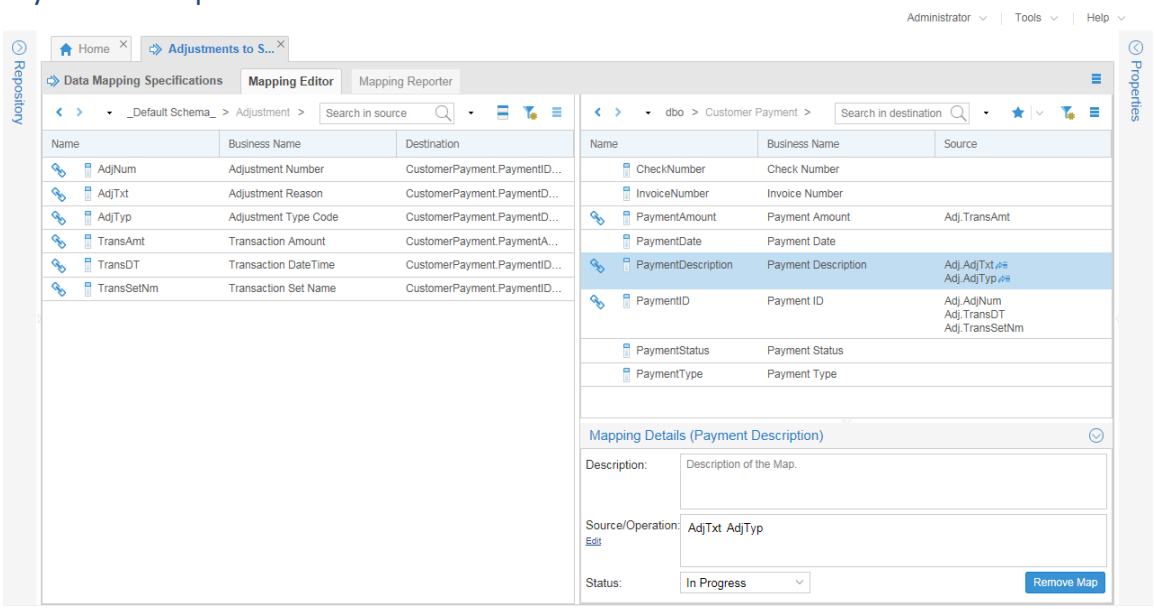

Figure 24 - Drag sources to target

As one can see from the above description, some of the source information will come from a lookup of the Adjustment Type Name and Adjustment Type Description in the Adjustment Type table. Hence, we will need to drag and drop from columns in the Adjustment Type table, and thus must have it opened on the source side. One could simply navigate to that table and perform the drag and drop. However, one could also take advantage of the Split feature and expand the Adjustment Type table in another panel on the source side:

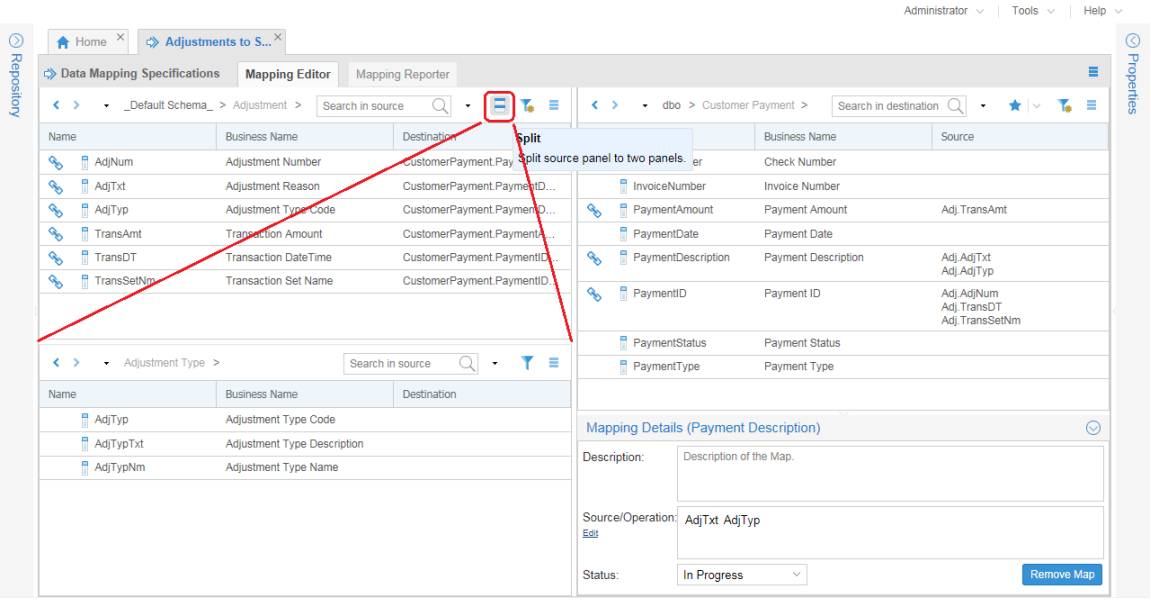

Figure 25 - Source side split

Now, one may drag and drop the two columns into the PaymentDescription column on the target side along with the Adjustment Type Code in the Adjustment Type Table and populate the Description. Please also set the Status for the map to Completed:

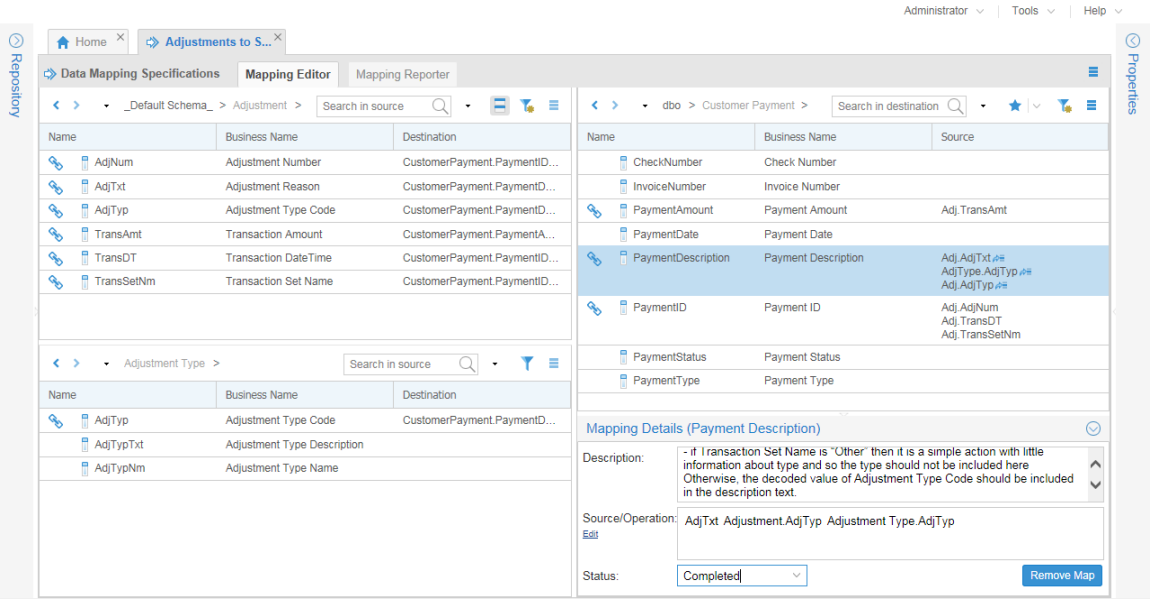

Figure 26 - Completed mapping of PaymentDescription

Click on the Mapping Reporter tab and note the maps as defined is presented:

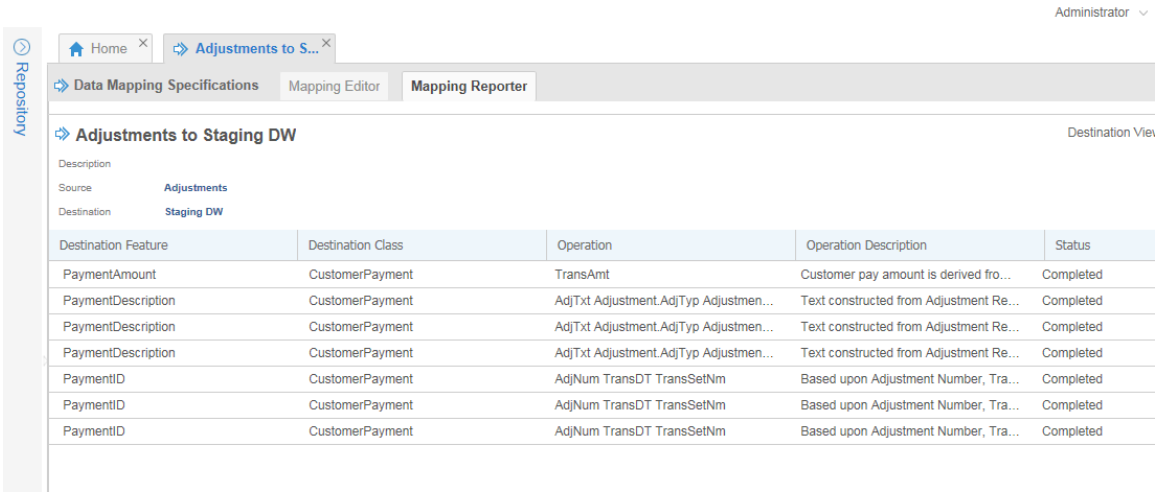

Figure 27 - Mapping Report

2.4.5 Review and approve maps

It is now time to review our work. Return to the Mapping Editor tab, navigate to the top of the target structure (dbo):

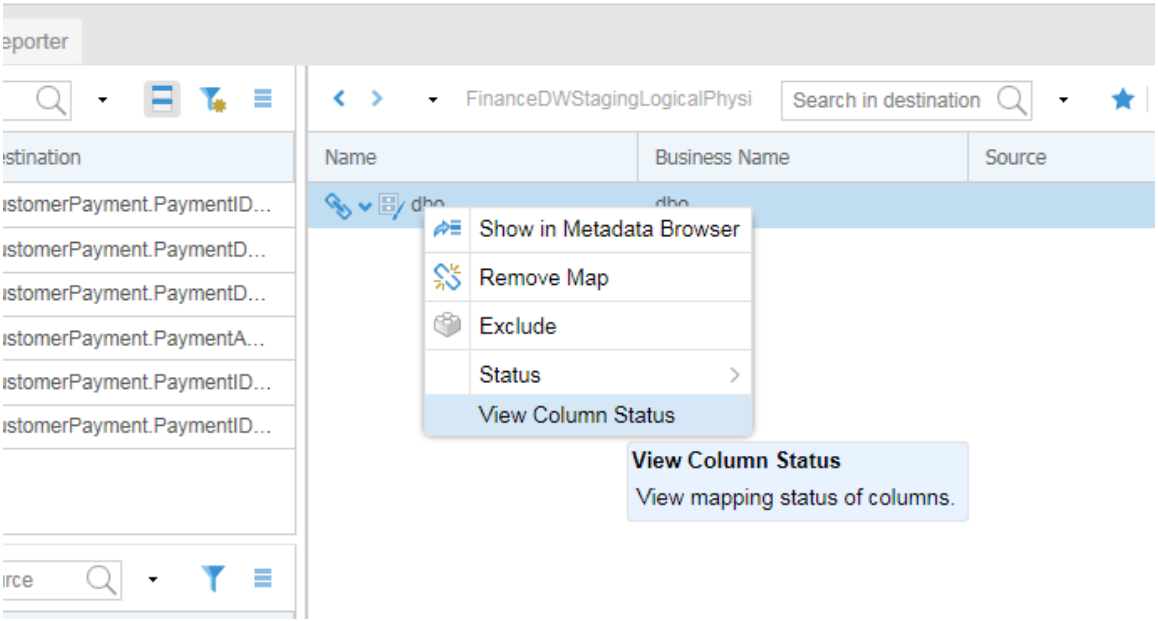

Figure 28 - View column status menu

Then select View Column Status:

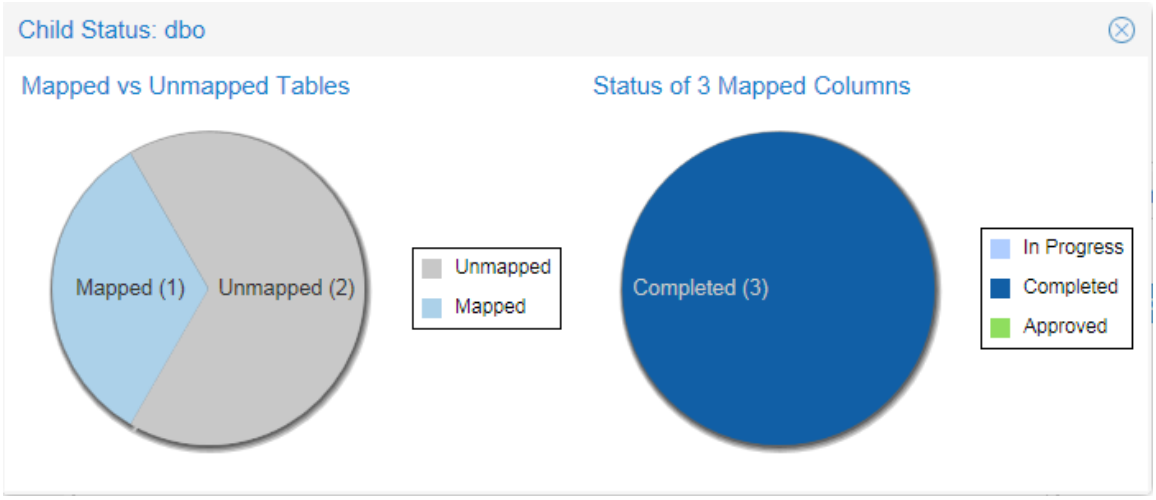

Figure 29 - View Column Status

Here we see that 1 out of 2 total tables (in scope) are mapped and that of the 3 mapped columns all three are completed but not are approved.

Now, someone should go through and review these maps. Based upon that review, the maps to the PaymentDescription and PaymentID columns within the Payment table are approved, so we should change the Status of those to Approved. Doing so updates the report as follows:

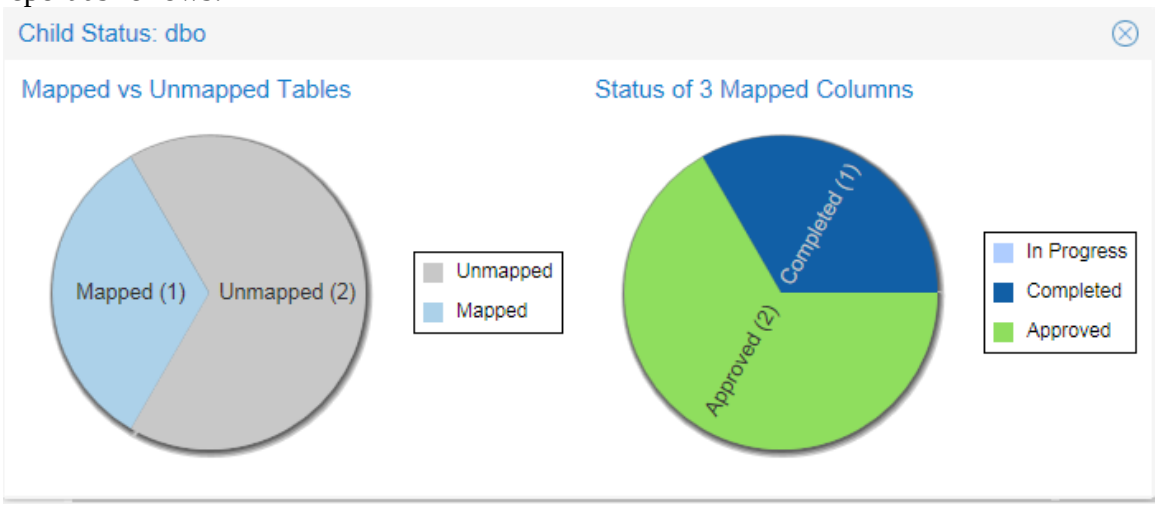

Figure 30 - View Column Status after approval

Now, to complete the maps for the other two target columns, please refer to the following Appendix. Here is the result in the Mapping Reporter:

2.4.6 Add to configuration

Let's add the model and mapping to our configuration. Drag the Adjustments model into the ODS [folder] in the Finance configuration. Also include the new mapping

specification (Adjustments to Staging DW) in the Warehouse [folder] in the configuration. Build the configuration.

