Metadata Management Tutorial

Version and Configuration Management 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.

NB: This is a self-contained tutorial. However, it assumes you have at least an understanding of how to use Meta Integration® Metadata Management (MIMM) for basic Metadata Management activities. Thus, it is best to have worked through the Metadata Management Tutorial first.

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 About Version and Configuration Management

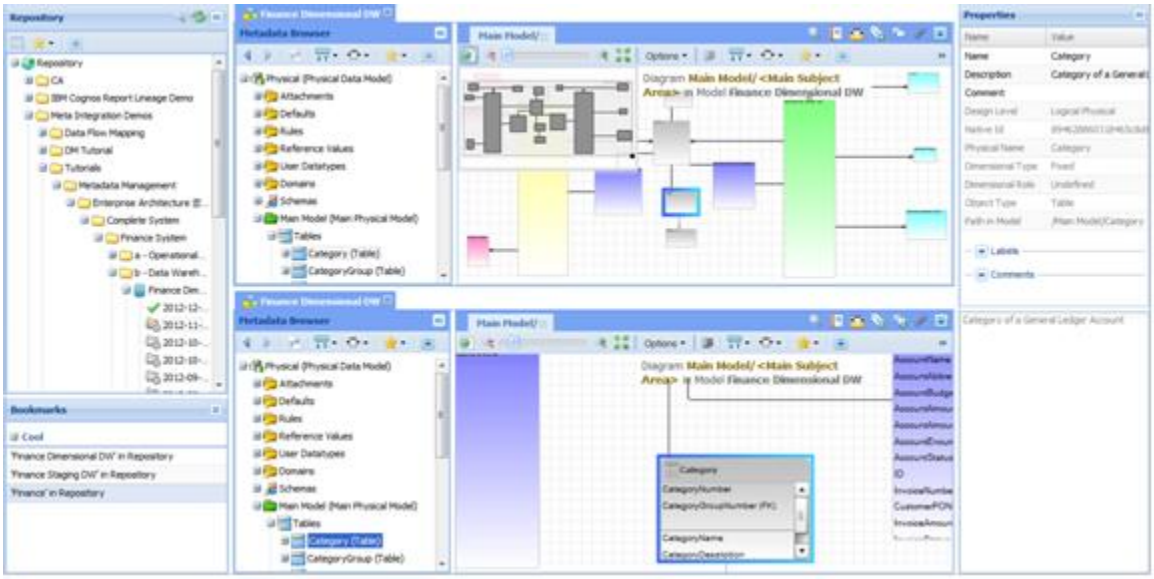


Figure 1 - Multiple versions of models in Meta Integration® Metadata Management (MIMM)

Not only must the repository be able to import (and export on demand in any format and to any tool) or harvest metadata many times as needed, it must be able to manage the *versions* created by this continuous activity. It must also be fundamental to the repository organization for administrators to then organize, publish and selectively present the information in appropriate *configurations* of metadata, as is required for the correct and precise answers to a wide range of "cuts" across this metadata.

Meta Integration® Metadata Management (MIMM) was designed from the ground up with version and configuration management as a key capability. In fact a significant portion of this tutorial surrounds good version and configuration management activities and practices.

3 Metadata Survey

Recall, from the Metadata Management Tutorial, we create a single version and single configuration architecture in the Published configuration.

Now, we will extend upon that an look at the same architecture, only we will manage the change associated with harvesting new versions or models, comparing and identifying changes, looking at impact analysis both in the data flow and semantic layer and create new versions of the configuration as we go.

To do this, we will "go back in time" in terms of the Tutorial and import earlier versions of the models. In addition, we will create two new configurations: Development and Production, representing the architecture and metadata in the Dev and Prod IT environments respectively.

3.1 Setting Up the Tutorial

In order to create this structure in the repository for the tutorial, a Javascript based backup of the Metadata is provided to you. To execute this script, sign in as Administrator (generally with password "Administrator" in a new installation), open Meta Integration® Metadata Management (MIMM) and right-click on the top of the *Repository Tree* (far left panel). Select More… > Scripts › From the Start: Metadata Management Tutorial – Version and Configuration Management From The Start (Create the Metadata Version and Configuration Management Tutorial initial structure) from the list of scripts.

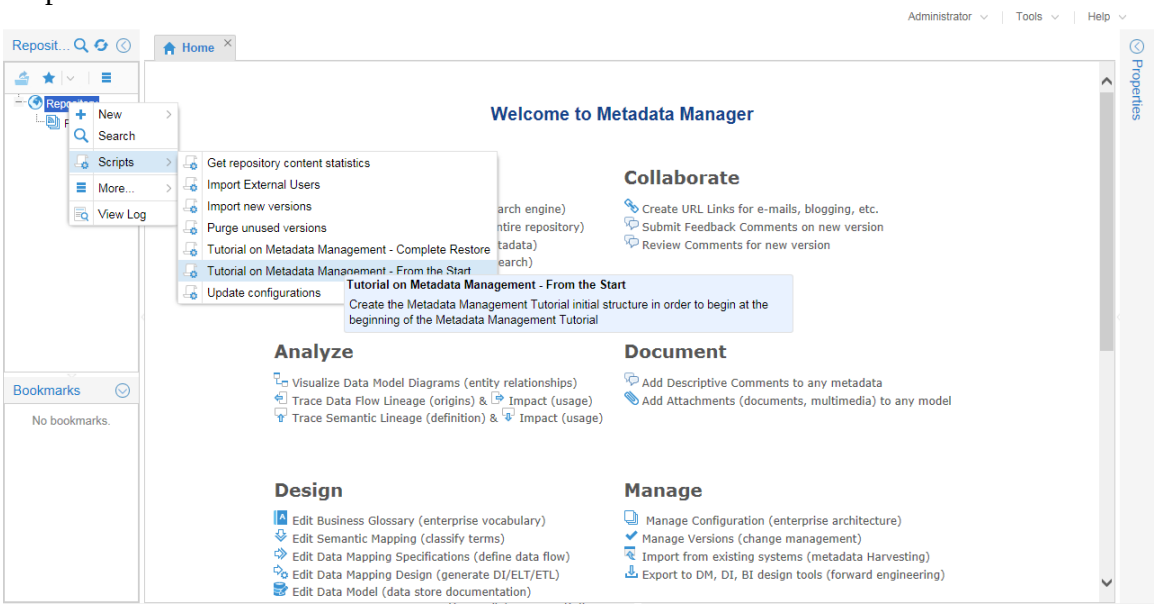


Figure 2 - Selecting the tutorial restore script

Click on the Run Script button in the Run Custom Script dialog that is presented.

Once the script has completed close the dialog. You may need to click on the Refresh control ( ) at the top of the *Repository Tree* panel. Note how the metadata cataloged in the previous section are reflected in the tutorial repository structure under the [folder]s Tutorial/Metadata Management.

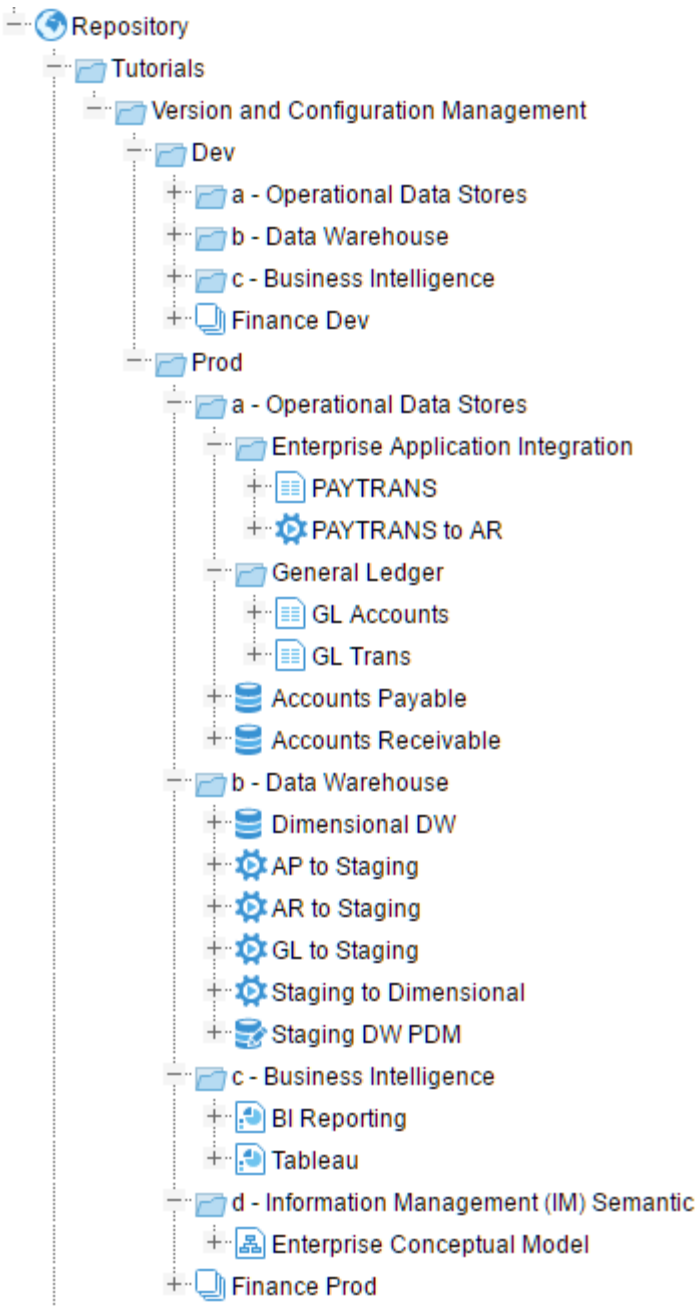


Figure 3 - Tutorial repository structure after executing the restore script

After executing the tutorial restore structure script, the basic [folder] structure of the repository is created. Note, there are two major sections Prod and Dev. The structure of Prod is very similar to the Core Principles section of the Metadata Management tutorial, which also already covered the mechanics of harvesting the metadata embedded in data stores and data process tools. The Dev section is nearly a copy of Prod. The distinction is that the models point to different file locations.

Also note that they configurations are already created, stitched and ready to go.

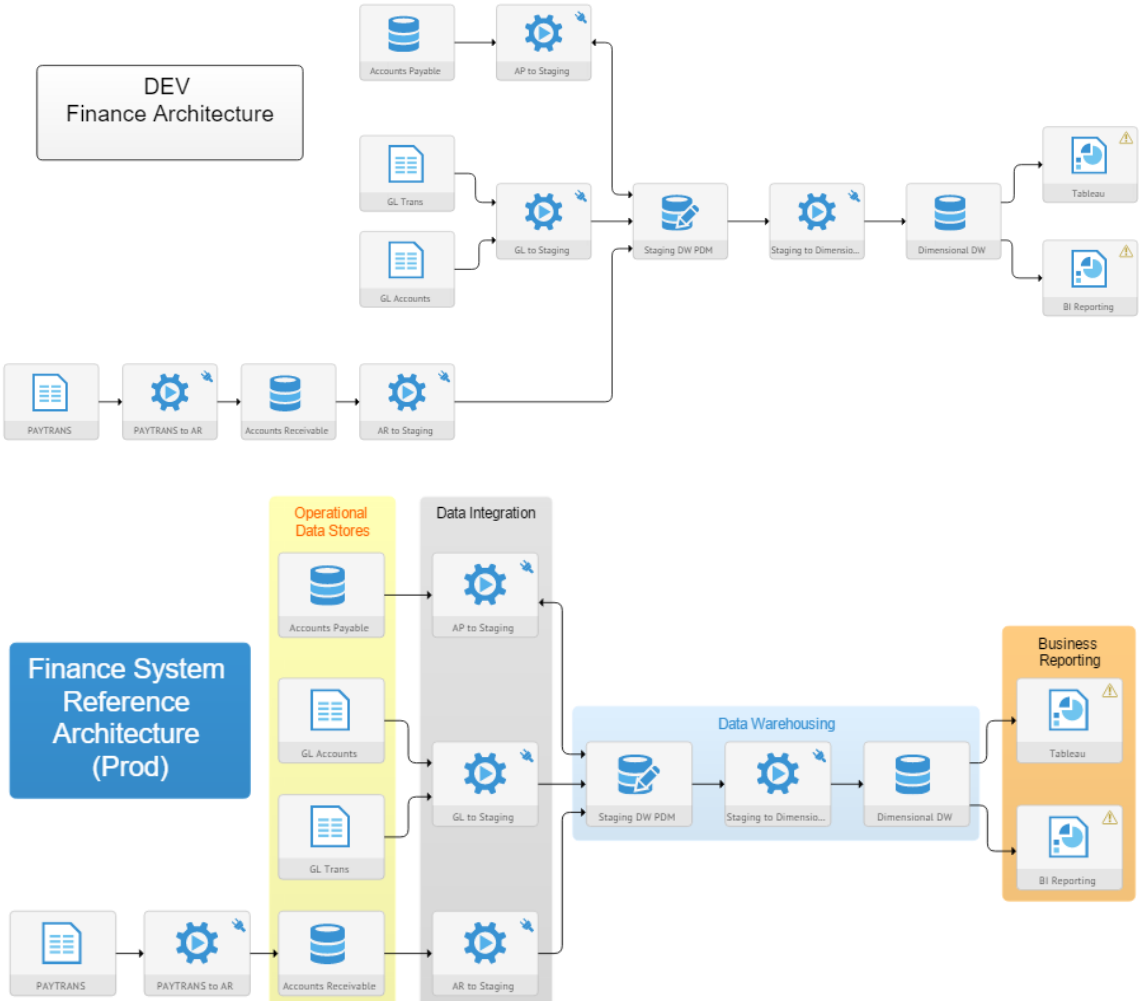


Figure 4 - Dev and Prod Finance configurations

3.2 Setting up source files for version management

As with the rest of the tutorials, all metadata will be harvested from files, rather than from live systems, as would normally be done. This is so that the tutorial may be followed without having all the databases and other metadata source systems up and accessible.

In addition, as new versions of metadata are simulated, we will still import from the same file locations, but only after having updated the actual source files with the new files that are provided.

Now, the current Models\MetadataManagement\Finance folders are populated with the final state of all the source files. This, we must restore them to their "v1" state. You may do this by copying the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v1

into

C:\Temp\Models\MetadataManagement\Finance

and agree to the replacement of files. This action will update the source metadata that will be harvested in the rest of this chapter.

As we have two "parallel" configurations, one tracking Prod and one tracking Dev, we also need to create a place where one can place the Dev metadata sources file. So, you should do this by copying the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v1

into

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\Dev

This action will create the source metadata that will be harvested in the rest of this chapter for Dev models.

Now, we will look at the implications of version and configuration management. The environment is all set. Let's take a look at what we have.

3.3 Looking at the Finance Prod System Metadata

In Meta Integration® Metadata Management (MIMM), expand the Repository Tree in the left hand pane to the [folder] at the path:

1 - Operational Data Stores

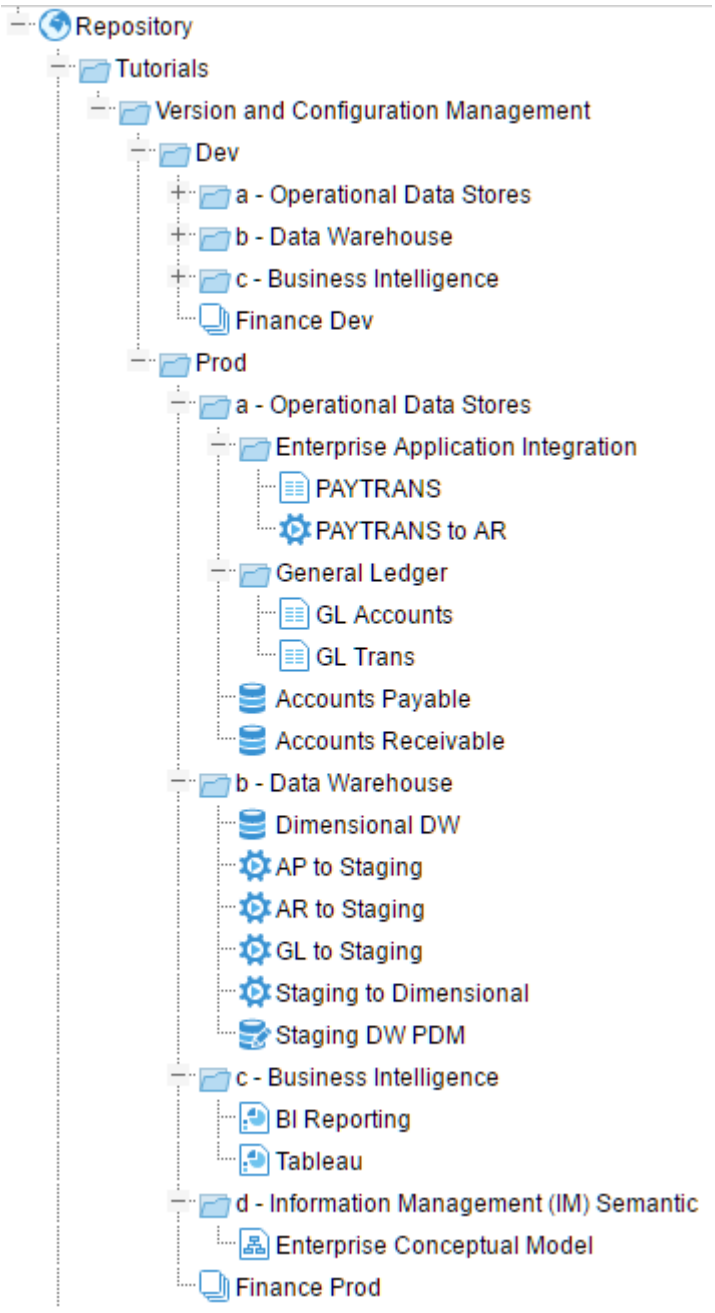


Figure 5 - Repository tree expanded

As seen above, the two configurations are quite similar, distinguished only by some nicer formatting of the architecture diagram for Prod. However, each of the models refers to a different file location between Prod and Dev. This is fairly common in actual

implementations, where file based bridges are used. As one would expect, the source metadata (ETL, BI, databases, etc.) are in different systems and locations.

3.3.1 Harvesting Information Management Metadata

This tutorial has one major difference in terms of information architecture from the Core Concepts architecture. The Information Management (IM) Semantic consists of one conceptual/semantic model, the Enterprise Conceptual.

This model is already harvested for you in the Prod section. It is a little different type of model than we have seen before (it is based upon a Data Modeling tool, erwin Data Modeler) and thus is already documented with logical names, descriptions, diagrams, etc.

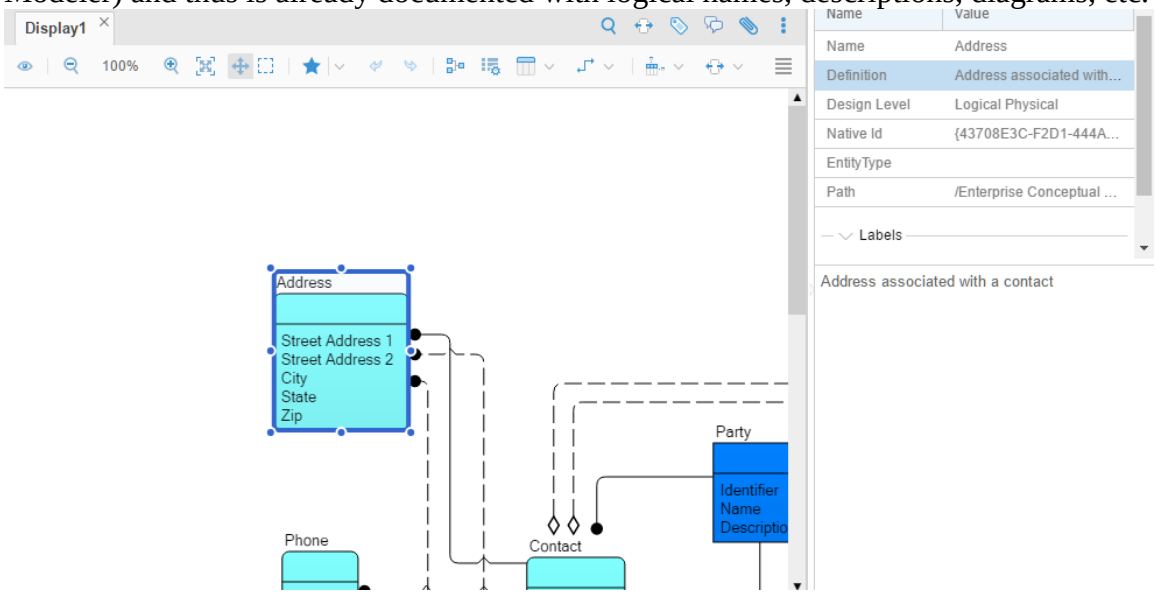


Figure 6 - Enterprise Conceptual Model

This model will be used as a conceptual model which contains high-level and over-arching data administration information for the version management use cases. In many cases, this same role may be accomplished using a glossary.

3.3.1.1.1 Diagram Visualization

If the model was harvested from a tool which provides diagram information, one may select a diagram in the browse tree and select Open, or select an object which would be in a diagram (e.g., a table or column) and select Show in Diagram.

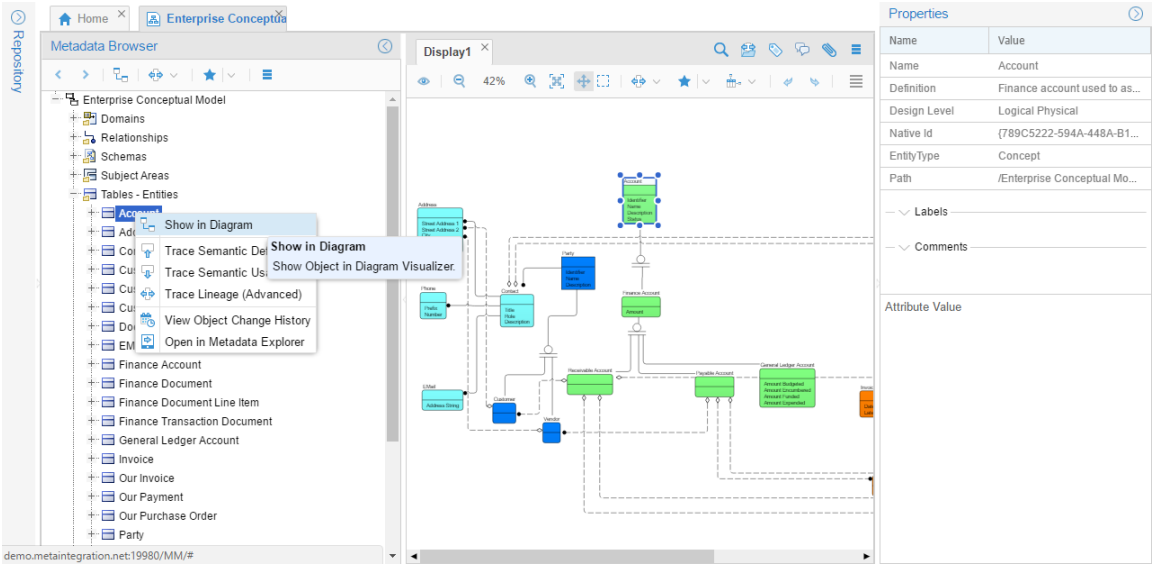


Figure 7 - Information Management semantic model

Note that the positioning, color, etc. are all faithfully captured by the bridge and rendered by the *Diagram Visualizer*. Also, various diagrams may be presented, both physical and logical.