Now go to the Architecture Diagram tab and edit the layout of the diagram to move the mapping specification and new model as shown here:

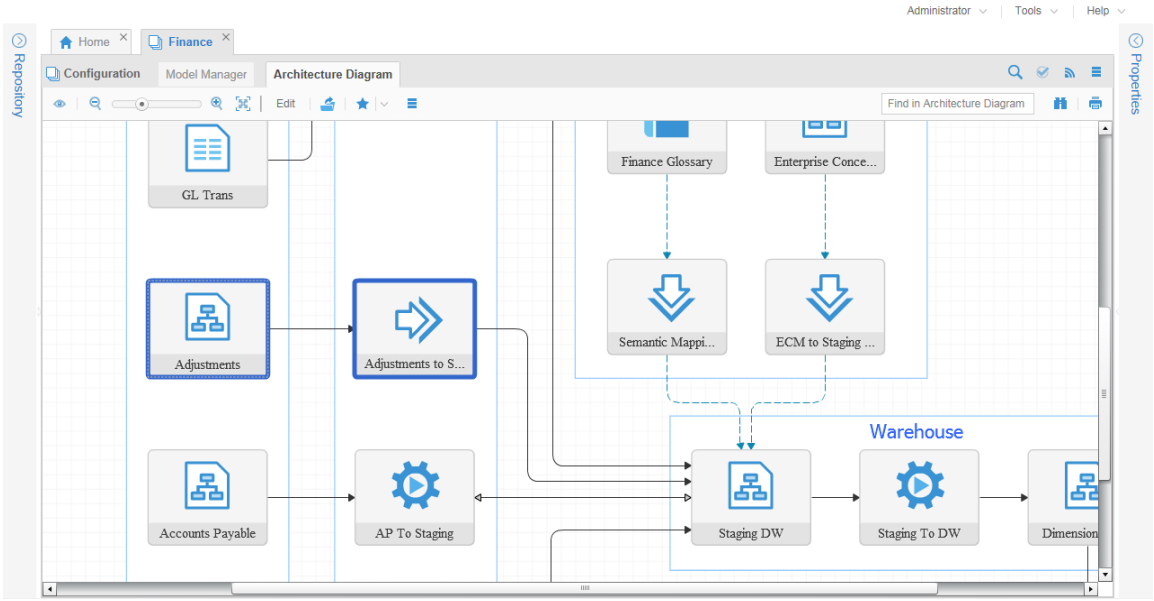

Figure 31 - New Mart and mapping

We now have a good configuration, showing the mapping specification to the Staging DW. So, now when we trace lineage we will see this new database as part of the lineage.

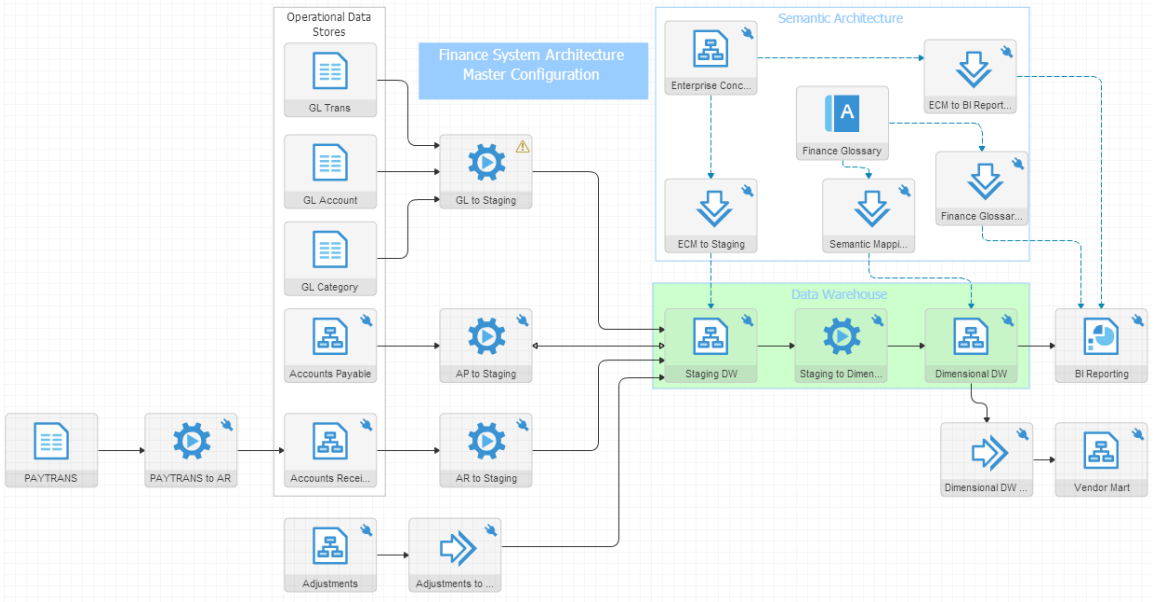

Figure 32 - Final configuration

3 Best Practices

3.1 Repository Organization

In general, the repository organization structure chosen should reflect the business cases being emphasized by your organization. The considerations are as follows:

- Ease of locating contents
- Efficiency of assigning roles and permissions against the contents

In order to address the above considerations, it is generally recommended that the Repository be organized the way in which project and product management responsibilities are organized. Thus, if your organization has separate "groups" and responsibilities by particular business administration area, e.g., Finance, CRM, Payroll/Personnel, etc., then that should be the high level organization. Alternatively, you may have an organization driven by the organization of the IT architecture, e.g., business intelligence, warehouse, operational data stores, etc. The tutorial is in fact a hybrid of the two structures above, where it is organized by business function (Finance) at the highest level and then by IT architecture at the lower levels. A third method revolves around the technologies involved. E.g., it may be that the investment in a large ERP drives responsibilities, or a particular BI technology. A fourth method is based upon initiatives, and can oftentimes relate to specific activities that the enterprise or the IT organization has prioritized, e.g., The 2020 Initiative, or the Big Data Project.

Given the above, let's take for example the organizational structure in the tutorial. In this case, we have organization by business area and then by architecture within that:

- Finance
    - Operational Data Stores (ODS)
    - Warehousing (DW) and Data Integration (DI)
    - Business Intelligence (BI)
    - Information Management (logical/conceptual modeling and semantic mapping)
- CRM
    - Operational Data Stores (ODS)
    - Warehousing (DW) and Data Integration (DI)
    - Business Intelligence (BI)
    - Information Management (logical/conceptual modeling and semantic mapping)
- PayPers
    - Operational Data Stores (ODS)
    - Warehousing (DW) and Data Integration (DI)
    - Business Intelligence (BI)
    - Information Management (logical/conceptual modeling and semantic mapping)
- Information Management (logical/conceptual modeling, business glossary and semantic mapping)

In terms of ease of location of contents and assigning permissions, it is important to remember that the Repository structure is only seen by users which have access to the Metadata Manager UI. The Explorer UI users, which are the vast majority, will use

configuration organization. Thus, considerations here (Repository organization) only relate to those users involved in the management of the metadata. Ease of locating contents and assigning permissions then will most likely be based upon the same criteria: the IT organization project management. Thus, the above organizational structure makes sense where the separate business areas are how IT project management is assigned.

Note also that the Information Management is at both levels. In general, you will find that while there are project specific logical or conceptual models and semantic mappings to the physical architecture, there is great importance placed on integrating this information into an overall whole. Thus, the business glossary will generally be placed at the highest level, along with enterprise level conceptual models and their semantic mappings.

Now, if project management is assigned by position in the IT architecture, e.g. "The Warehouse Team" vs. the "BI Folks", you may wish this Repository organization:

- Operational Data Stores (ODS)
  - Finance
  - CRM
  - PayPers
- Warehousing (DW) and Data Integration (DI)
  - Finance
  - CRM
  - PayPers
- Business Intelligence (BI)
  - Finance
  - CRM
  - PayPers
- Information Management (logical/conceptual modeling, business glossary and semantic mapping)
  - Finance
  - CRM
  - PayPers

Note here, the Information Management [folder] contains specific folders for the enterprise business areas. It is likely that such assets will exist in the organization. However, it is also generally the case that the ultimate goal is to integration and unify this information, hence it is all placed under Information Management, not under the warehouse vs. BI.

Finally, initiative may come and go and this could affect the organizational structure either temporarily or long-term. This is especially true in the case of initiative influenced organization. Thus, one may see the following (hybrid) for a manufacturing company:

- ERP
  - Operational Data
  - Warehousing (DW) and Data Integration (DI)

- Major Product Line One
    - Operational Data Stores (ODS)
    - Warehousing (DW) and Data Integration (DI)
    - Information Management (logical/conceptual modeling and semantic mapping)
- Major Product Line Two
    - Operational Data Stores (ODS)
    - Warehousing (DW) and Data Integration (DI)
    - Information Management (logical/conceptual modeling and semantic mapping)
- Big Data Initiative
    - Hadoop
    - DI
    - Self-server BI
- Business Intelligence Tool (BI)
- Information Management (logical/conceptual modeling, business glossary and semantic mapping).