3.4 Creating the Semantic Mapping

Note, if you open the two configurations, only the Enterprise Conceptual Model remains unconnected in the architecture. This is because it is not connected via a data flow, by a *semantic mapping*.

Semantic relationships between models are generally use to associate elements with similar meaning. Often-times this is a reflection of the fact that one model is derived from another, such as a design layer mapped semantically to the physical or run-time model.

In this case, the Enterprise Conceptual Model is a highly conceptual version of the data elements managed by the Finance system. However, the physical models and databases (e.g., the Staging DW PDM or Dimensional DW models) were NOT derived from this conceptual model. Thus, there are no relationships to harvest from a source tool which could be used to populate a semantic mapping.

Instead, we will create a new semantic mapping named ECM to Staging. It is a semantic mapping between this conceptual model of a generalized finance type system and the actual staging database model. We will manually map them using the Semantic Mapper.

Create a new semantic mapping in the 4 – Information Management (IM) Semantic folder:

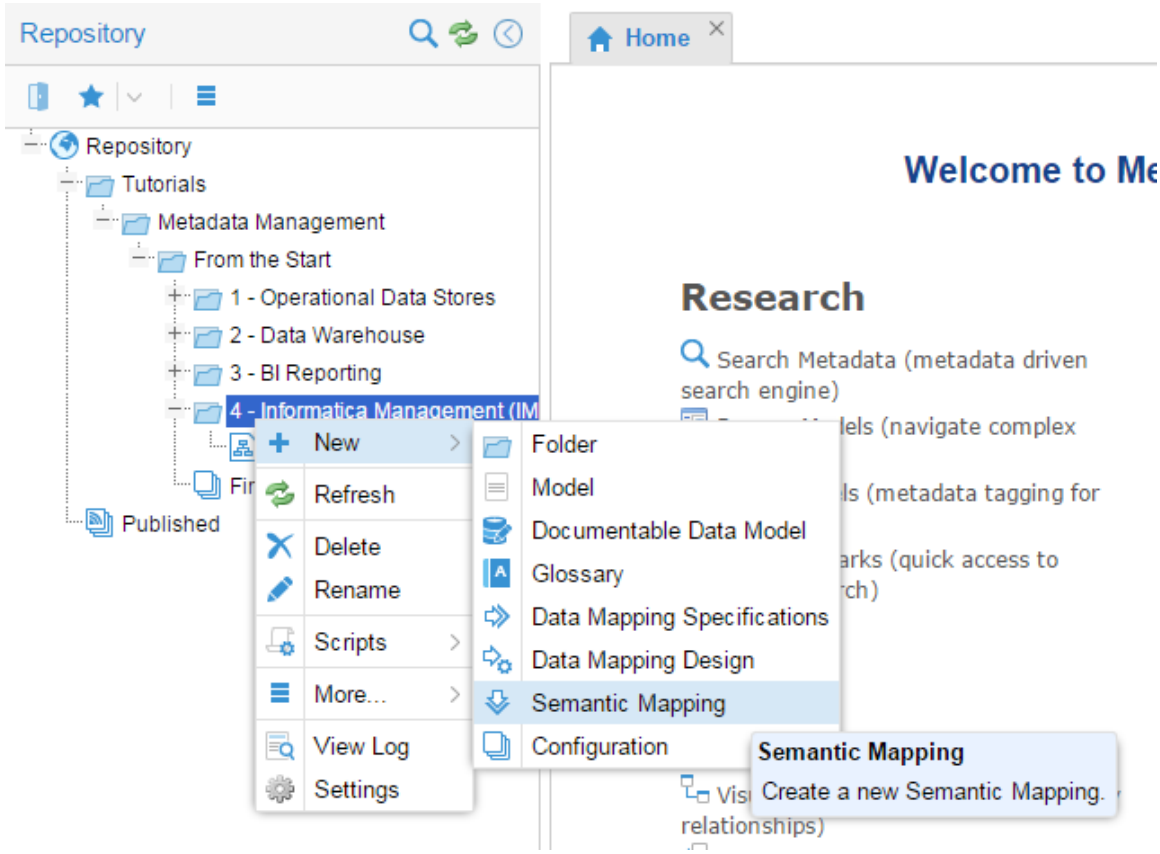


Figure 8 - Create semantic mapping

Named ECM to Staging. The mapping will then be opened for editing.

Now, drag the Enterprise Conceptual model to left or source of the mapping as shown:

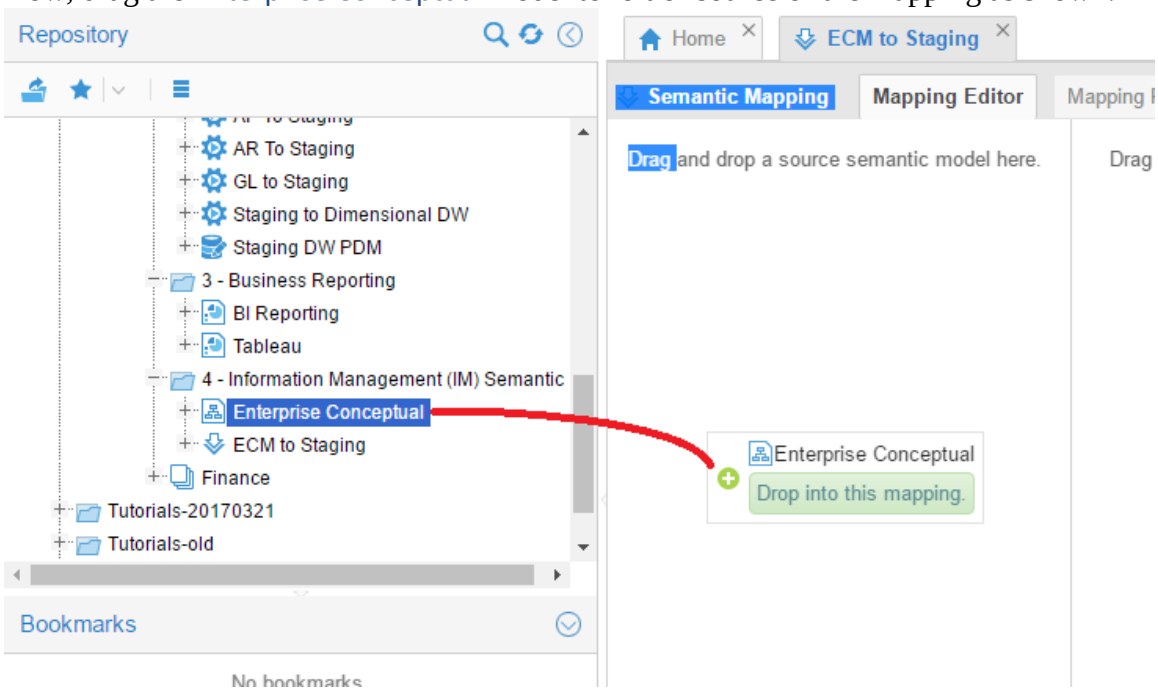


Figure 9 - Adding ECM as source of the semantic mapping

Then drag the Staging DW PDM model to the to the right or destination side of the mapping as shown:

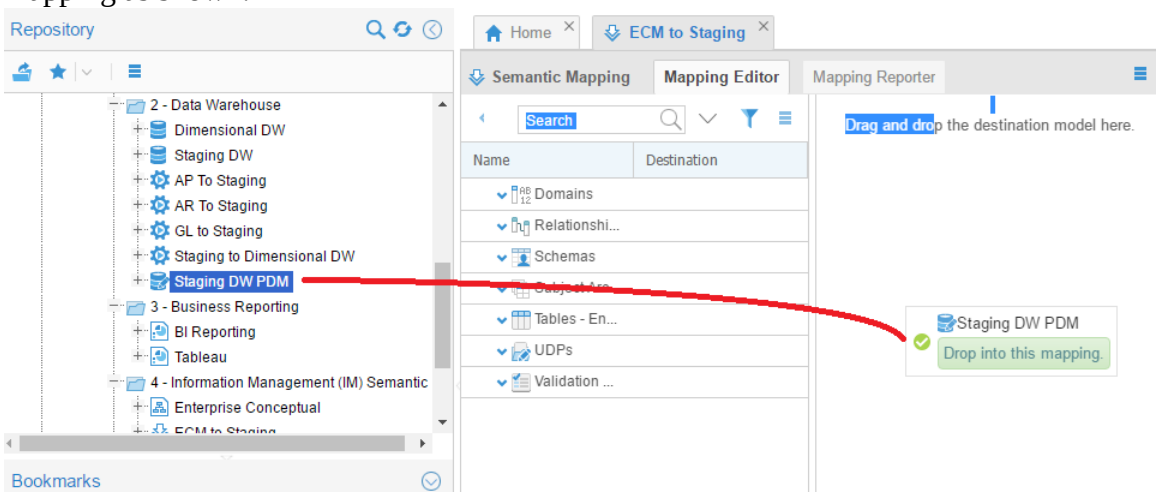


Figure 10 - Reconnecting Staging DW as the target of the semantic mapping

Minimize the Repository and Properties panels so we have screen real estate.

Now, let's map a couple of conceptual data elements to the data elements in the Staging DW model. Navigate to:

Tables - Entities ➔ Finance Document ➔ Column - Attributes

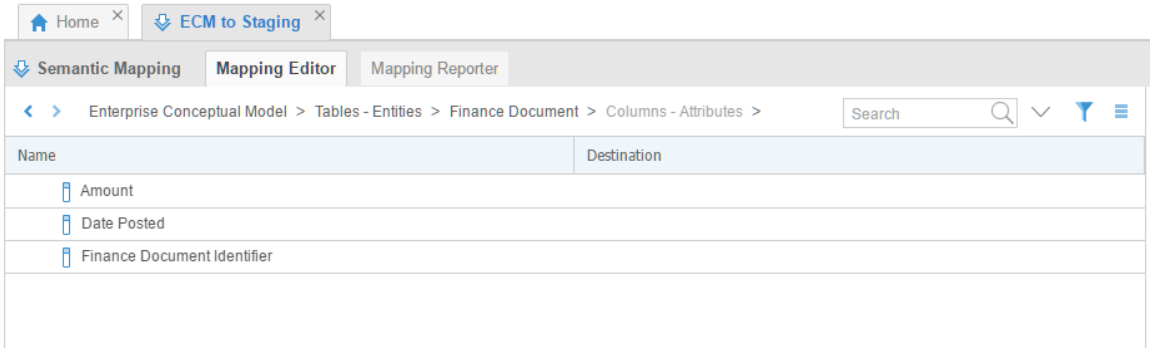


Figure 11 - Navigate to location in ECM

Then, on the right hand side, enter "*amount*" in the search criteria and click on the search icon (the prefix wildcard search only works for Oracle based repository databases, not for Microsoft SQL Server or PostgeSQL based ones) :

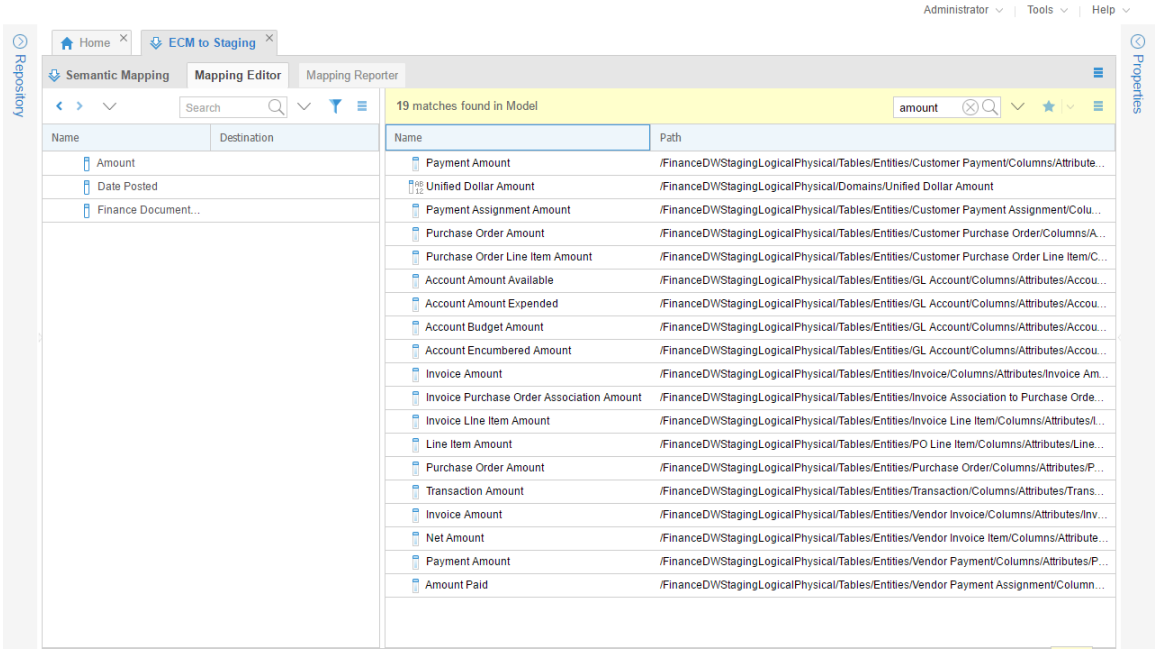


Figure 12 - Search for target data elements

Here we are seeing all of the target data elements in the Staging DW model that have the same meaning as the Finance Document.Amount attribute in the Enterprise Conceptual Model. Now, not all are the same. In fact, we should note that there is another Finance Document Line Item entity with an Amount attribute as well. Thus, we should only map this Amount attribute to those finance documents (like Invoice or Purchase Order) and not their line items.

So, drag the Amount attribute on the left to the Payment Amount attribute on the right:

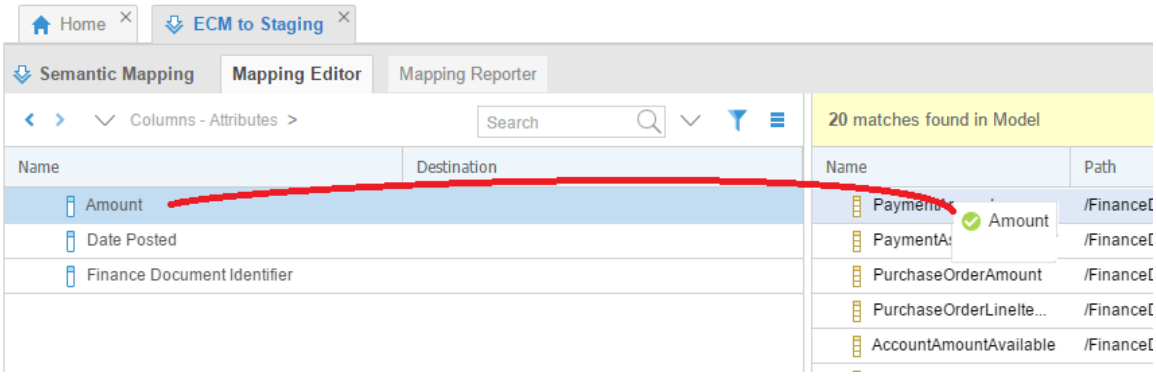


Figure 13 - Drag and drop semantic mapping

Now, you can see the link icon next to each showing what is mapped.

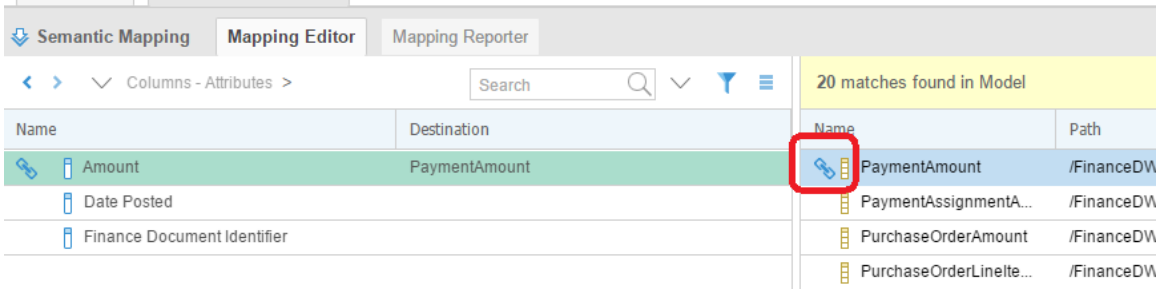


Figure 14 - Link icon for a map

Next, map to the following attributes on the right:

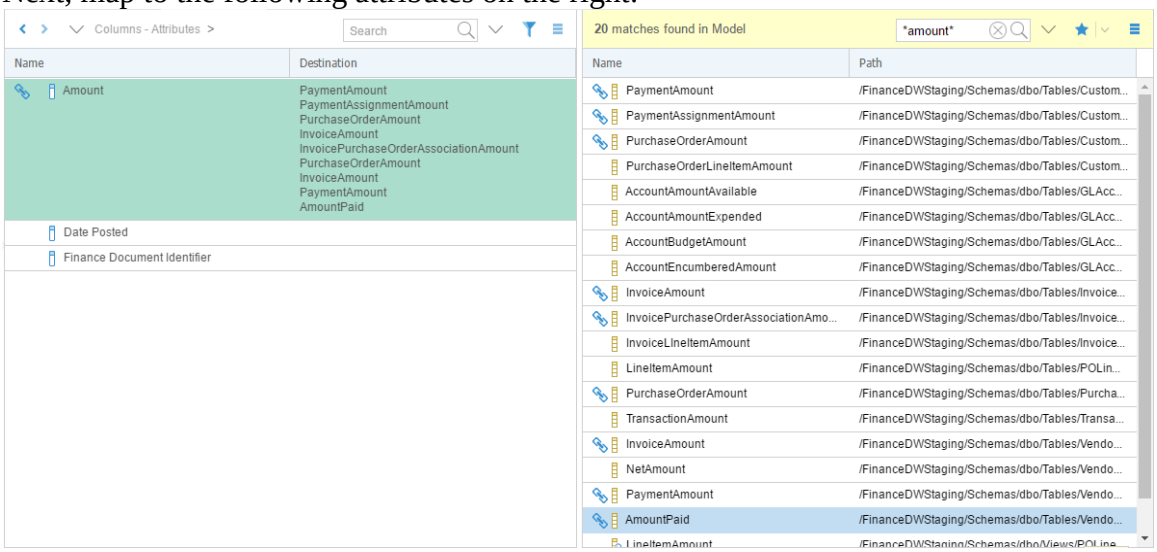


Figure 15 - Semantically mapped attributes

In addition, navigate to Finance Document Line Item, (two levels up on the left then open that entity and then Columns - Attributes) and map Amount to the following:

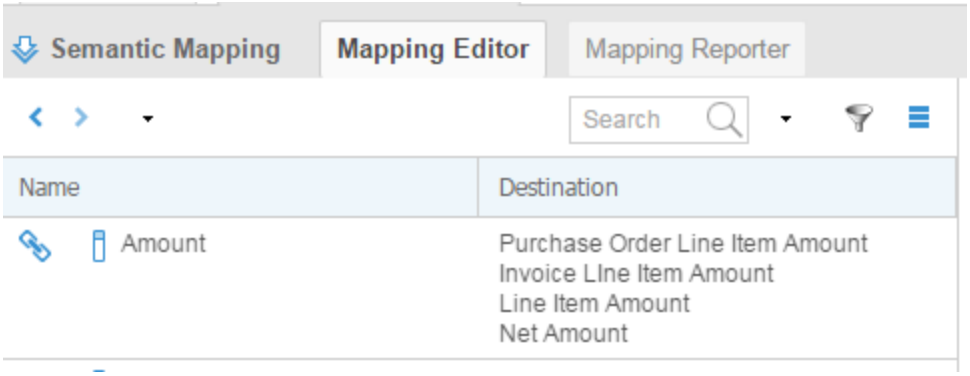


Figure 16 - Semantically mapped attributes

Finally, let's map the amount columns General Ledger Account to the equivalent columns in the GL Account table.

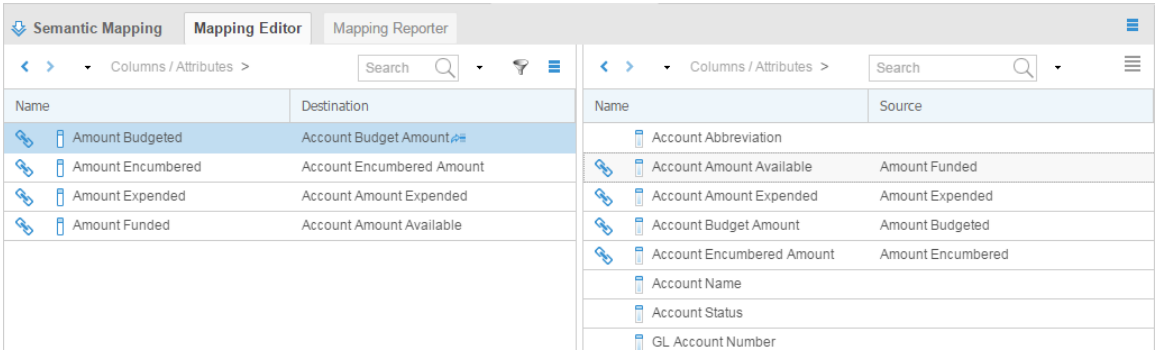


Figure 17 - General Ledger amount mappings

Finally, we want to populate the configuration with the semantic model and mapping we just constructed so that we may analyze them.

Close the semantic mapping.

Open the Repository panel, open the Finance Prod configuration and drag and drop the Enterprise Conceptual model into the configuration and note that the semantic mapping is automatically added.

Now, click the Build icon again.

Switch to the Architecture Diagram tab and then click on Edit. Do not use the Auto Layout, as that would layout everything and all we want to do is move the new model and mapping to a good location. Please do so like this:

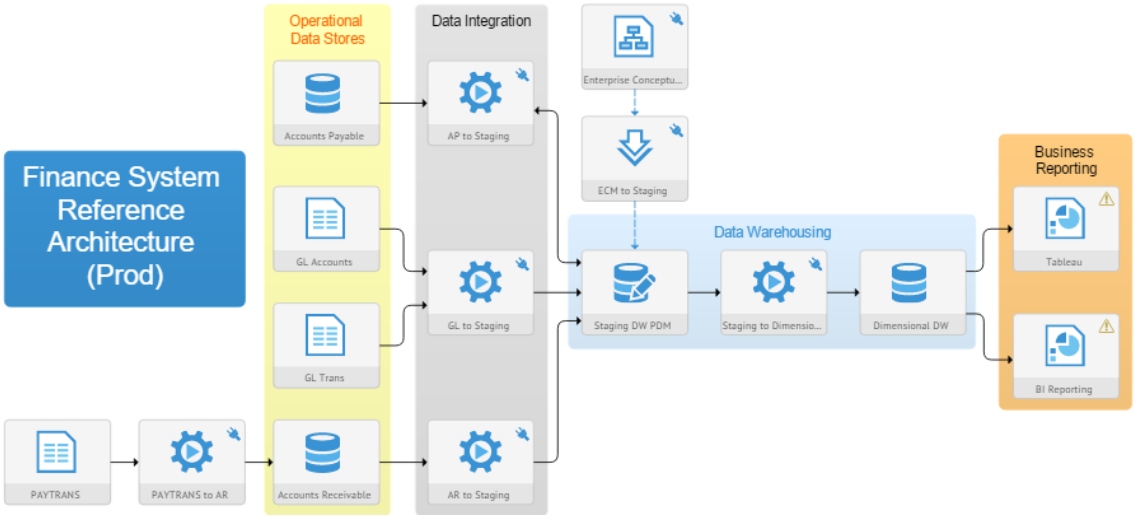


Figure 18 - Final diagram

Click on Edit again to save the layout.

Finally, be sure to *Publish*

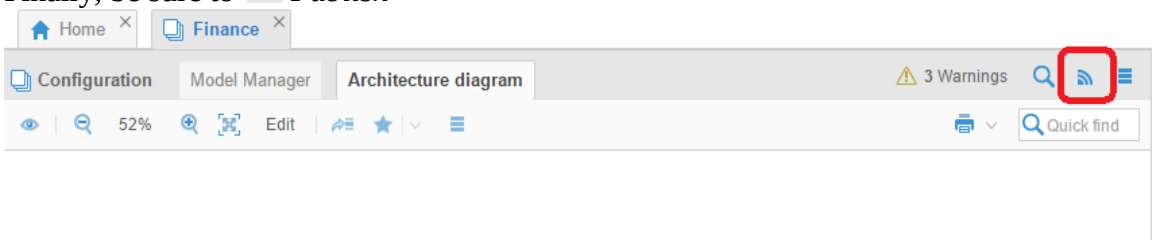


Figure 19 - Publish icon

the configuration so that business users may see it.

3.5 Explorer UI from Administrative Users

One may also access the Explorer UI even though signed in as a user associated with a rolename which is not defined as a business user. In this case, one simply goes to Tools › Explorer UI and then sign in again. You are then prompted to indicate your configuration, as the Metadata Explorer UI always zeros in on a specific configuration for analysis.

Finally, when one opens a configuration, one may choose the Open in Metadata Explorer option and go the Metadata Explorer UI for that particular configuration.

4 Change Management

This section will detail how one uses the Meta Integration® Metadata Management (MIMM) user interface to perform life-cycle based impact and lineage analysis on the tutorial metadata that you have placed in the Meta Integration® Metadata Management (MIMM).

Each time one harvests new *versions* of one or more models, new versions of that configuration which they are a member of must be created. Once several versions of models and mappings are collected, repository analysis could become quite complicated. Merely the simple case of the changed element in the PAYTRANS XSD message could create the need for updated versions of the ETL to the Account Receivable database, that database model as well, and then subsequent ETL and databases *impacted* by that change.

As a result of the new versions of configurations, searches in the repository and lineage analysis will report on, and even the repository structure contains, several nearly identical models, identically named and related data elements, and multiple data lineage and impact analysis traces. While it is important for this information to be properly captured and maintained, it should be possible to concentrate only on specific *version of a configuration* of related versions of objects in the Repository.

4.1 Notification of changes

In this example, there is an unexpected change to the PAYTRANS message in which the PTAMT will be extended to a Decimal(10,2), from a Decimal(8,2), field size. When a change like this occurs, ETL/DI process, database field that are too small, etc., can all cause trouble.

Meta Integration® Metadata Management (MIMM) allows one to configure automatic notification when a change is detected on harvest. To do so on the PAYTRANS (in the Prod section, because this is an unexpected change by an external organization) model, right-click and select Settings:

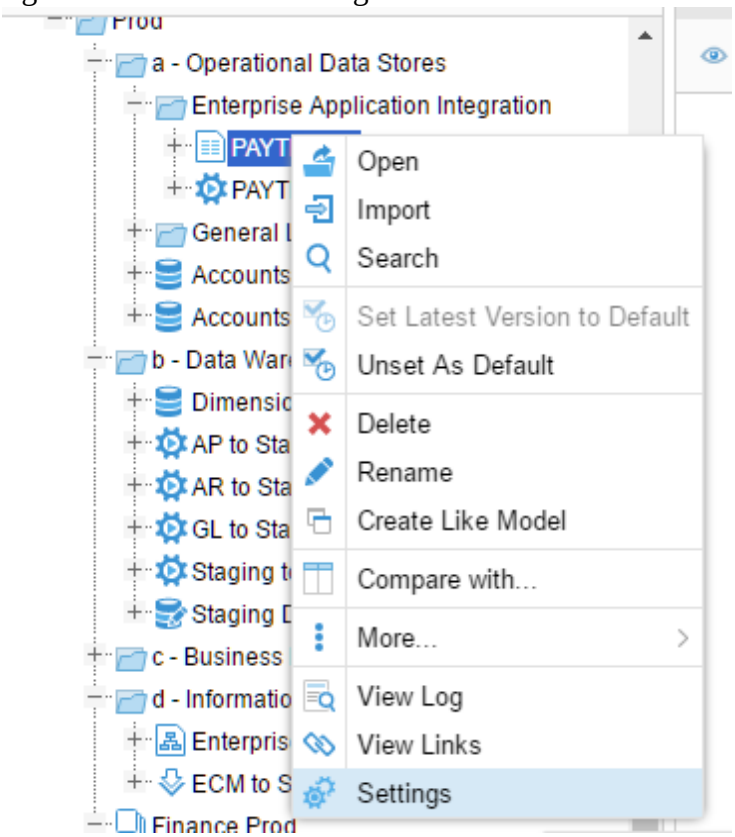


Figure 20 - Settings

and check the Send Import Notification checkbox:

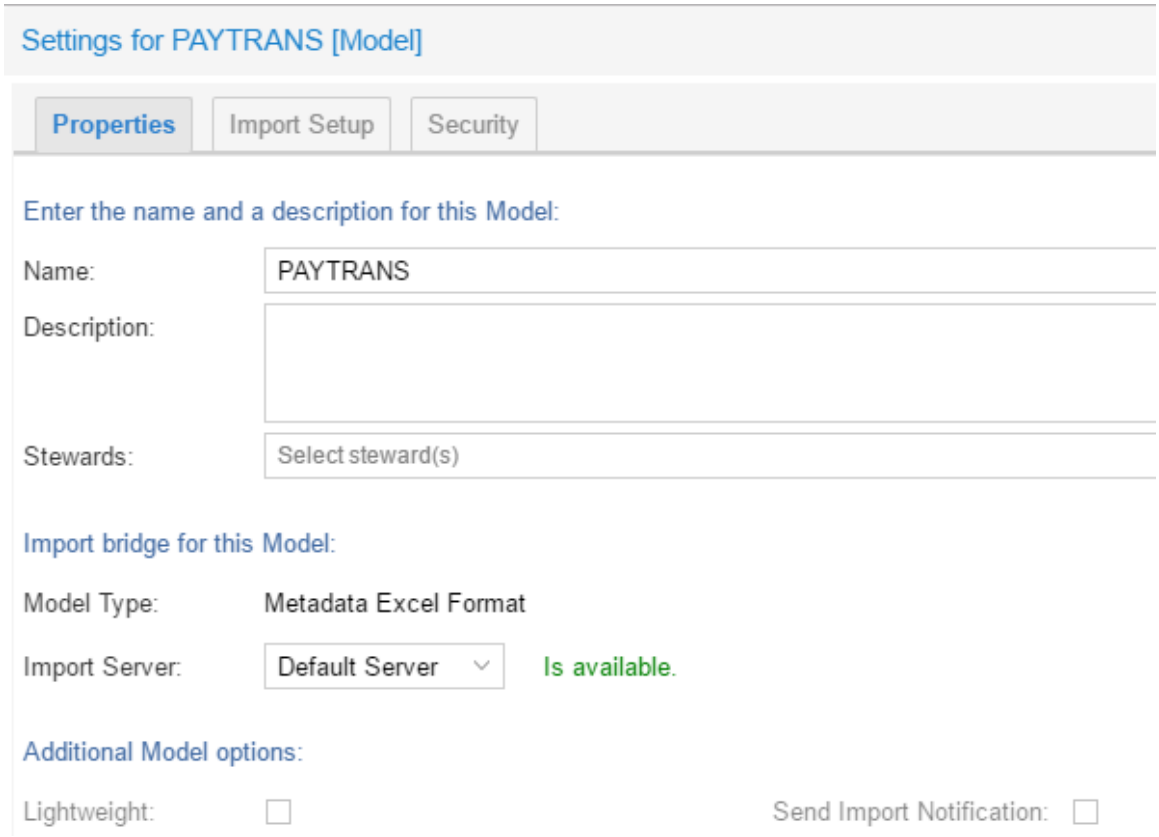
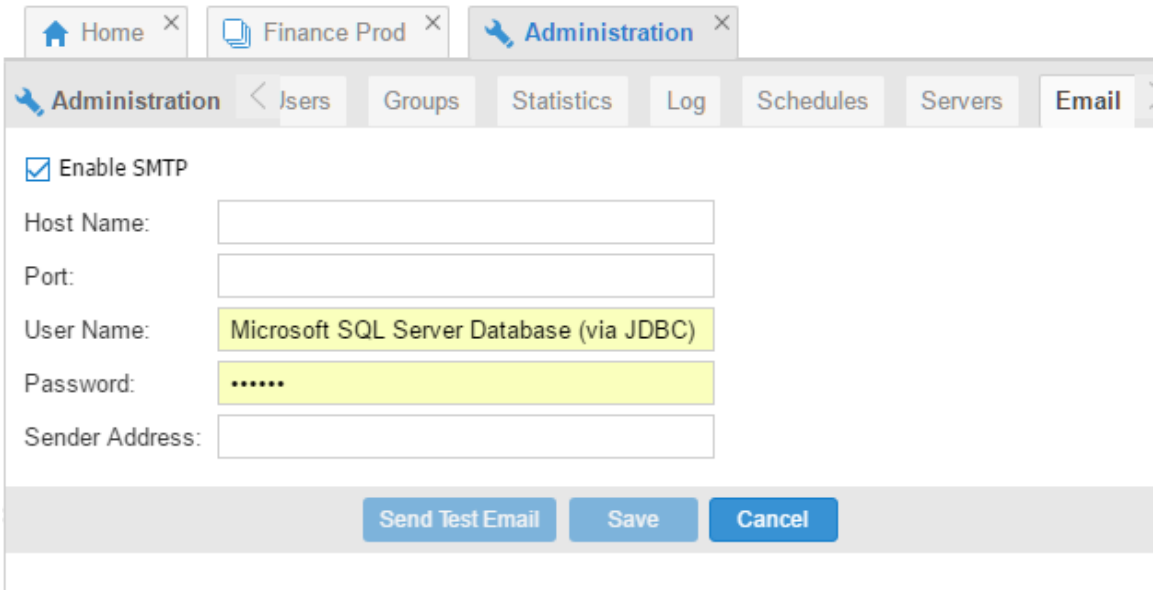


Figure 21 - Setting dialog

Unfortunately, the box is not enabled because the e-mail notification is not configured.

To do so, go to Tools > Administration > Email, check the Enable SMTP checkbox and then enter the appropriate parameters for that server. You must have an SMTP server available to enable e-mail notification.



In addition, we need an e-mail address to send the notification to. Notifications are sent to anyone who is a Steward of the model in question. Thus, we need to provide an e-mail address for some user and then assign them as steward of the PAYTRANS model.

Go to Tools > Administration > Users and click on the Bob user. Then enter an e-mails address for that user, check the Steward checkbox so he may be selected as a steward, and click on the Save button.

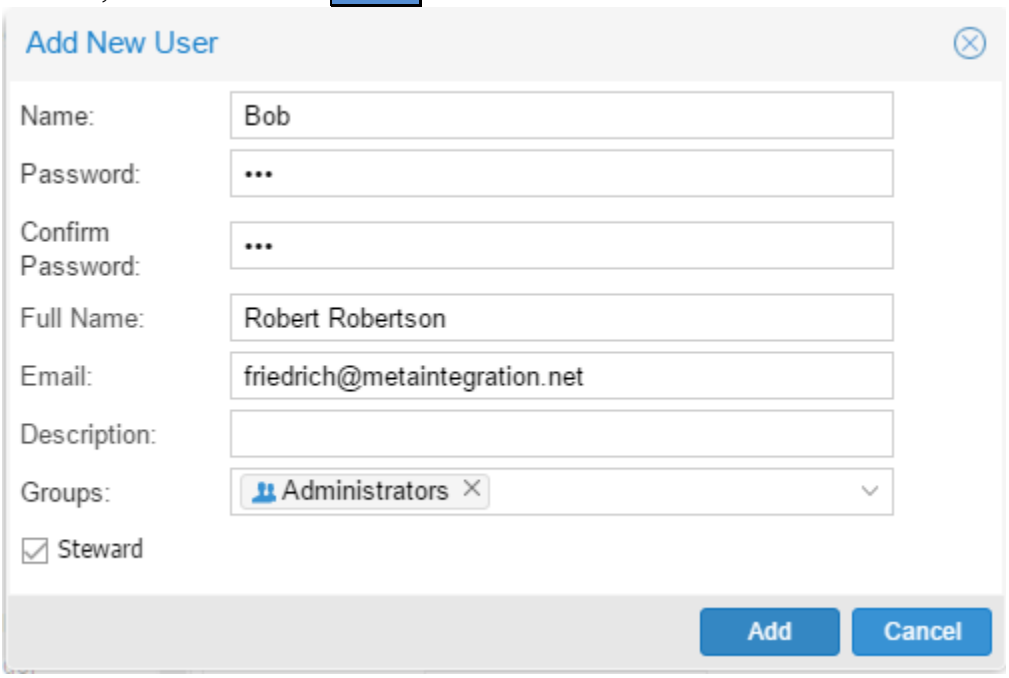


Figure 22 - Bob user

Now, go back to the Settings dialog, and check the Send Import Notification checkbox and set Administrator as Steward for PAYTRANS:

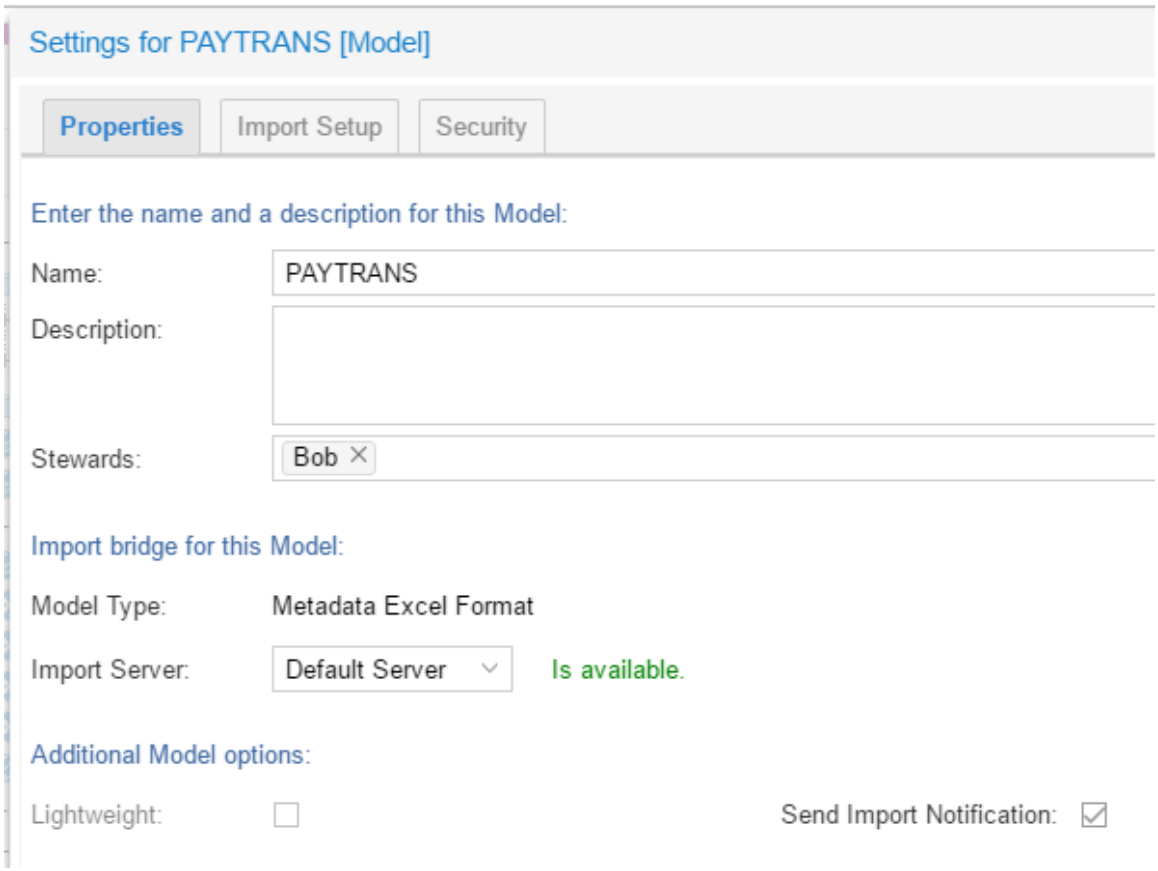


Figure 23 - Settings

We should now be notified.

In this first use case, we will note what happens when the flat file that we read from external sources (PAYTRANS) is changed in format. To see what is changed, go to:

C:\Temp\Models\MetadataManagement\Finance\MicrosoftExcel\

and open the two files:

- PAYTRANS-v1.xlsx
- PAYTRANS-v2.xlsx

Note, the change made is that the PTAMT field as imported earlier has a data type of decimal(8,2):

Models-v4.1 How to use

Model / Catalog		Schema	Entity / Table / Record / View	Attribute / Column / Field			
Name	Type	Name	Name	Name	Data Type	Length	Scale
PAYTRANS	FILE	PAYTRANS	PAYTRANS	PTCID	varchar	12	0
				PTID	varchar	12	0
				PTDT	varchar	14	0
				PTAMT	decimal	8	2
				PTINVNR	varchar	20	0
				PTINVLINR	varchar	2	0
				PTCHKNR	varchar	20	0
				PTPYTYP	varchar	2	0

Figure 24 - Version One of PAYTRANS

And this same PTAMT field in PAYTRANS-v2.xlsx to be imported here has a data type of decimal(10,2):

Models-v4.1 How to use

Model / Catalog		Schema	Entity / Table / Record / View	Attribute / Column / Field			
Name	Type	Name	Name	Name	Data Type	Length	Scale
PAYTRANS	FILE	PAYTRANS	PAYTRANS	PTCID	varchar	12	0
				PTID	varchar	12	0
				PTDT	varchar	14	0
				PTAMT	decimal	10	2
				PTINVNR	varchar	20	0
				PTINVLINR	varchar	2	0
				PTCHKNR	varchar	20	0
				PTPYTYP	varchar	2	0

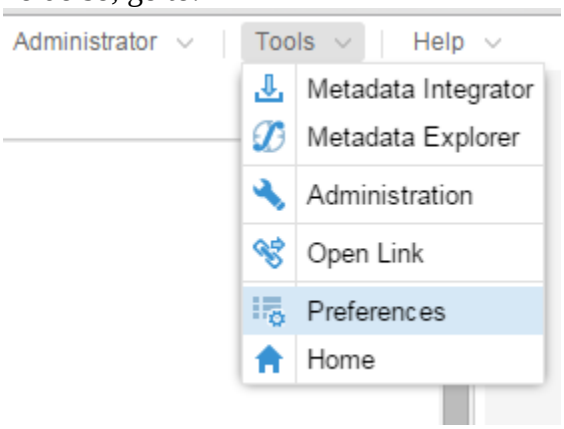
Figure 25 - Version 2 of PAYTRANS

In the Meta Integration installation directory on the Application Server

- Create a backup copy of C:\Temp\Models\MetadataManagement\
- Copy the contents of C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v2 into C:\Temp\Models\MetadataManagement\Finance\ and agree to the replacement of files (PAYTRANS and ETL models).

As we are going to be importing many of the models again, we will want to see these new versions in the Repository tree pane. However, up to this point we have been in *versionless* mode. Such is the default mode of the Repository, and one must explicitly change it at this point in order to see the new versions of the models we imported.