3.2 Repository object naming standards

As we have seen above, Meta Integration® Metadata Management (MIMM) has a robust and extensible [folder] structure in which one can uniquely place and refer to models, configuration, mappings, etc. Because of this, it often appears to make sense to define simple and short names for objects, as they can be identified uniquely by their location in the [folder] structure.

Unfortunately, while the above assumption makes sense in the context of the Repository panel tree, these same objects can oftentimes be found in lineage trace diagrams, configuration architecture diagrams, search results or pick lists where the context may be difficult to see or not readily available. Thus, it is important to give objects in the Repository a complete enough name to be easily identified in any of these contexts.

An added consideration is that one should use a name that is not too long, so that it will show uniquely in the lineage and architecture diagrams, as well as tabs in the central work panel. Only the first 12 - 20 characters are displayed for objects in these tabs and only the first 11-18 characters are displayed for objects in the lineage

Finally, it is important to not use the object type in the name, as this is superfluous information and can make the name too long. E.g., instead of naming a model the "Accounts Payable Model", it is not necessary to have the work "Model" in the name, as it will already have the icon for a model associated with it.

Appendix A

Mapping Specification:

Destination Class	Destination Feature	Operation Description	Source Feature	Source Class
CustomerPayment	CheckNumber	Fixed “Adjustments\ Check Number"		
CustomerPayment	InvoiceNumber	Fixed “Adjustments\ InvoiceNumber"		
CustomerPayment	PaymentAmount		TransAmt	Adj
CustomerPayment	PaymentDate		TransDT	Adj
CustomerPayment	PaymentDescription	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTxt	Adj

CustomerPayment	PaymentDescription	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTyp	Adj
CustomerPayment	PaymentDescription	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTyp	AdjType

CustomerPayment	PaymentDescription	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTypNm	AdjType
CustomerPayment	PaymentDescription	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTypTxt	AdjType
CustomerPayment	PaymentID	Based upon Transaction Set Name, Adjustment Number and DateTime	AdjNum	Adj
CustomerPayment	PaymentID	Based upon Transaction Set Name, Adjustment Number and DateTime	TransDT	Adj
CustomerPayment	PaymentID	Based upon Transaction Set Name,	TransSetNm	Adj

		Adjustment Number and DateTime		
CustomerPayment	PaymentStatus	Set to \Paid in Full\""		
CustomerPayment	PaymentType	Fixed “Adjustments\ PaymentType"		
General Ledger Account	Account Amount Available	Subtract Transaction Amount from the Account Amount Available for the fixed “Adjustments\ GLAccountNumber"	TransAmt	Adj
General Ledger Account	General Ledger Account Number	Fixed “Adjustments\ GLAccountNumber"		
VendorPayment	CheckNumber	Fixed “Adjustments\ Check Number"		
VendorPayment	Description	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTxt	Adj

VendorPayment	Description	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTyp	Adj
VendorPayment	Description	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTyp	AdjType

VendorPayment	Description	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTypNm	AdjType
VendorPayment	Description	Text constructed from Adjustment Reason and the decoded value of Adjustment Type Code depending upon the Transaction Set Name: - if Transaction Set Name is “Other” then it is a simple action with little information about type and so the type should not be included here Otherwise, the decoded value of Adjustment Type Code should be included in the description text.	AdjTypTxt	AdjType
VendorPayment	PaymentAmount	Customer pay amount is derived from all adj amount (Transaction Amount)	TransAmt	Adj
VendorPayment	PaymentType	Fixed “Adjustments\ PaymentType"		
VendorPayment	Status	“Paid in Full\ Status"		

VendorPayment	VendorID	Fixed “Adjustments\ VendorID"		
VendorPayment	VendorInvoiceNumber	Fixed “Adjustments\ VendorInvoiceNumber"		
VendorPayment	VendorPaymentID	Based upon Transaction Set Name, Adjustment Number and DateTime	AdjNum	Adj
VendorPayment	VendorPaymentID	Based upon Transaction Set Name, Adjustment Number and DateTime	TransDT	Adj
VendorPayment	VendorPaymentID	Based upon Transaction Set Name, Adjustment Number and DateTime	TransSetNm	Adj