To do so, go to:



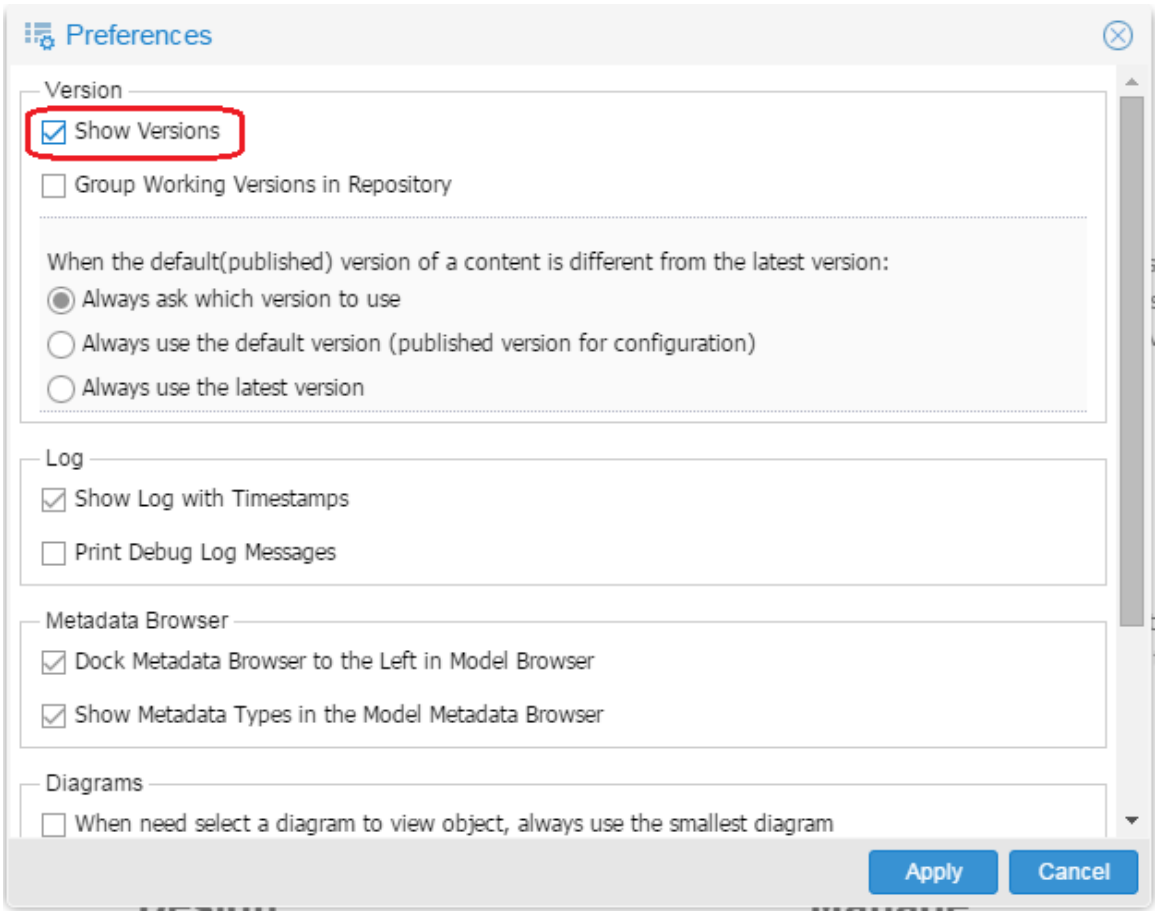


Figure 26 - Tools->Administration->Preferences

and select the check box for Show Versions.

Now, let us begin to harvest new versions of the existing models. In this scenario, a new version of the PAYTRANS flat file message (modeled in Excel for these purposes) is proposed. Thus, we should harvest a new version of this model. In order to do so, we need to point to another source file when harvesting, or replace the one in the original location with the updated version of that file. For this first example we will do the former

Navigate to the PAYTRANS model content in the Repository Tree and right-click and select Ímport.

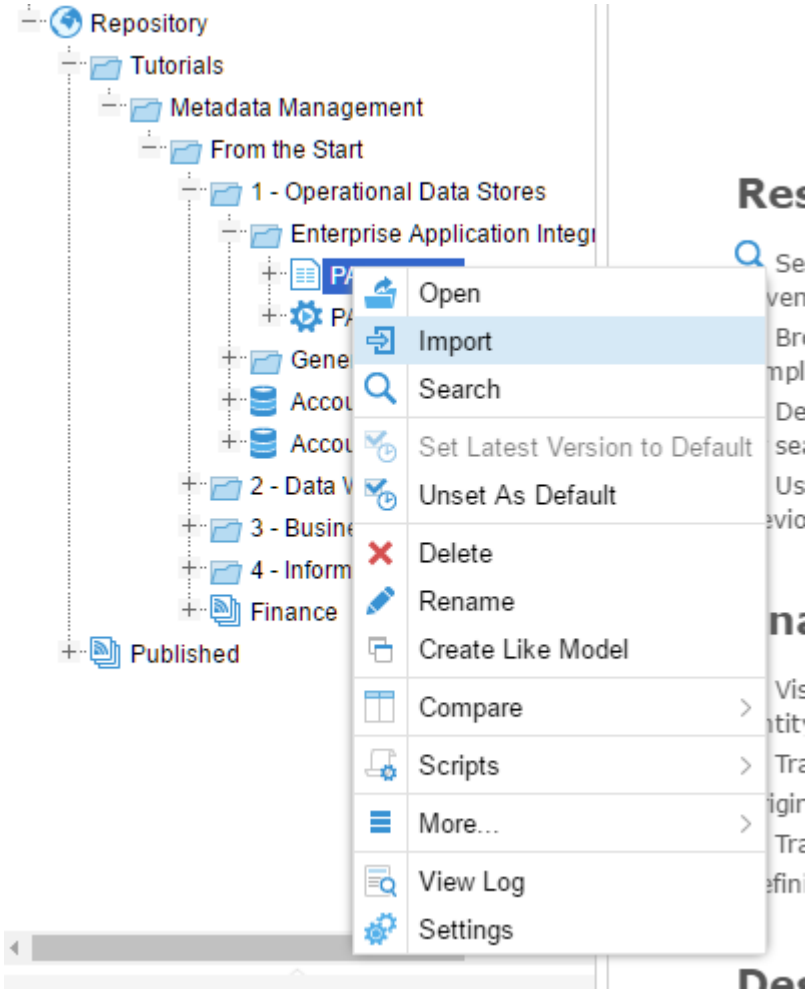


Figure 27 - Import the PAYTRANS Model

And you have the import dialog.

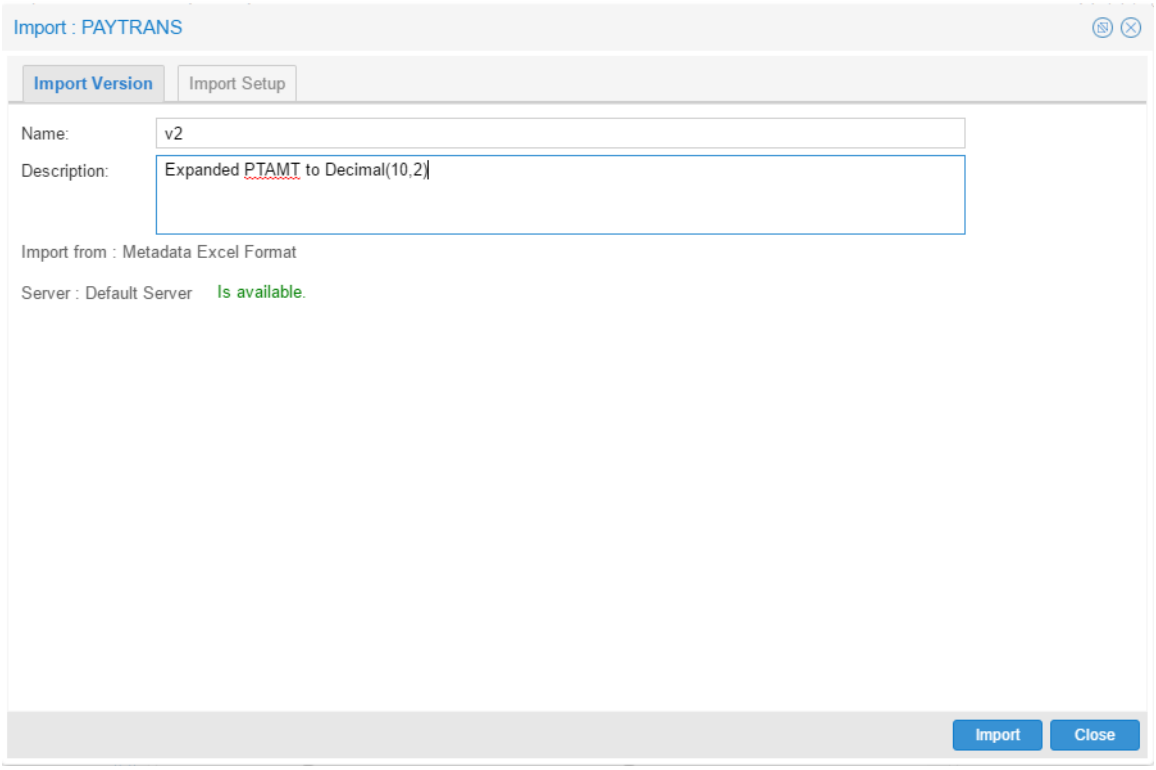


Figure 28 - Import dialog for PAYTRANS model content

Enter "v2" in the Name field and "Expanded PTAMT to Decimal(10,2)" in the Description field.

Go to the Import Setup tab.

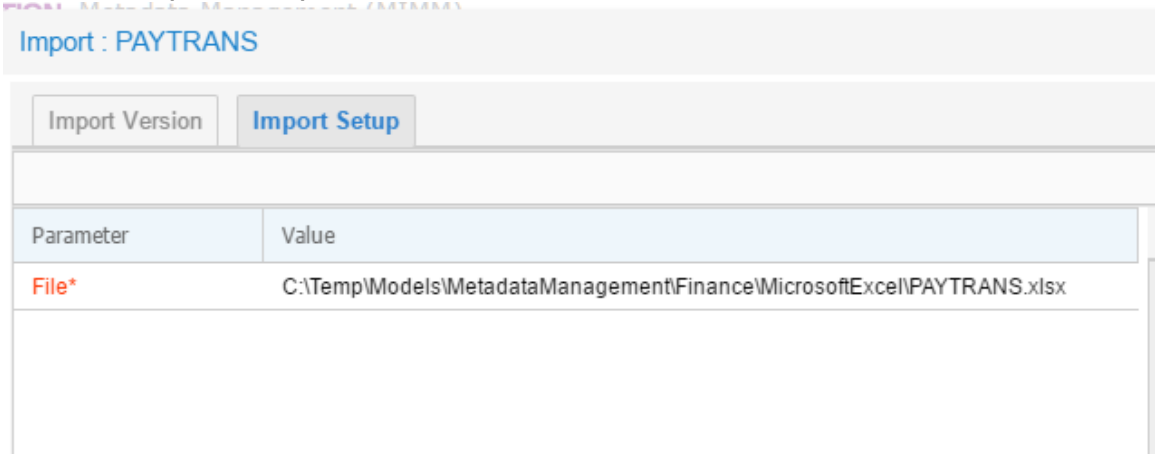


Figure 29 - Import Setup tab

Note the File* parameter is still pointing to the file named PAYTRANS.xlsx. However, you have already copied the newer file to this location, so simply importing with the same location will actually obtain version 2.

Now click on the Import button and allow the import to occur. Remember, this will import the new version of the model we copied earlier even though it refers to the same location. When completed successfully, open the model content version and review. Rename the model version that you just created to "v2", by clicking on the version of the model in the Repository panel and then editing the Name property in the Properties panel.

It should look like the following:

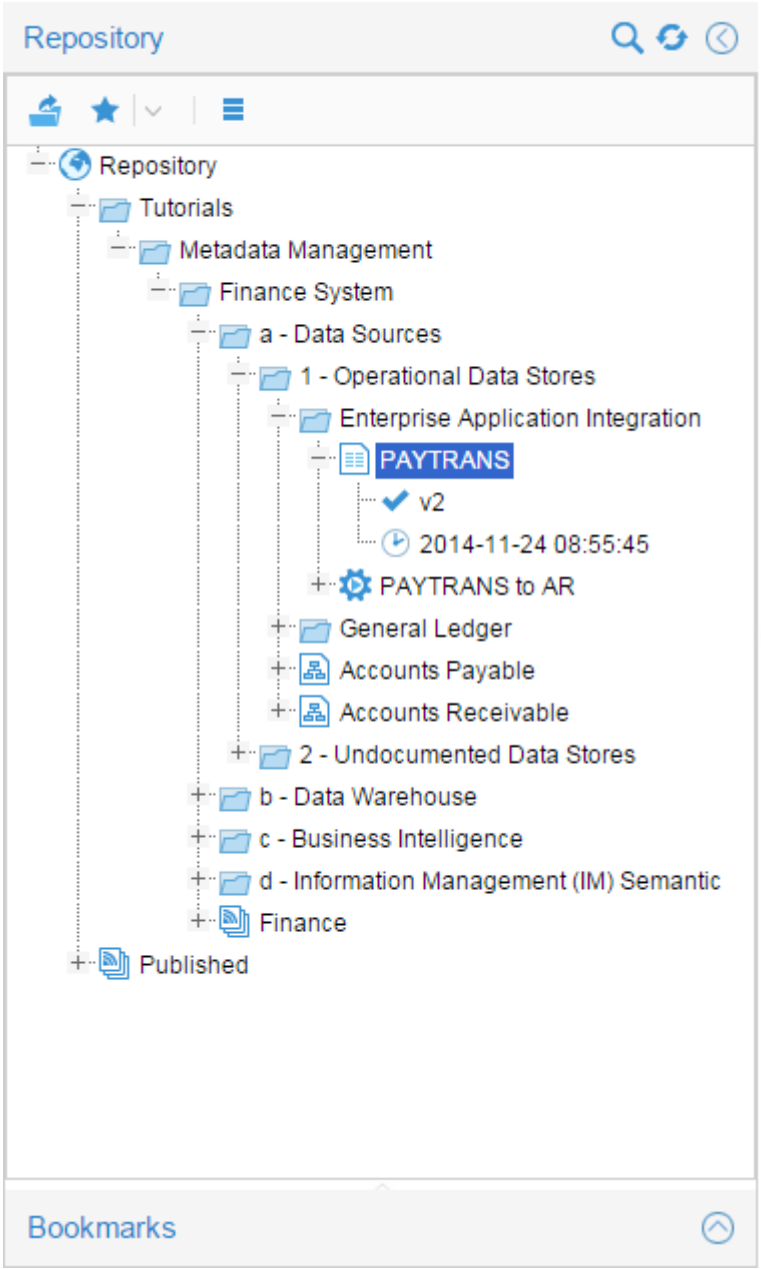


Figure 30 - PAYTRANS model content in the Model Browse tab

Note also, a notification e-mail was sent:

The content /Tutorials/Version and Configuration Management/Prod/a - Operational Data Stores/Enterprise Application Integration/PAYTRANS was imported at 2017-06-28 09:21:28.

There is no error in the import log.
A new version is created.
Summary of differences from the previous version:

DISPLAY NAME	ID	ADDED	REMOVED	CHANGED
Column - Attribute	276000003	0	0	1

Click on the hyperlink then takes you back to Meta Integration® Metadata Management (MIMM) to the location of the new model:

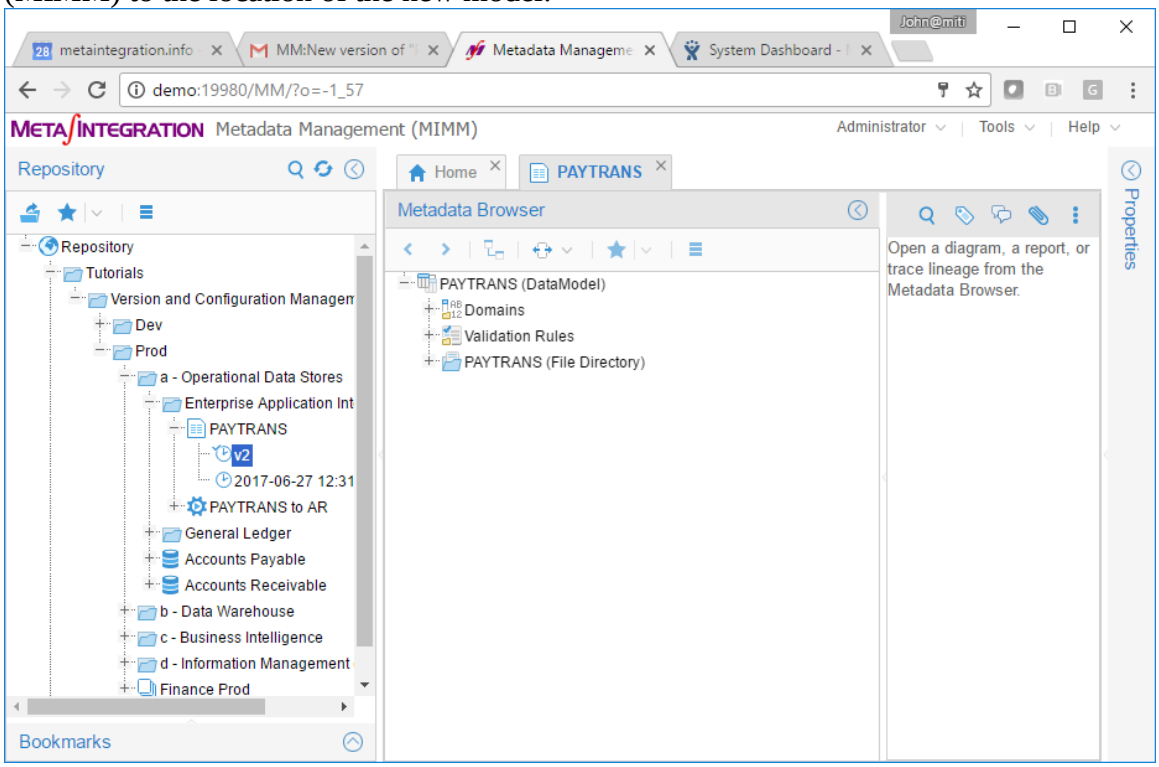


Figure 31 - Hyperlink result

Now that we know there is a change, it is important to determine what systems *downstream*, in terms of the data flow (*forward lineage*), are impacted.

4.2 Forward lineage (impact) analysis

The steps we will go through in this tutorial are:

- Harvest a new version of the model with the *harvest* features
- Create a new *strawman* version of the configuration (Finance) with the *migrate* feature
- Compare the strawman version with the prior published version
  - Compare updated model versions (PAYTRANS only at this point)
  - Determine impact of change
- Harvest updates to *impacted models* (data stores and processes)
- Migrate to a new *published version* of the configuration with new models
- Determine *completeness* of new stitchings

A generalized form of this process is pictured below, and is the basis of all good metadata management processes.

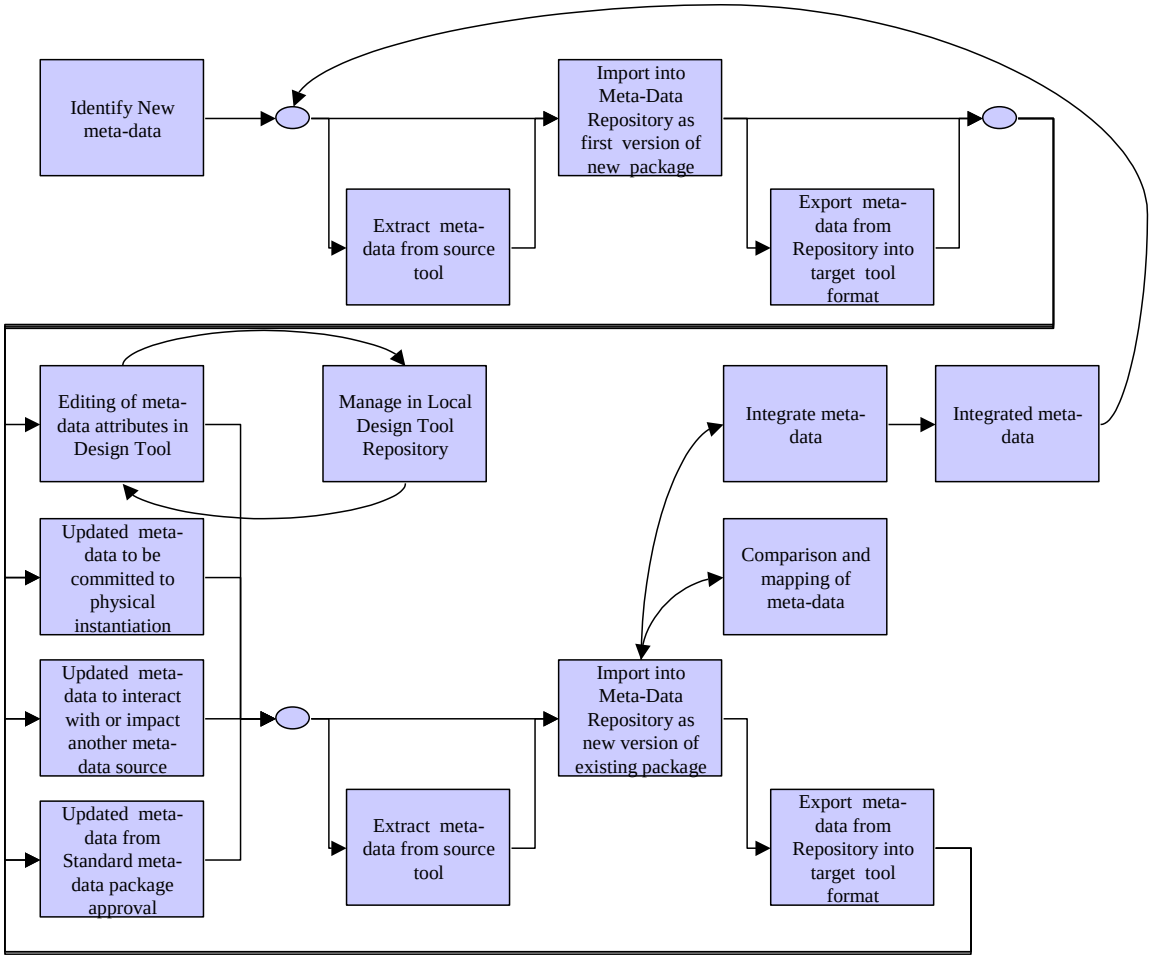


Figure 32 - Generalized metadata lifecycle process

4.2.1 Migrate to New Straw Man Configuration Version

With the new models that were imported, we will next produce a new s*traw man* configuration. This configuration will be used as a reference to determine changes from the previous (likely published) version of the configuration. It will also be the basis of a new published configuration once all changes to all impacted systems are made and they are ready to be published to the business analyst users.

Expand the Finance Prod configuration in the Repository panel, right-click on the version and select Create A Copy.

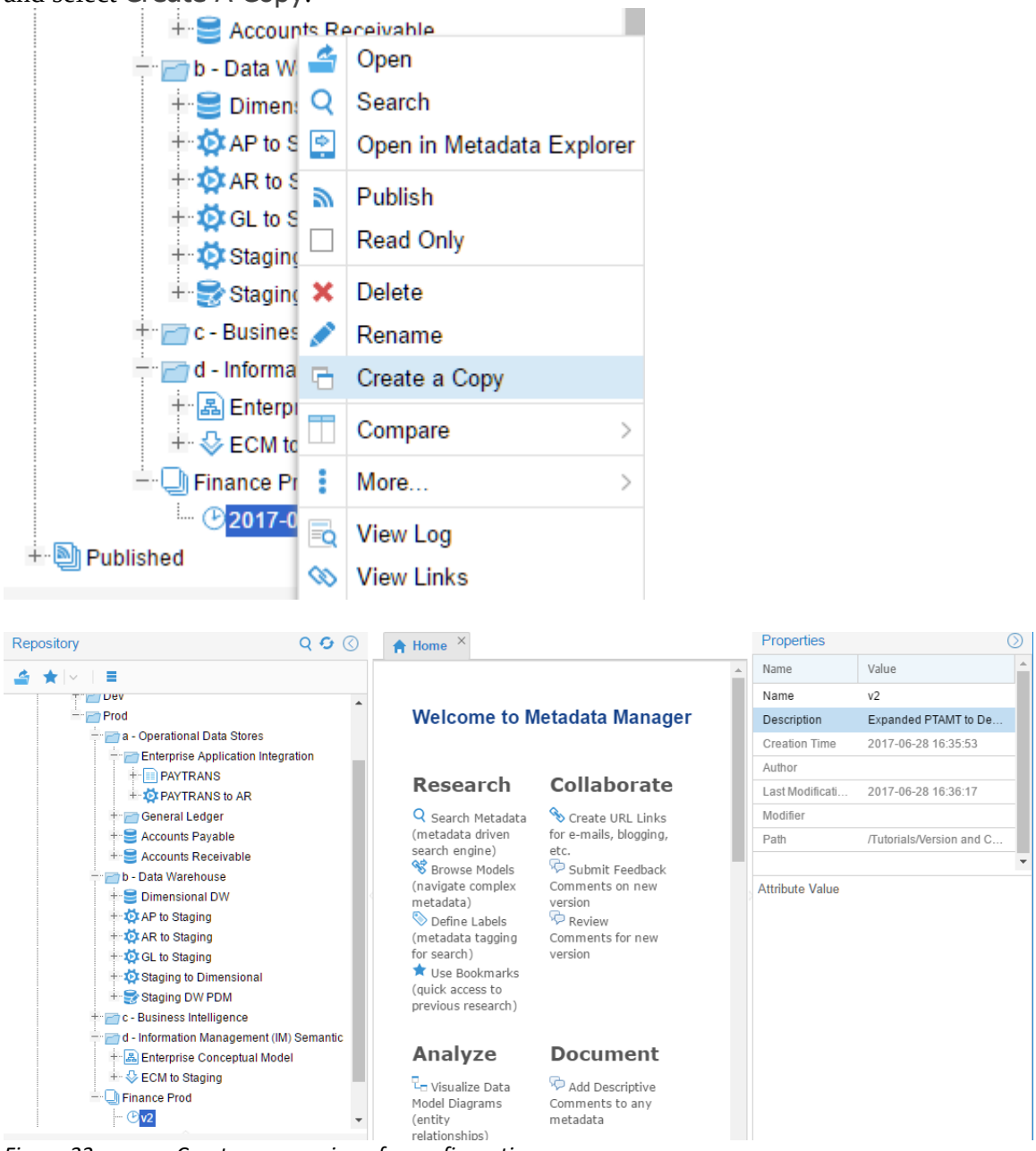


Figure 33 - Create new version of a configuration

Name this version “v2”. This new version should be described by "Expanded PTAMT to Decimal(10,2)". Be sure not to set the checkbox for Publish, as this is a straw man configuration to be used to determine what changed between versions. You do not want business users to see it just yet.

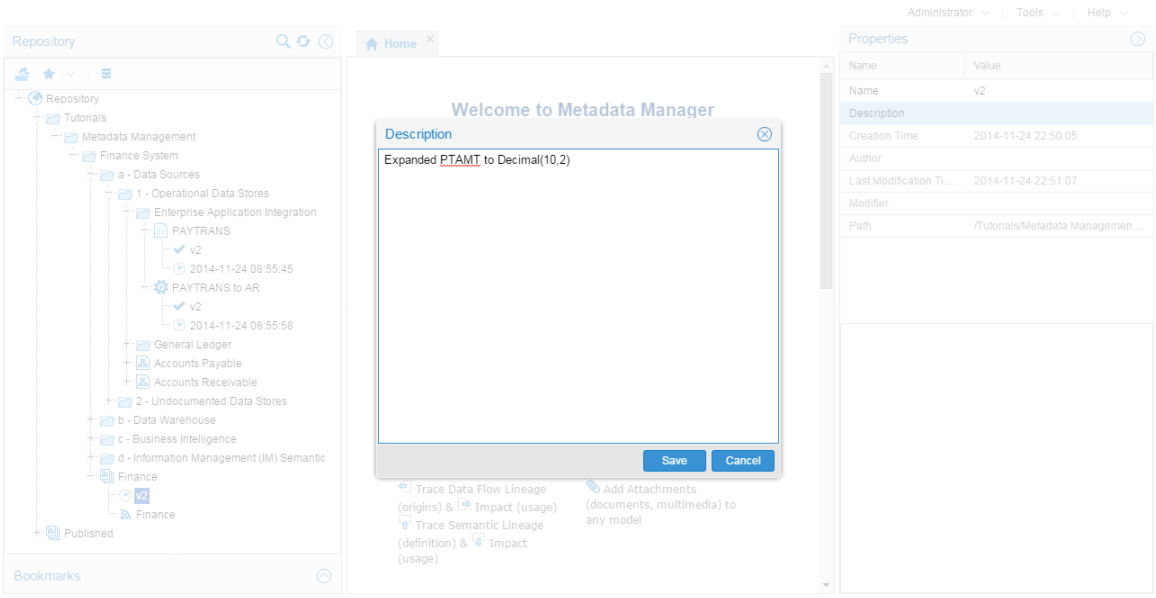


Figure 34 - Editing description for configuration version

Also, note that the new versions of the model (PAYTRANS) which was imported are not yet included in this configuation. You may see this fact by noting the version name next to the name of the model inside the configuration.

Meta Integration® Metadata Management (MIMM) will *migrate* these changes only if you explicitly ask for them to be migrated. This is because while harvesting took place, the option to Automatically Update was never specified for the Finance configuration. This was intentional for these exercises.

Automatic migration means that each time a contained model is updated (re-harvested for instance), the configuration then is updated to include the new version of that model (replacing the old one). This is not what we want here as we will want to see the differences caused by the changes by comparing different versions of the configuration.

First, we want to ensure that we save out the original version of the configuration. This is only important for authored content (e.g., Physical Data Models, glossaries, and mappings), i.e., content which may be edited in the same version. Existing versions of harvested content (imported from external sources) are not changed by re-harvesting. Thus, we only need to ensure that frozen versions (read-only historical versions) are created for authored content.

In this case we have two types of authored content. One type consists of all the semantic mappings. As these will not change in this chapter we will simply ignore them.

The other type is the Staging DW PDM. In this case, we will see updates harvested. Thus, we should save out a frozen version for the original configuration.

First, rename the original version of the Finance configuration (currently named Finance) to be Original.

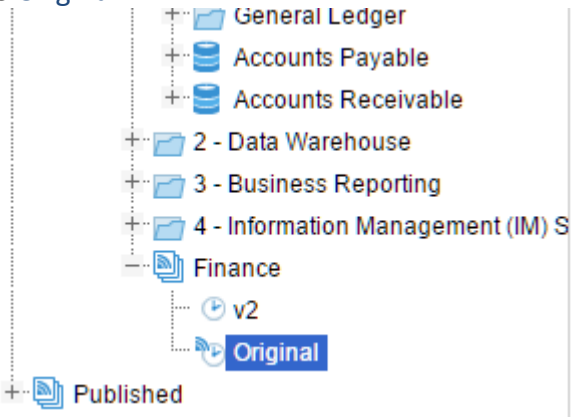


Figure 35 - Renamed version

Then, right-click on the Development version of the Staging DW PDM and select Create a Read-only Copy.

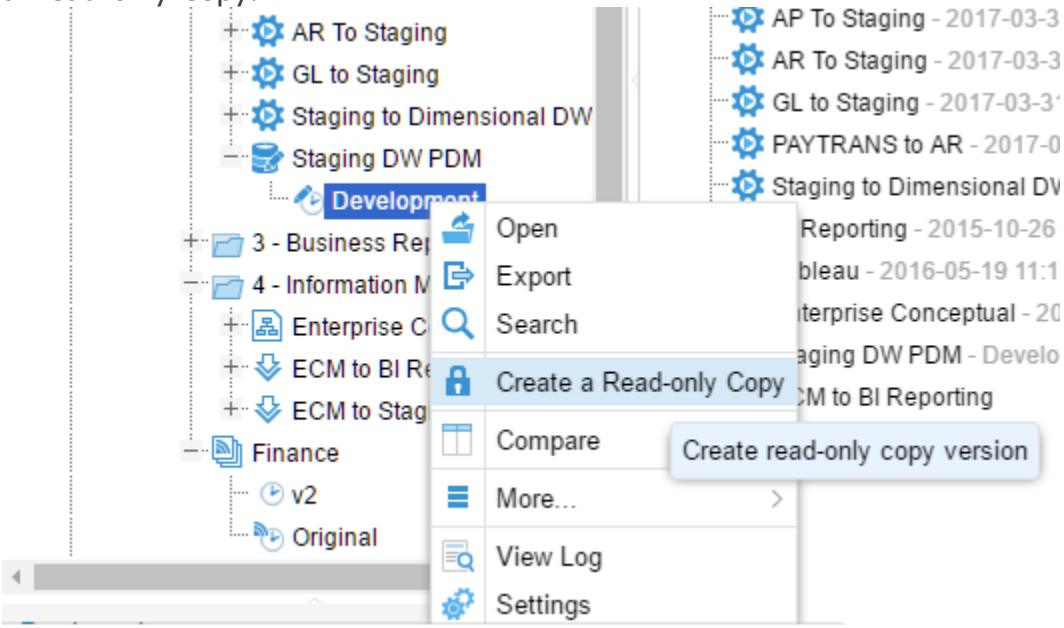


Figure 36 - Create a Read-only Copy

Rename this new version of Staging DW PDM to "Original" and give it a Description of "Original version of the model archived as of <today's date>".

Now, open the Original version of the Finance configuration and drag the Original version of the Stating DW PDM (that you just created and renamed) into the opened Finance configuration.

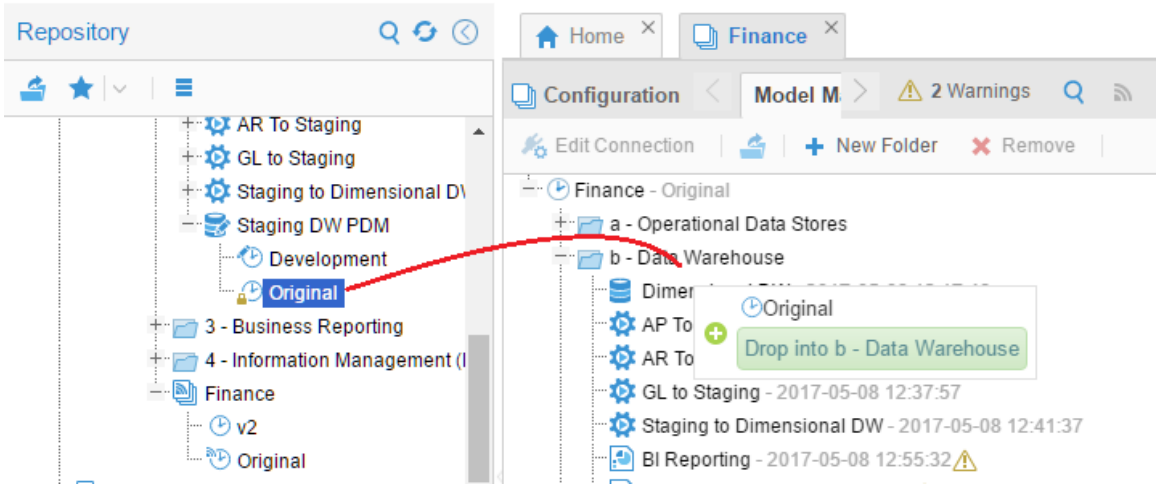


Figure 37 - Drag into the configuration

You will see the following dialog:

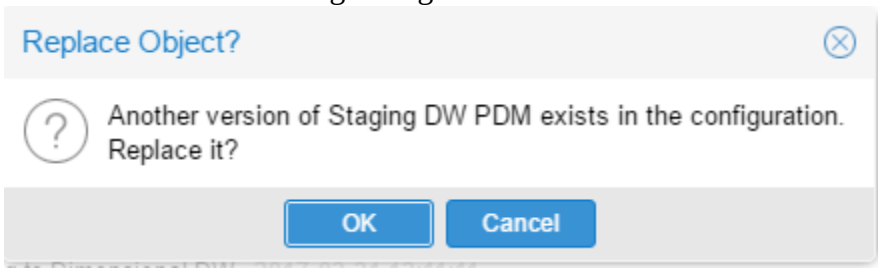


Figure 38 - Replace object

As we are replacing the Development version (editable version) of the Staging DW PDM with this archive version, Meta Integration® Metadata Management (MIMM) simply wants confirmation. Now, click on the OK button.

Note, the version is now the Original:

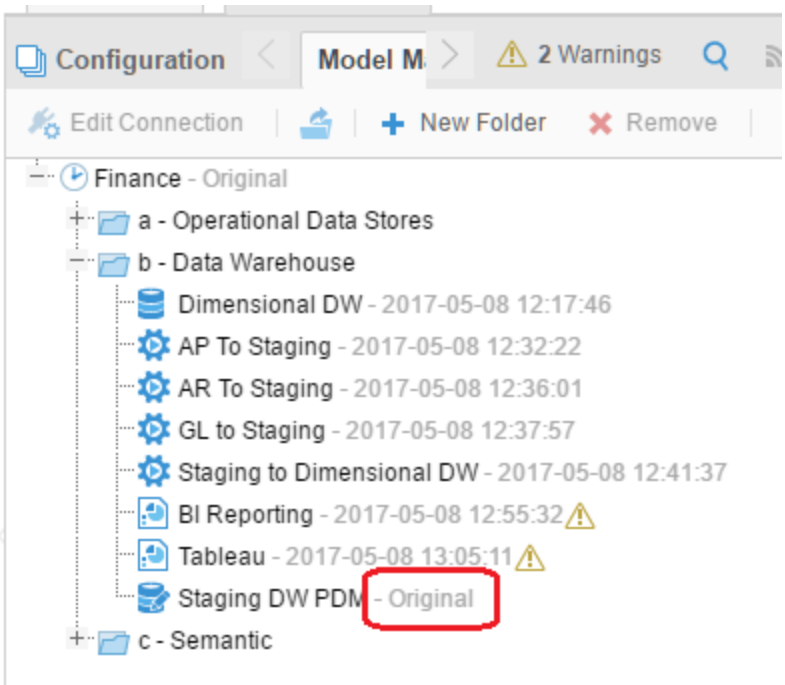


Figure 39 - Frozen historical (archive) configuration

We now have an historical configuration recording the Original state of the configuration. Be sure to right-click on this Original configuration version and select Read Only.

So, now it is time to migrate the new configuration forward to include the new v2 versions of the PAYTRANS and PAYTRANS To AR models.

Close the Original version of the Finance configuration.

Open v2 of the Finance configuration. Right-click on the More (⋮) icon and select Migrate:

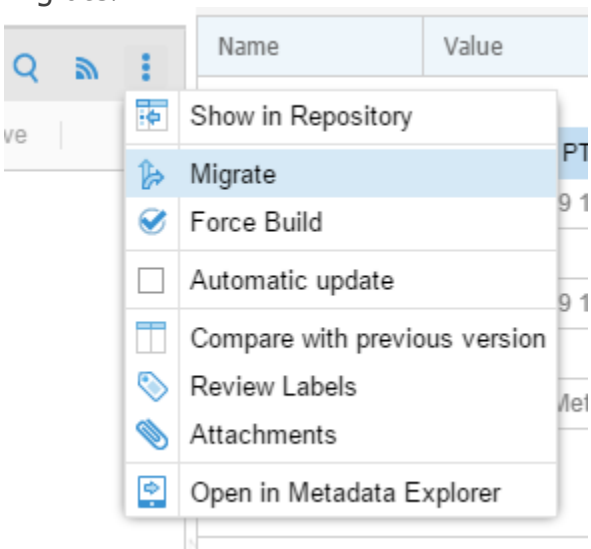


Figure 40 - Migrate dialog for the Finance Configuration

Note now that the new versions of the two models which were imported are included in this configuation, replacing the older ones. Again, you may see this:

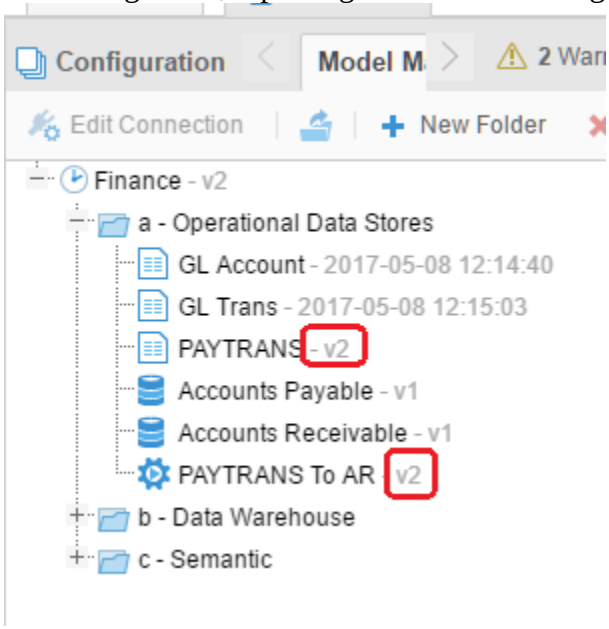


Figure 41 - New version of models is in Finance configuration

Once the operation is completed, go to the Architecture Diagram tab the configuration. Note, it has the same basic connections as the Original configuration version.

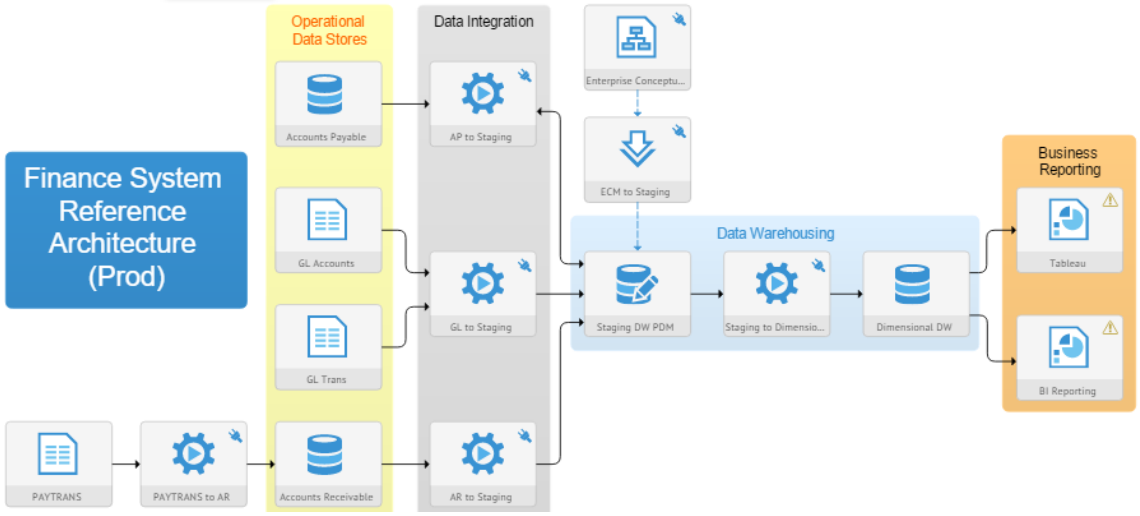


Figure 42 - Migrated Finance Configuration

4.2.2 Compare Straw man Configuration Versions with Published Version

So, now we have two versions of the configuration, the published one, which Explorer UI users see when they sign on, and the new migrated configuration which is not published and is only a straw man which we will use to analyze the changes made to the imported models.

Right-click on the new version of the PAYTRANS and select Compare with previous version.

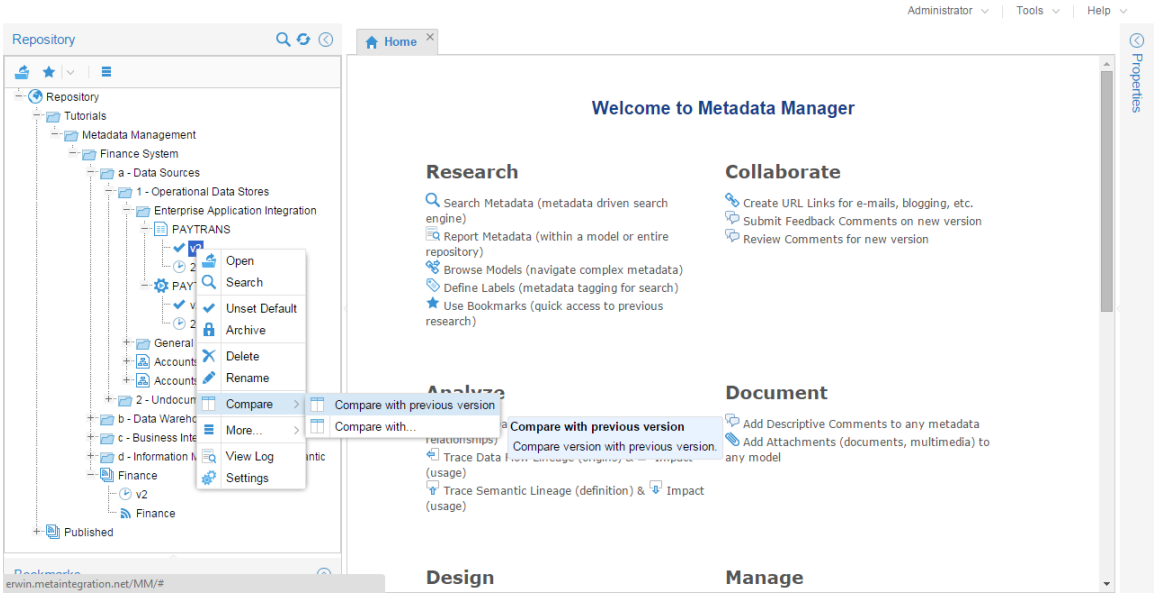


Figure 43 - Options for Comparing versions

Note there are two options here. One may compare with the immediate predecessor version (Compare with previous version), or may compare with any other version, even from any other configuration. We will compare with the previous (first published) version, so select that option.

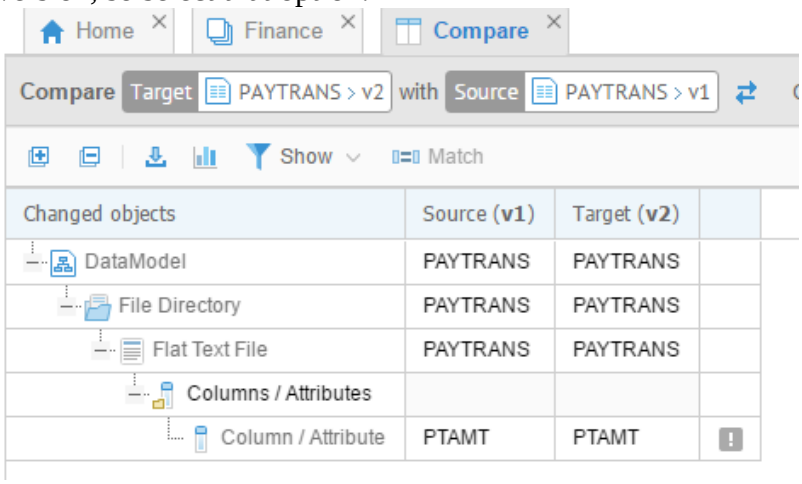


Figure 44 - Compare with previous version of a Configuration

Now we see that the only change is to the PTAMT field. However, what is the change? To see this, one should click on the row that is changed:

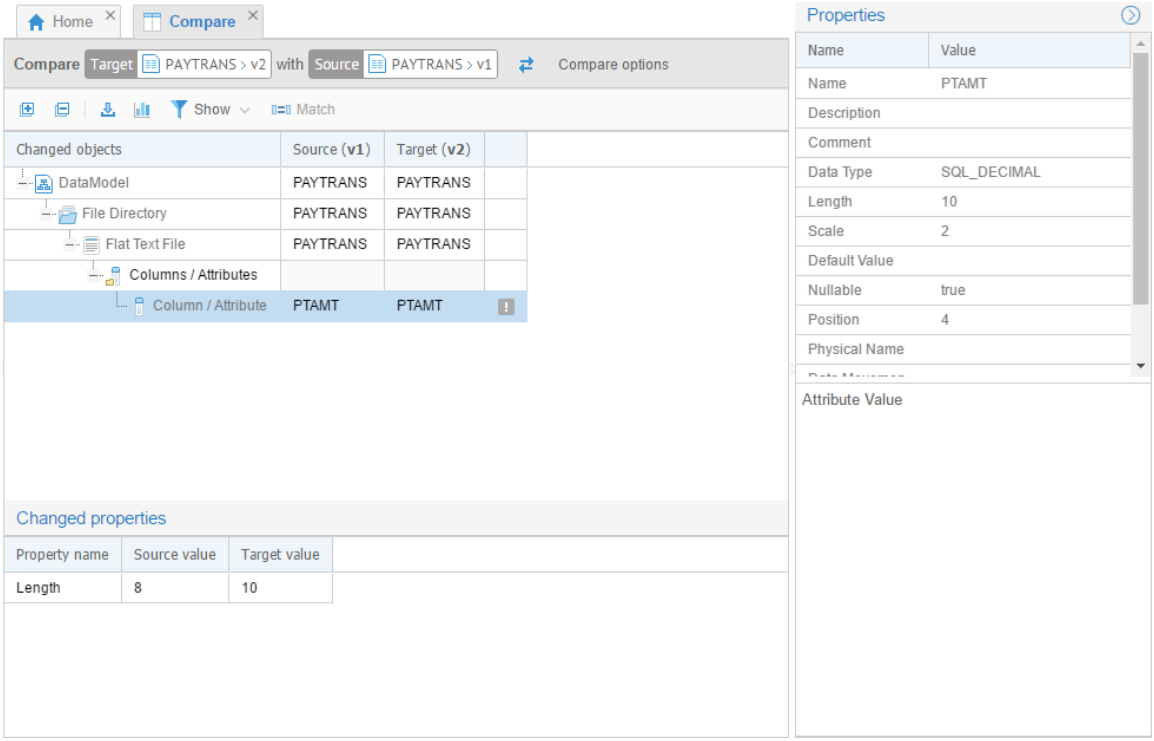


Figure 45 - Adding columns to comparison report

So, the PTAMT field is going to be larger, going from 8 digits to 10.

4.2.2.1 Determine Impact of Change

Now, the next question will be "what is the impact to the existing configuration?" I.e., what would be required in terms of changes to accommodate the larger field. Thus, we want to check this in the existing published configuration, not the new strawman, as that one has the complete current picture.

Right-click on the PTAMT row and select Show Source in Metadata Browser.

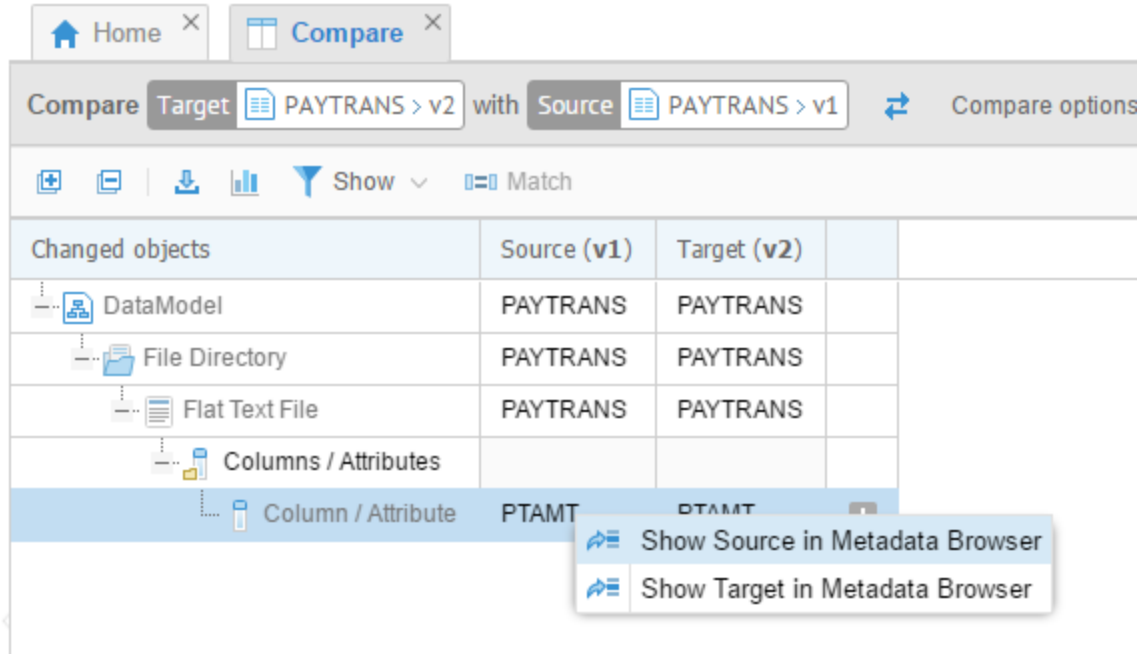


Figure 46 - Show Source in Metadata Browser

It is the source (not target) as indicated in the header of the report.

Then, right-click and select Trace data impact.

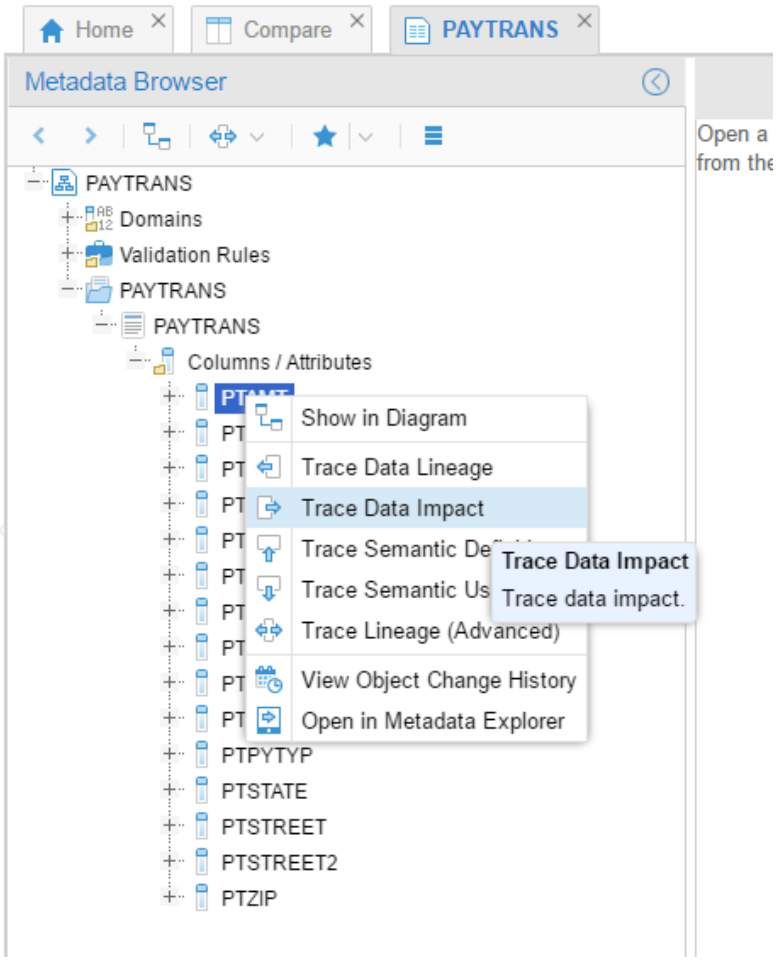


Figure 47 - Trace data impact due to change in PTAMT

Select the Finance configuration and specify the Textual lineage:

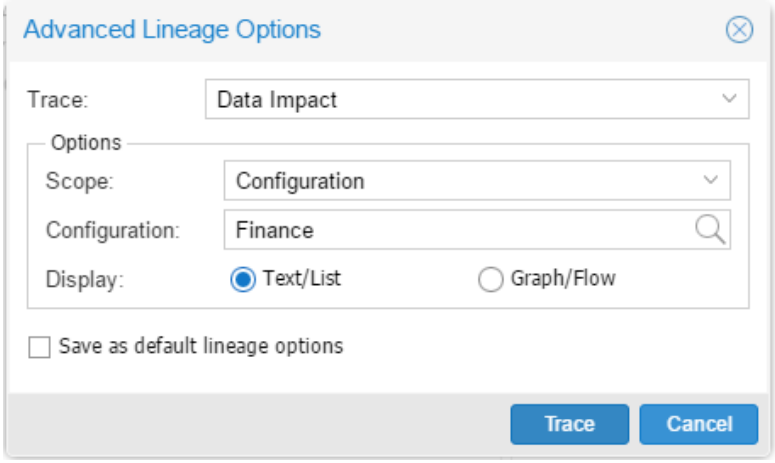


Figure 48 - Advanced Lineage Options

And the result is:

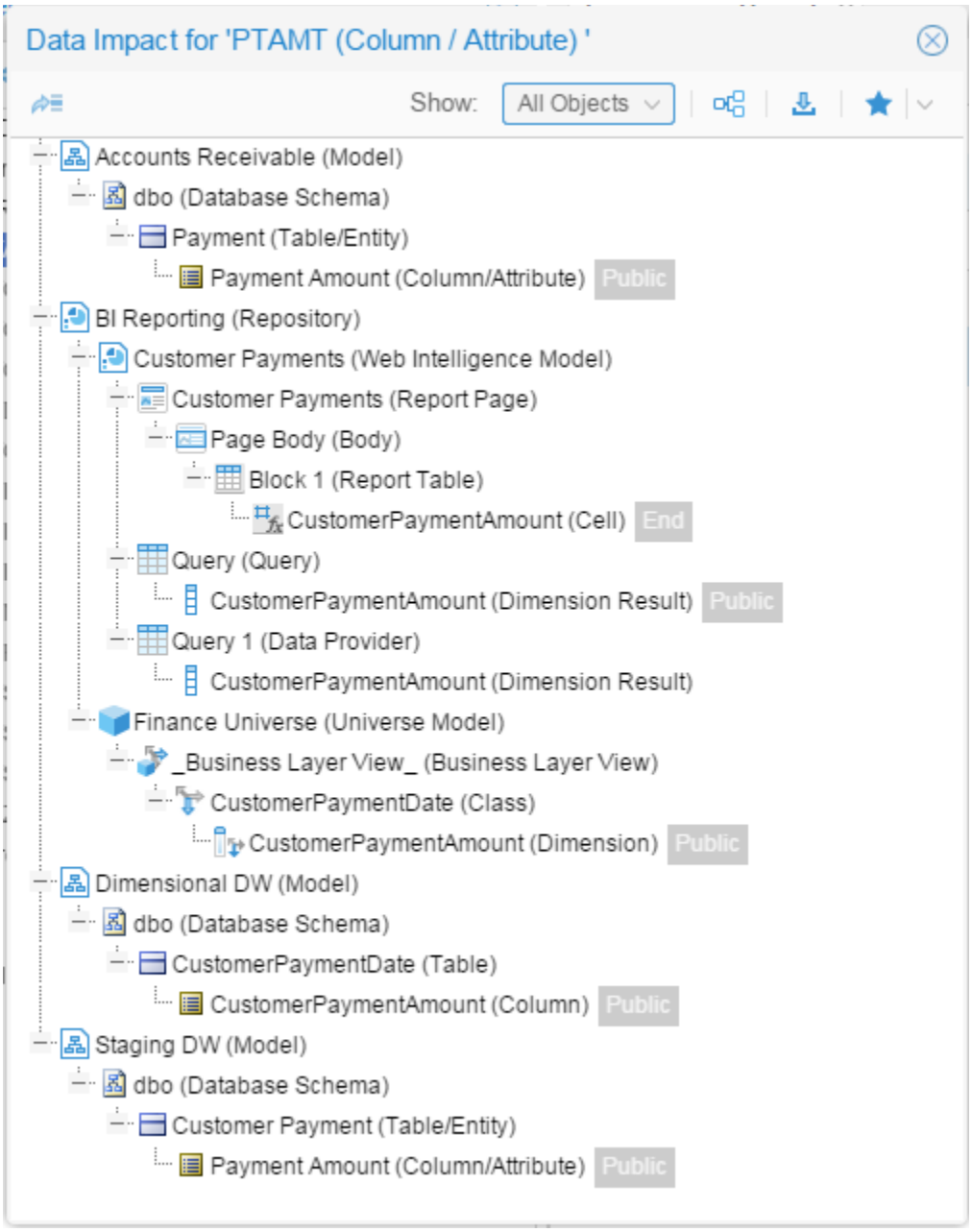


Figure 49 - New lineage in configurations for PAYTRANS PTAMT

As we are trying to determine ALL of the impacted models, not just the end of the data flow (reports), be sure to specify All Objects.

4.3 Using the Impact Analysis results

4.3.1 Determine Cost of Impact and Approve Changes

With the impact of change report available, the specific points of contact for the impacted systems may be contacted and made aware of the change. This is the generally the responsibility of the *Repository POC*. The *Program Managers* for the particular impacted systems then cost out the change request (estimate of the cost for this change), approved, and new metadata (versions of the models) will be provided based upon updates due to the change.

To send this impact report to the interested parties, simply click on the Bookmarks icon and select Get Link. Copy the URL and paste it into an e-mail and send that off, with a short explanation, pointing out that this report should be used to cost out the impact from this change.

Performing a little of this analysis, right-click on the Payment.Payment Amount column in the Accounts Receivable model and select Show in Metadata Browser.

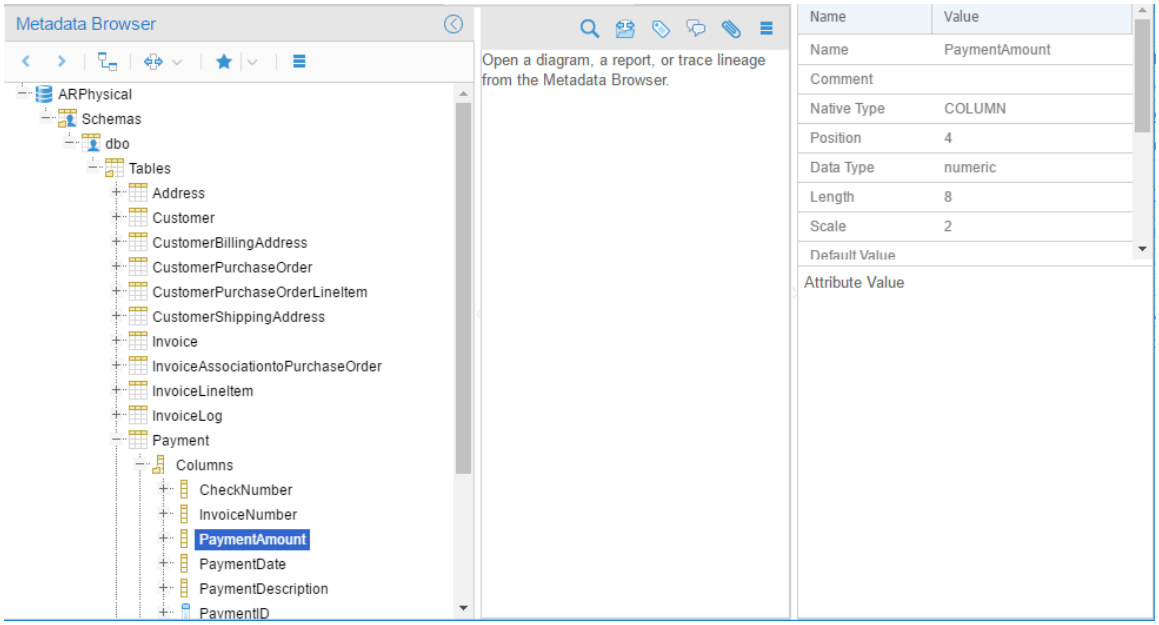


Figure 50 - Payment column in the Accounts Receivable model

Note, in the Properties panel you will see the Decimal (8,2) datatype defined for this column. Hence, this will need to be changed in this system.

If you do the same for the CustomerPaymentAmount dimension in the Dimensional DW model, you will see that this field is already expanded. In this case, based upon their analysis, the following databases and ETL processes needed to be updated:

Model	Version Name	Version Description
PAYTRANS To AR	V2	Expanded

		Payment.Payment Amount to Decimal(10,2)
Accounts Receivable	V2	Expanded Payment.Payment Amount to Decimal(10,2)
AR To Staging	V2	Expanded CustomerPayment.Payment Amount to Decimal(10,2)
Staging DW	V2	Expanded CustomerPayment.Payment Amount to Decimal(10,2)

Figure 51 - Updated model versions for this configuration

Fortuitously, the Warehouse itself already accommodated the larger amounts, so no changes are required there, nor any to the business intelligence solution.

Note, even in this view the ETL/DI processes are not shown as boxes and elements to click on. Instead, you see a thick line that can be clicked on. Click on this line:

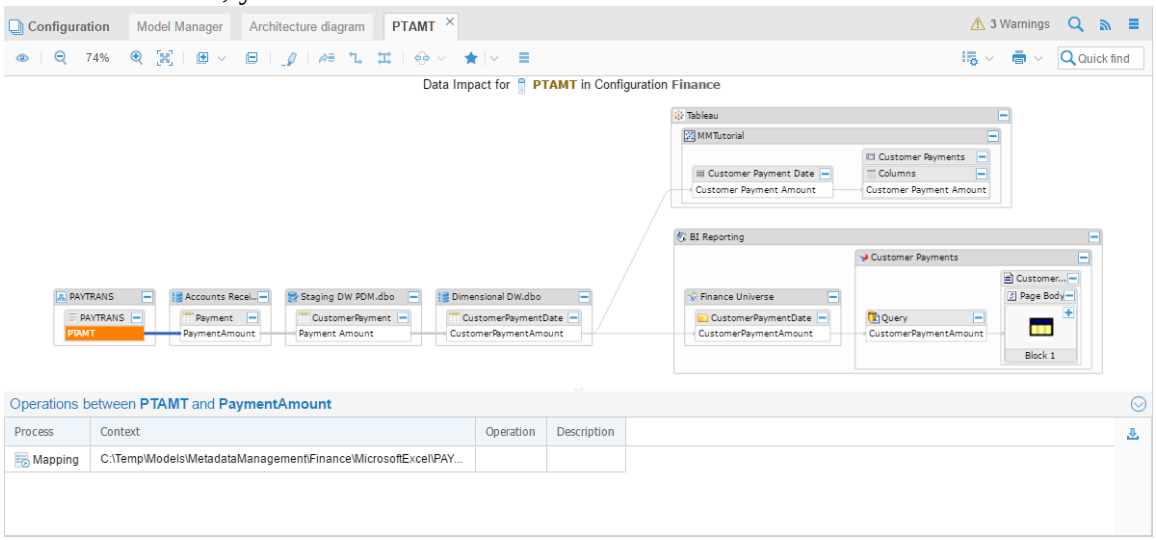


Figure 52 - Click on data process line

Note, the information about the DI operation below. Right-click on this row and then select Open ETL Details.

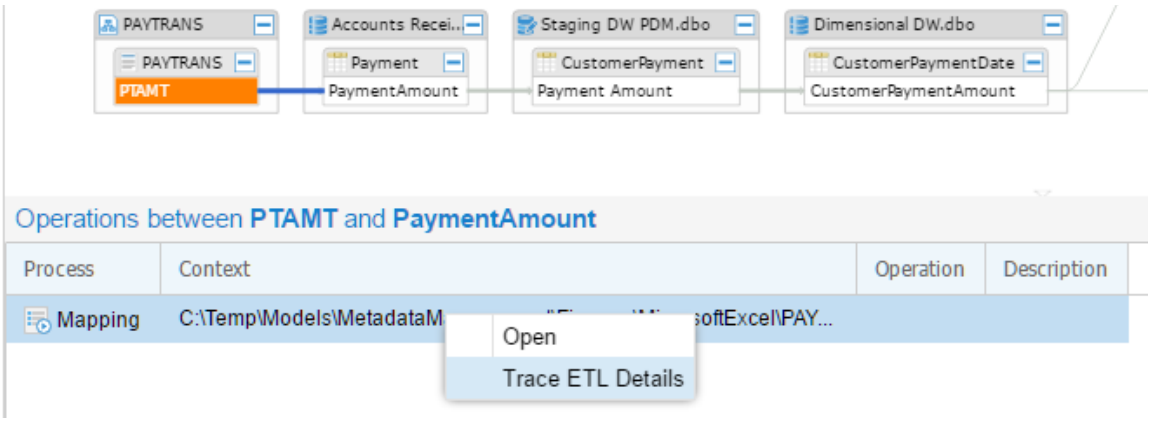


Figure 53 - DI/ETL

And you see the impacted connections in the DI/ETL.

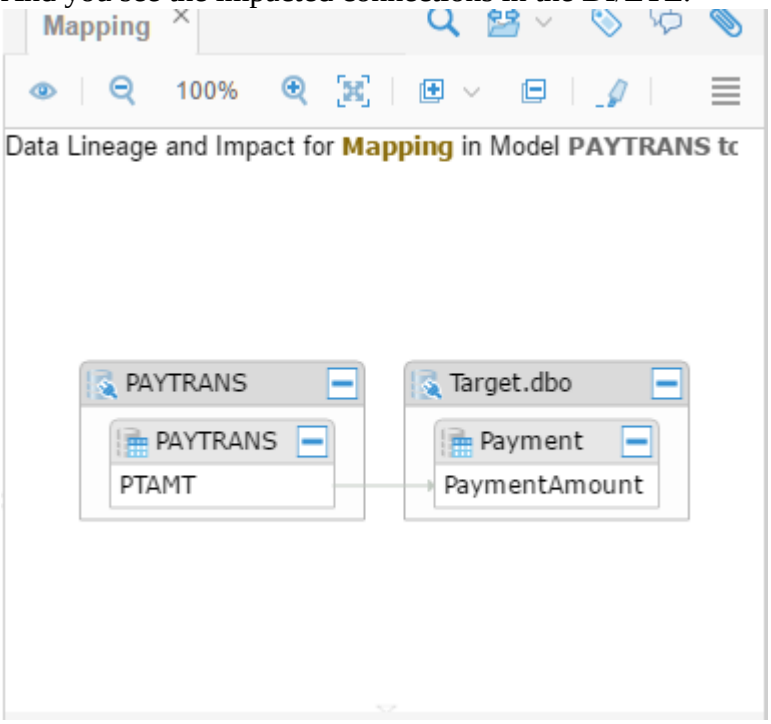


Figure 54 - Trace ETL Details

You may also simply Open the DI/ETL. This takes you to the mapping for this process. Right-click and select Open:

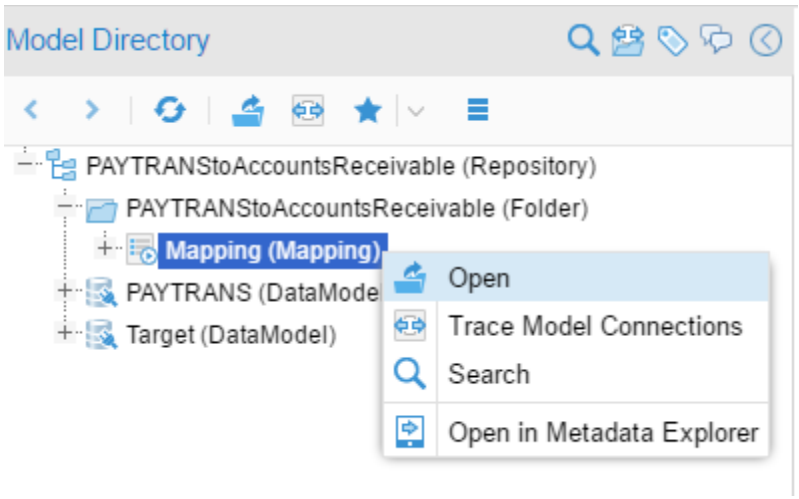


Figure 55 - Open MappingClick on the Data Flow Overview:

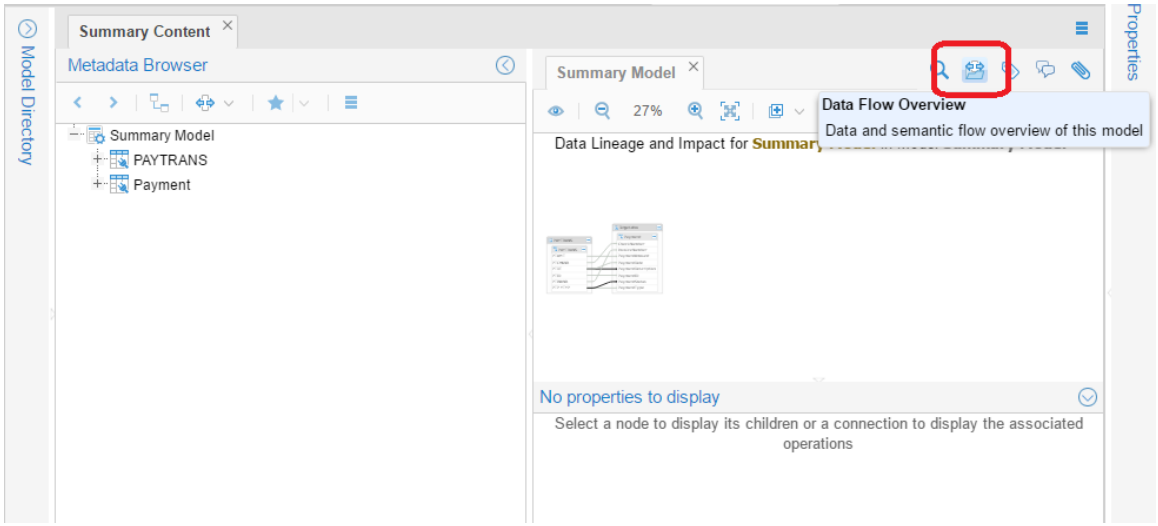


Figure 56 - Data Flow Overview

And select the Summary Mapping Overview.

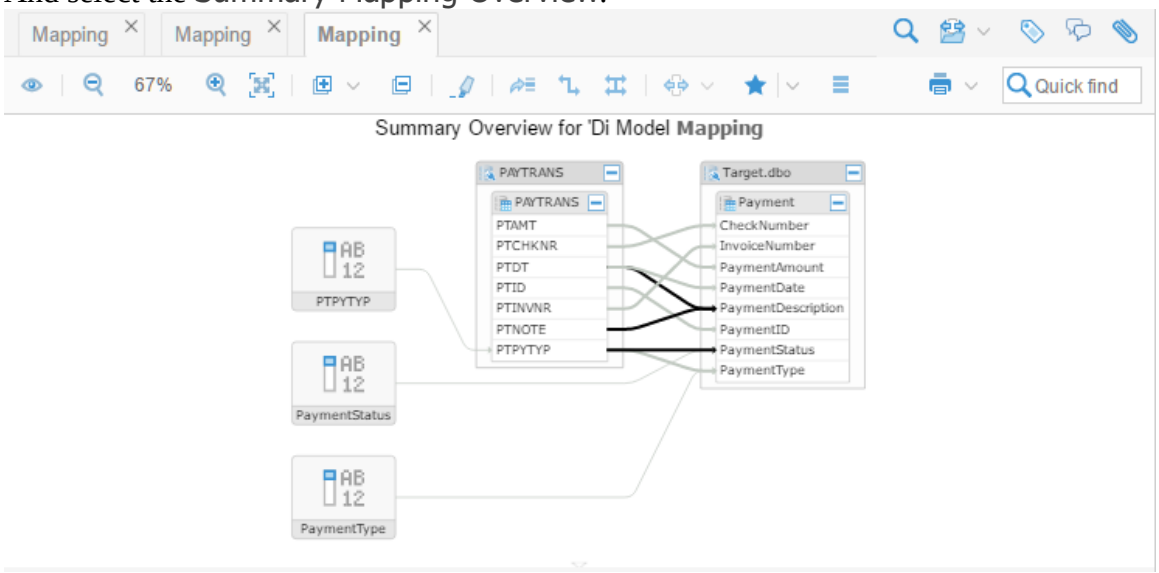


Figure 57 - Data Process Summary Content

At this point, the development teams for the impacted systems need to get busy and update their databases and ETL processes.

4.3.2 Harvest Updates to Impacted Models

Once the new metadata is available, it is time to harvest again and then migrate the configuration. For the purposes of the tutorial, we will simply import from the same file locations, but after having updated the actual source files with the new metadata. These updates are in the v3 directories.

Then, in order to assure that the new versions of the updated ETL metadata files and data store model files copy the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v3

into

C:\Temp\Models\MetadataManagement\Finance

and agree to the replacement of files. This action will update the source metadata for the impacted systems "downstream" of the change to the payment amount data type, just as if they were provided by the specific Repository POC's.

Now import the impacted models into Meta Integration® Metadata Management (MIMM) using either the manual process or automatic scripts and assigning names and descriptions as defined above.

First, we will also import that new version of the ETL model.

So, import the model content labeled PAYTRANS To AR

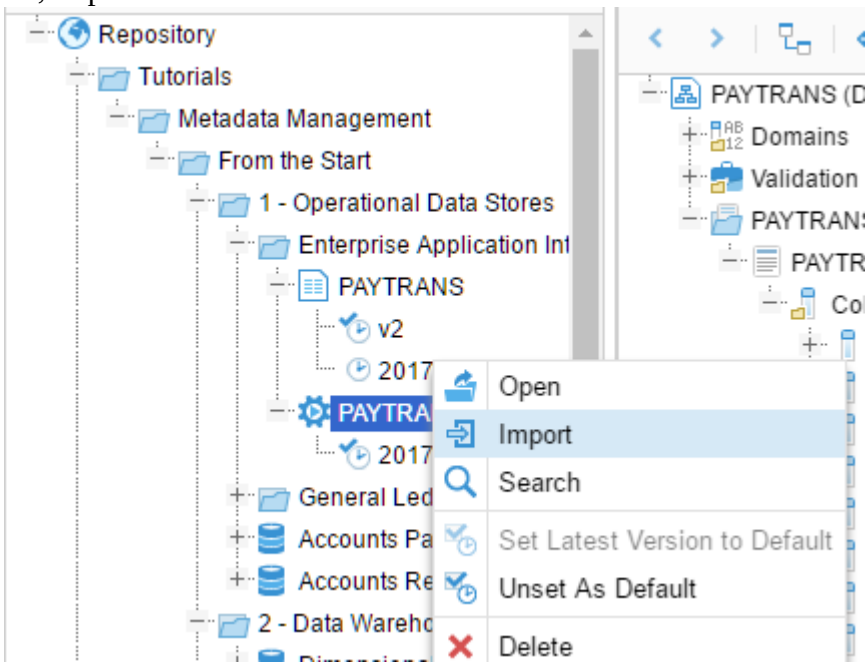


Figure 58 - Import from context menu

and also placing "Expanded Customer Payment.Payment Amount and PTAMT to Decimal(10,2)" in the Description field and place "v2" the Name field.

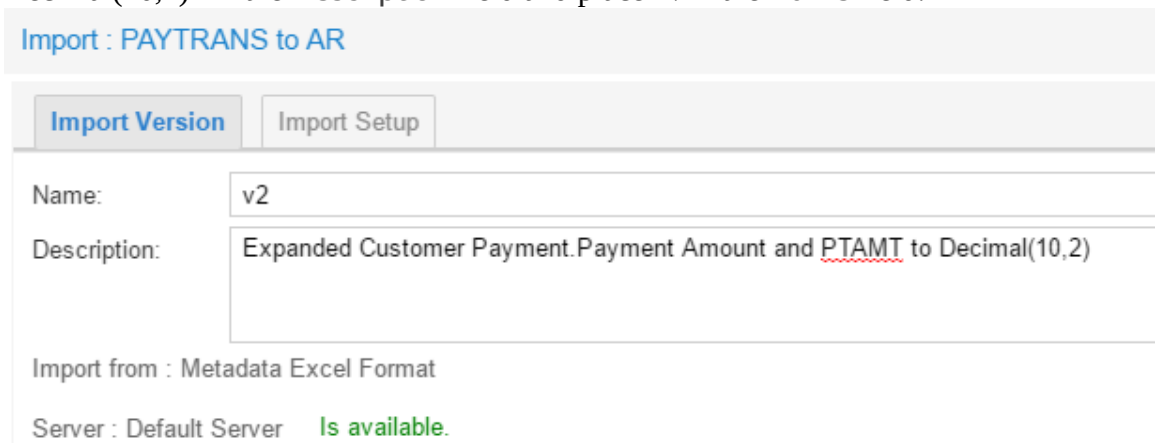


Figure 59 - Import dialog

And we have the following:

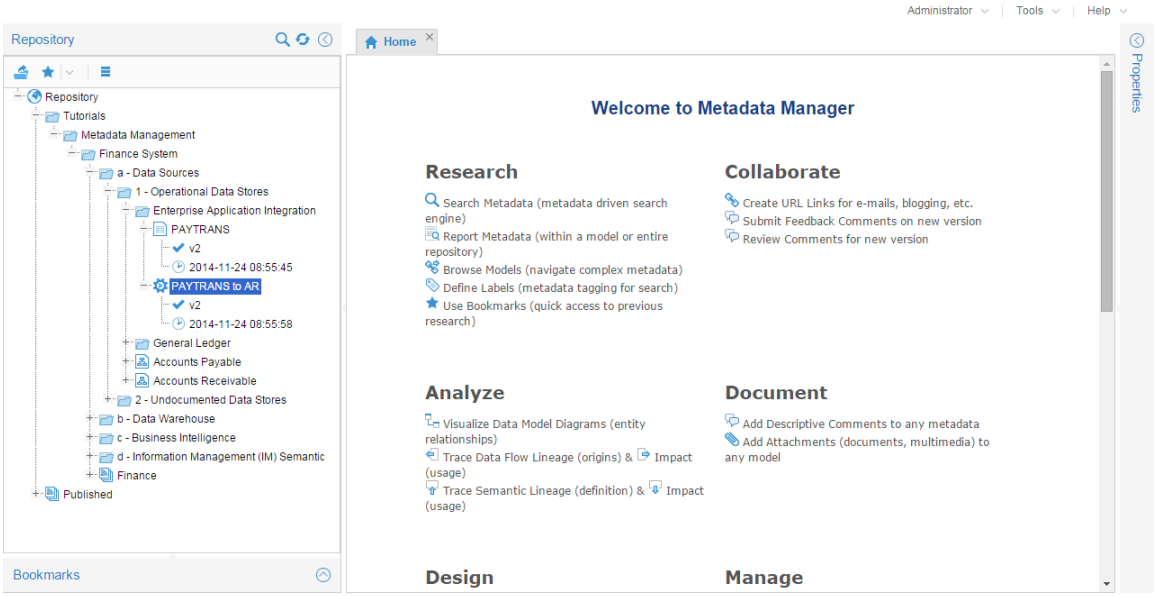


Figure 60 - PAYTRANS To AR model in the browse tab

Now, right-click on the PAYTRANS connection [folder] content and select Open. This is the connection definition in the ETL mapping which points to the source model, in this case the PAYTRANS message.

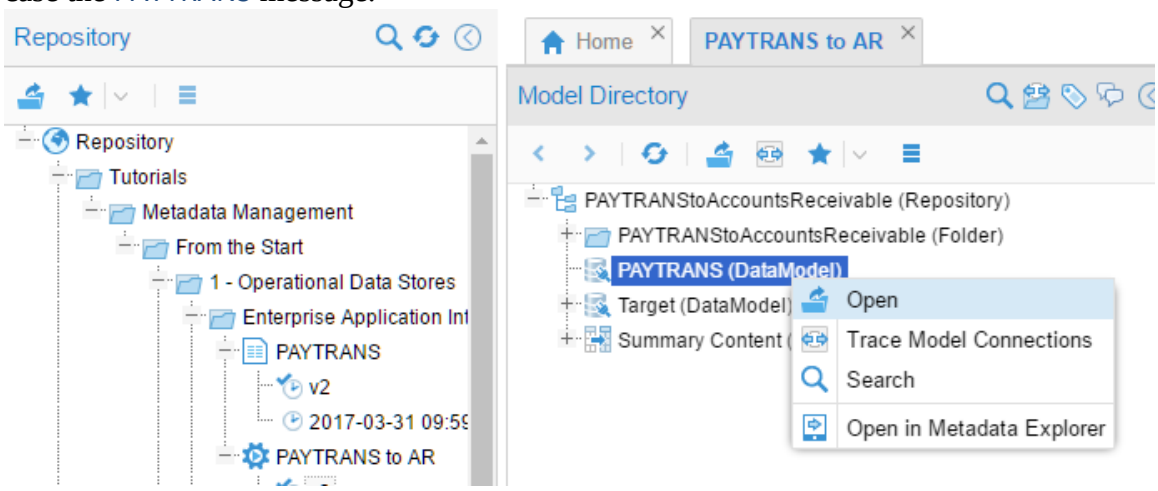


Figure 61 - Open the connection

There is now a new tab in the center panel. If one then navigates down to the PTAMT attribute and its data type, one will see that the ETL on the source side has also been updated with the new data type.

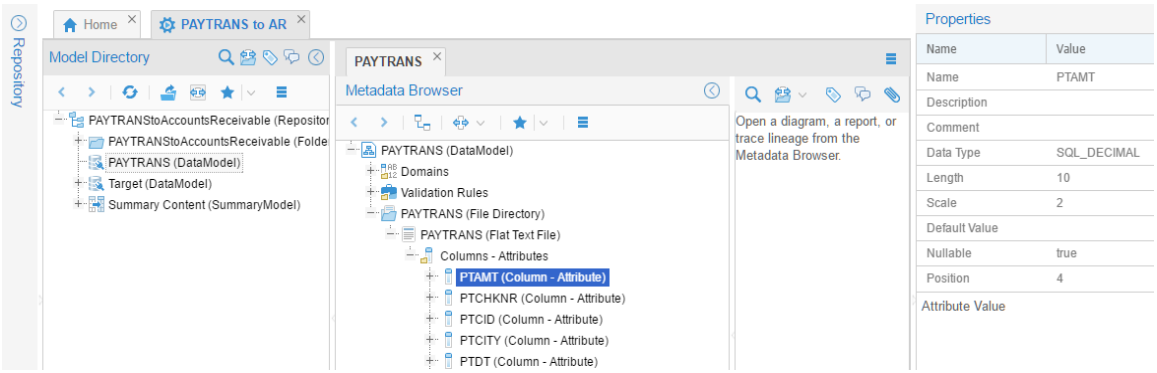


Figure 62 - Updated data type

Note the check mark next to the new version. This notation indicates that the new versions are the *default* versions. Only one version of a given [repositoryobject] may be assigned default status, and a default version is treated specially. Basically, any action that is applied to the model (not the version) level is automatically applied to the default version.

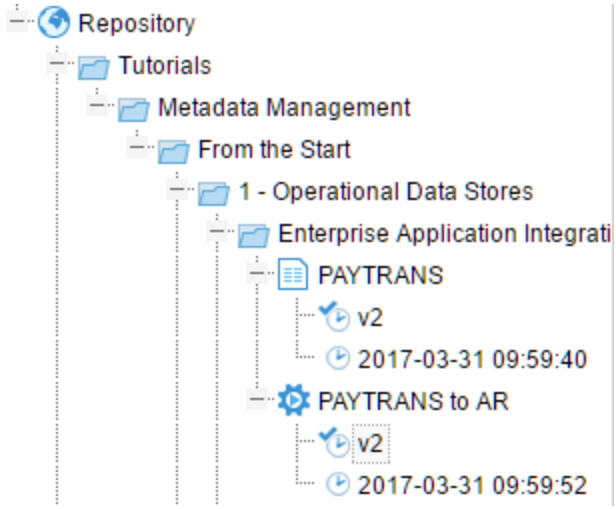


Figure 63 - Updated data type in ETL model from PAYTRANS

Now, import the rest of the new models. These are the

- Accounts Receivable and Staging DW databases
- AR to Staging data process

Please use the reference at the end of the chapter for version names and descriptions.

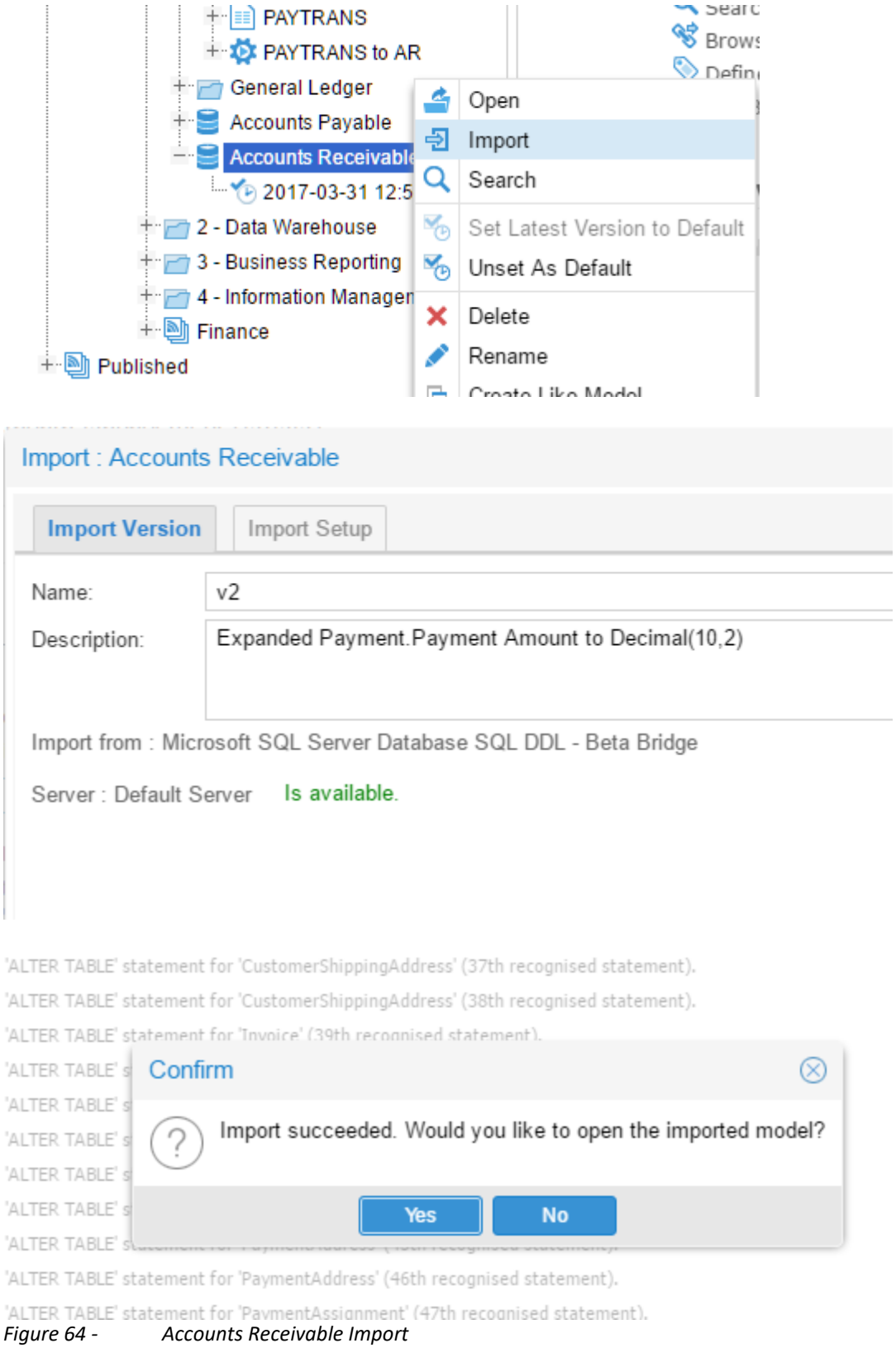


Figure 64 - Accounts Receivable Import

Now we Sync with Database for the Staging DW PDM:

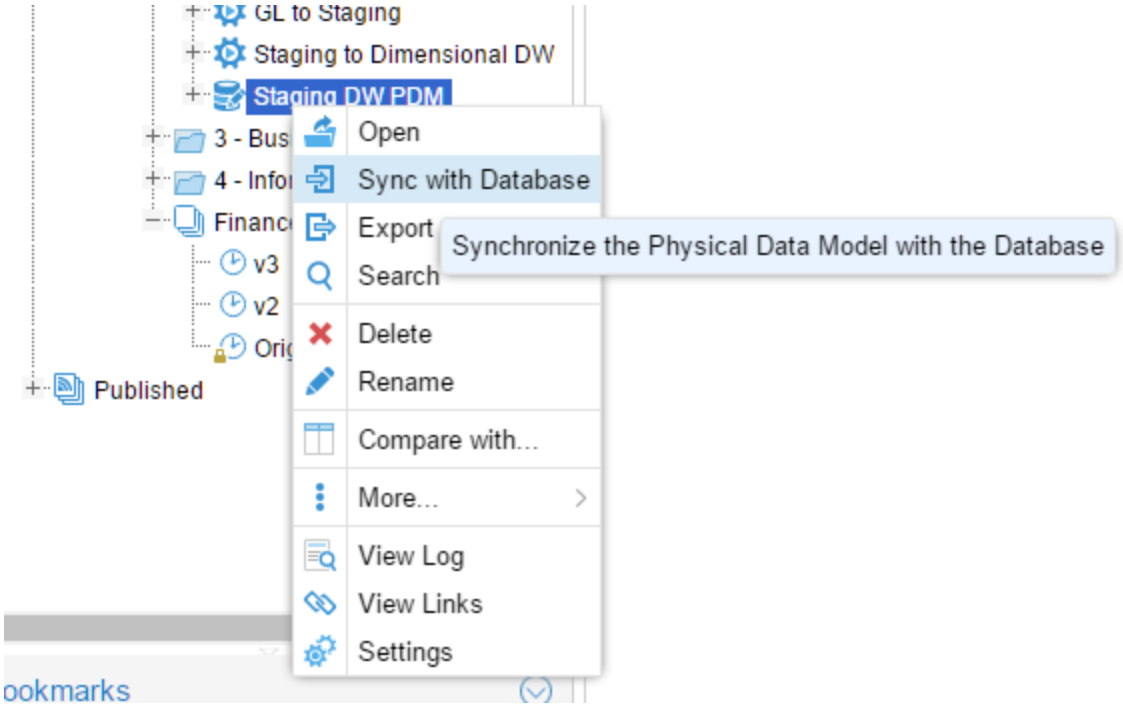


Figure 65 - Import Staging DW

Thus, the Staging DW PDM is updated anyway with a SyncBackup version.

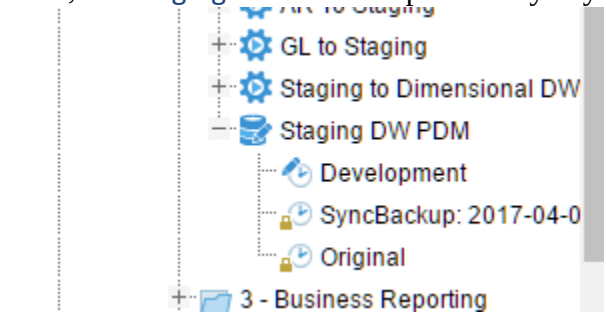


Figure 66 - Updated Staging DW PDM

Rename this one as v2, Expanded Payment.Payment Amount to Decimal(10,2)

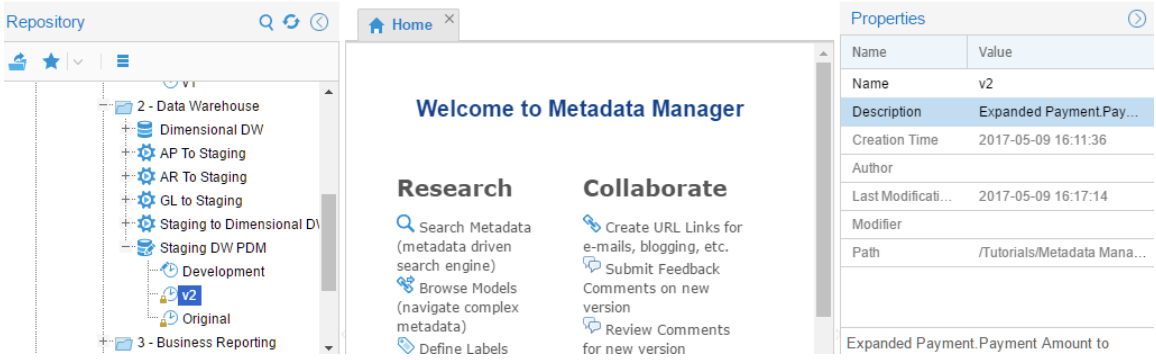


Figure 67 - Archived version v2 of the PDM

Now that the PDM is done, import the AR to Staging data process.

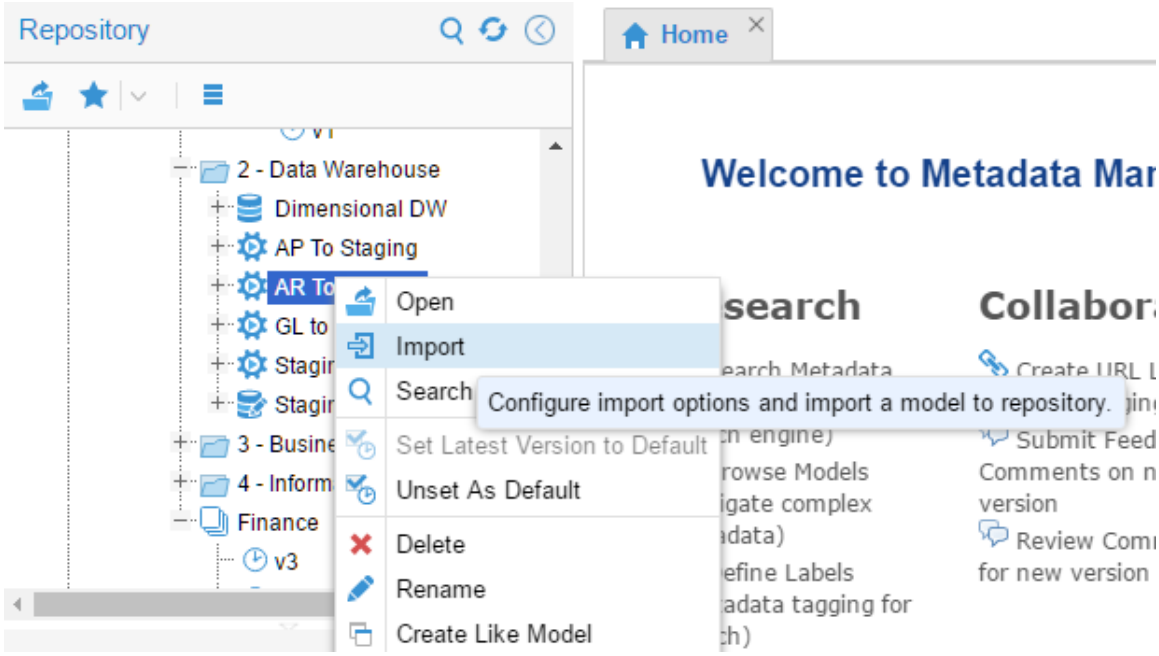


Figure 68 - Import AR to Staging

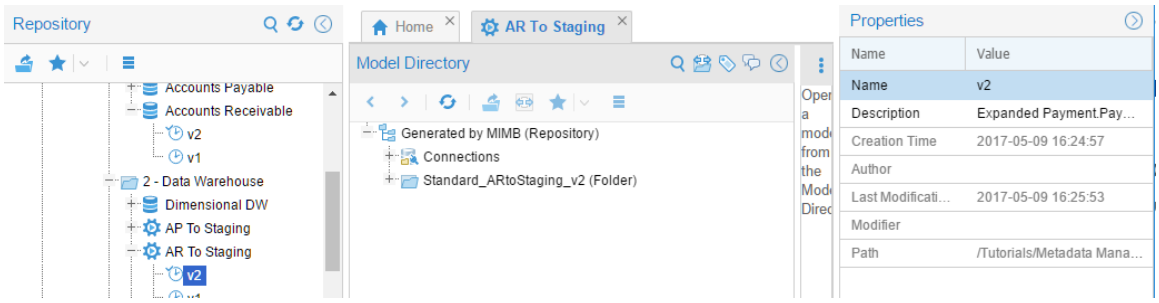


Figure 69 - Imported and renamed

4.3.3 Re-Migrate the Configuration Version

With the new models that were imported, we could always produce a new *To-be* configuration version. This configuration version is unnecessary, however, unless one needs to maintain an historical version or a roll-back position. However, the last good roll-back would probably be the already created Original version of the Finance configuration. Thus, instead we will simply use the existing v2 version but will rename it v3 so as to keep it straight with the table at the end of this chapter. Update the description also to "Expanded Payment.Payment Amount to Decimal(10,2)" and the version name "v3".

Open the configuration. Note, it has the same basic stitching rules as the published configuration. Note the Accounts Receivable model points to the older harvested version,

just like the older configuration version.

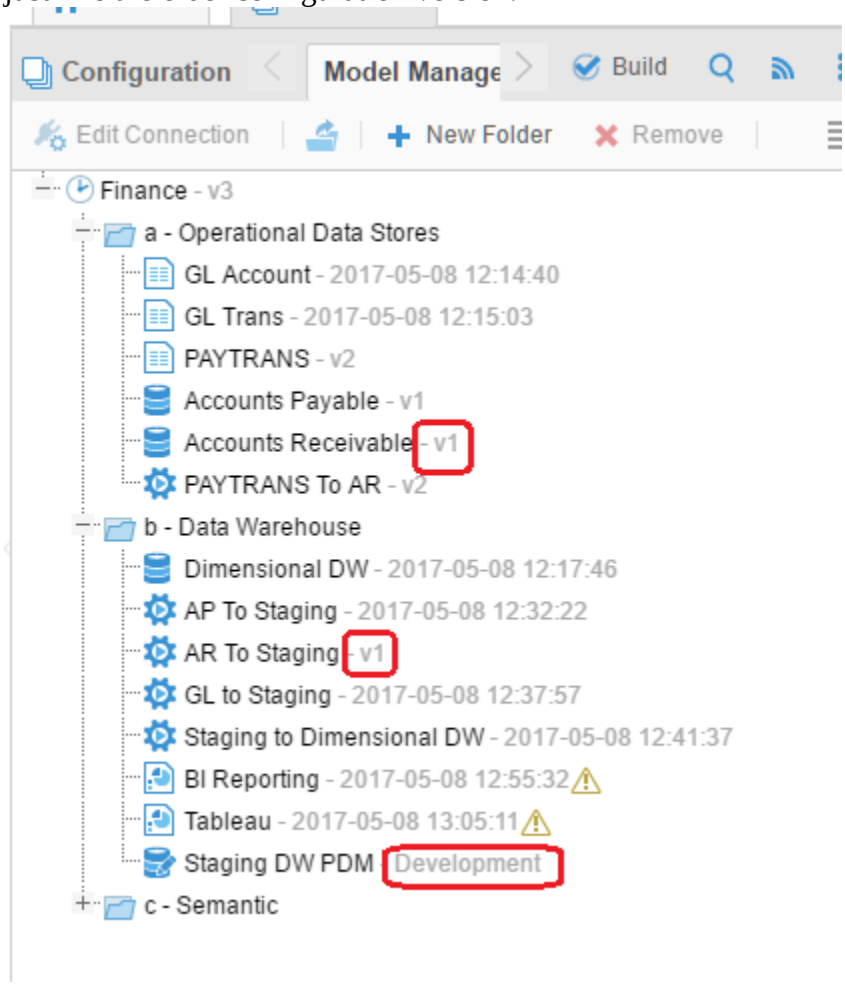


Figure 70 - Older version in configuration

However, the Staging DW PDM is already the Development version. This is the way that [physicalDataModels] work. When the underlying model that synchronized with it is harvested, a new read-only version is create and the Development version with the latest harvesting information is included in the configuration.

Select Migrate. Now, the Accounts Receivable and AR to Staging models points to the newly harvested versions, as opposed to the Original configuration version, which points to the older versions of the models.

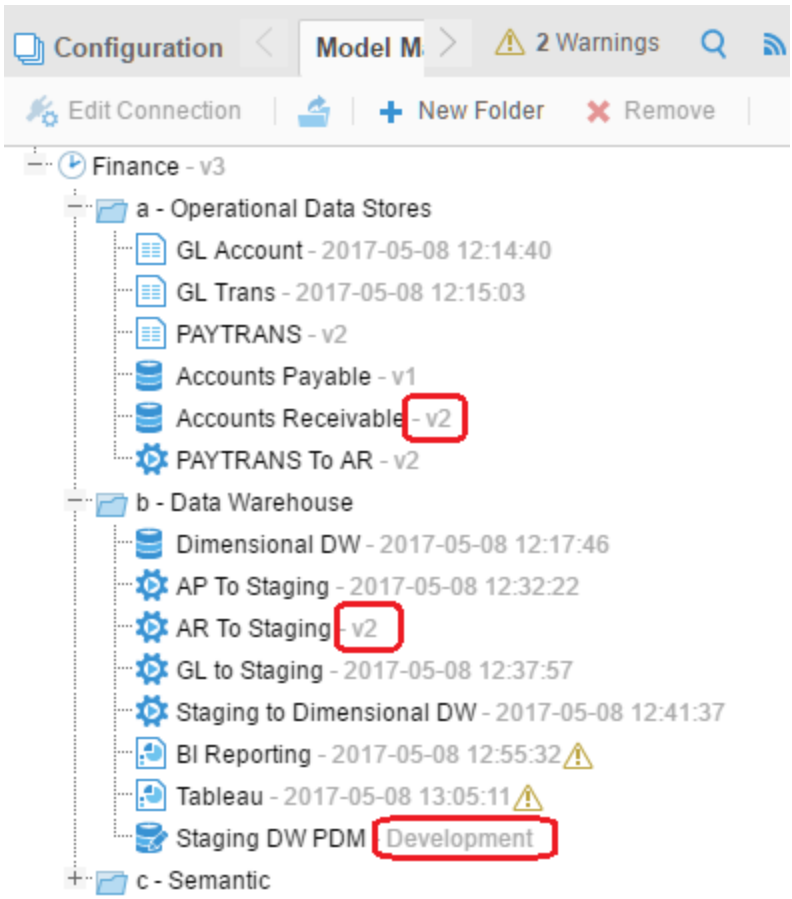


Figure 71 - Updated versions

4.3.4 Determine Completeness of a New Stitching

Build the configuration version.

When building the new version, a set of logs are produced. One may view these at the end of the migration process in the dialog present upon completion. One may also right-click on the version of the configuration (or the configuration itself, if one is filtering out versions from the Repository Tree panel) and select Show Log to see the log for that version migration. In addition, one may open the configuration and either select Edit Connections for a given model with connection definitions.

Once one has verified that the stitching are indeed *valid*, and if this represents the information you wish to provide to the business analysts, etc., then it is time to publish this new version of the configuration. To do so, right-click on the version of the configuration and select Publish. This way, the configuration updates reflecting the new Payment Amount data type length are presented to business users when viewing the configuration.

4.4 Transitive Closure – Propagating Conceptual Impacts

In the previous section, we have a complete and published set of new models and configurations. They are certainly valid unto themselves. However, it would be a good idea to determine any impacts that the changes have on any of these other configurations.

Remember, when investigating the impacts due to the changes in Payment Amount, it is important to consider *semantic* lineage as well.

4.4.1 Dev configuration

This work will be conducted in the Dev environment and then be implemented again in the Prod environment. These activities will be conducted outside of Meta Integration® Metadata Management (MIMM) so all we need to worry about is harvesting the proper models when ready. However, Meta Integration® Metadata Management (MIMM) will prove very valuable in the development effort.

4.4.2 Discovering a need for standardization

Updating the size of the Payment Amount columns in several models to decimal (10,2), may have larger impacts system wide. It is not difficult to imagine that there are a large number of "amounts" used throughout the different systems, many of which may be decimal (10,8) or other sizes.

Obviously, variations such as this could cause several types of problems:

- There could be a loss of information downstream or in terms of related information, e.g., a decimal (10,2) column could feed into a decimal (8,2) field and then later become a decimal (8,2) field. We performed an impact analysis of the downstream data flows already, though, so this possibility is less likely here.
- There may be amounts which currently also need to be expanded but such expansion has been delayed and worked around. E.g., the Accounts Payable system will have a number of payment and invoiced amounts which may need to be expanded in the future. It may be wise to do this all at once while this effort is ongoing in order to *standardize* these elements.
- We may already have a *standard canonical form* (defined standards reused throughout our systems) which is defined for amount-type elements, that should be updated to reflect our new need for larger numbers. This standard or canonical form is like defined as a data type or domain within our data dictionary or Enterprise Conceptual Model that we harvested earlier.

There are several approaches which may be used to determine the extent of such a variation in the current systems, and then how one might ameliorate any potential problems. Here we investigate some of these.

4.4.2.1 Searching for terminology in a configuration

Meta Integration® Metadata Management (MIMM) has a very sophisticated text based search that is highly metadata specific. In a previous section, searches were performed using the Explorer UI. Here the administration UI will be used for searching.

Recall that the advanced search is scoped by configuration (and by version within that configuration) so as to avoid repetitive "hits" of the same element in multiple versions of the different models, multi-models and mappings contained within. Thus, the first question for the search becomes one of scope, i.e., how broad a search (configuration) and for what release or development stage (version).

In this case, the Finance configuration will work well, and the currently published version is the correct one to use, as it is now the approved version of metadata for those system.

To perform the search, right-click on the published version of the Finance Prod configuration in the Repository panel and select Search.

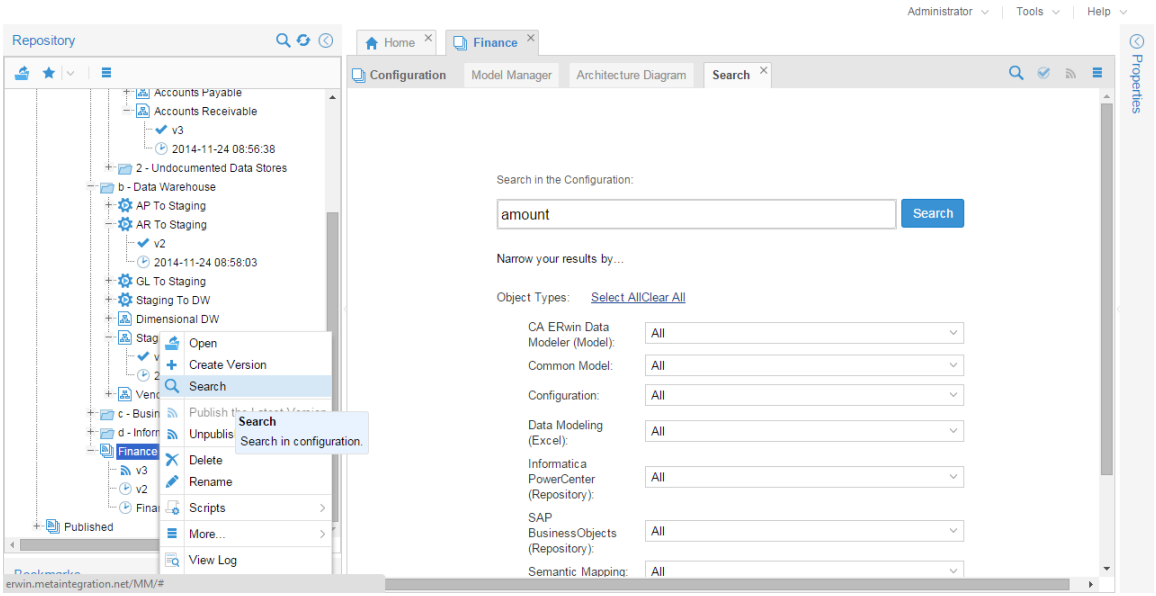


Figure 72 - Searching for other fields of type "amount"

Note, the configuration has already been selected (by our action). Here, enter the phrase "amount". This way, we will only find metadata elements which have the word "amount" in the Name. Now, click on the Search button.

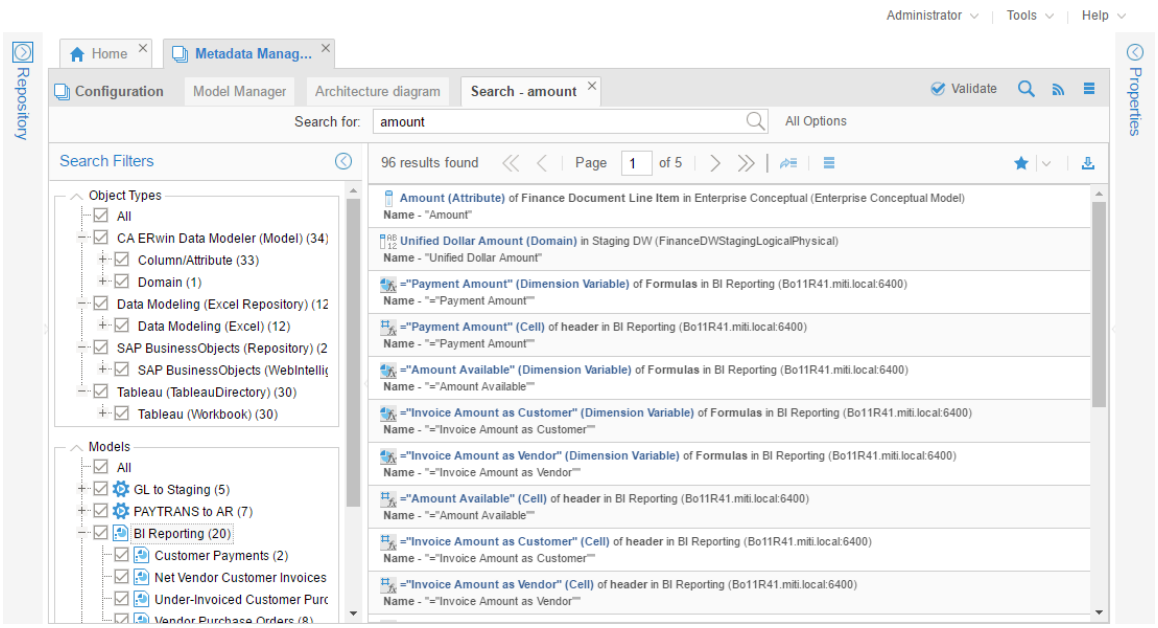


Figure 73 - Result of search for other account-type metadata elements

At this point, as we are looking for stored data columns, and not mapping data elements or BI, we may remove the check marks next to the Semantic Mapping entry which provided matches.

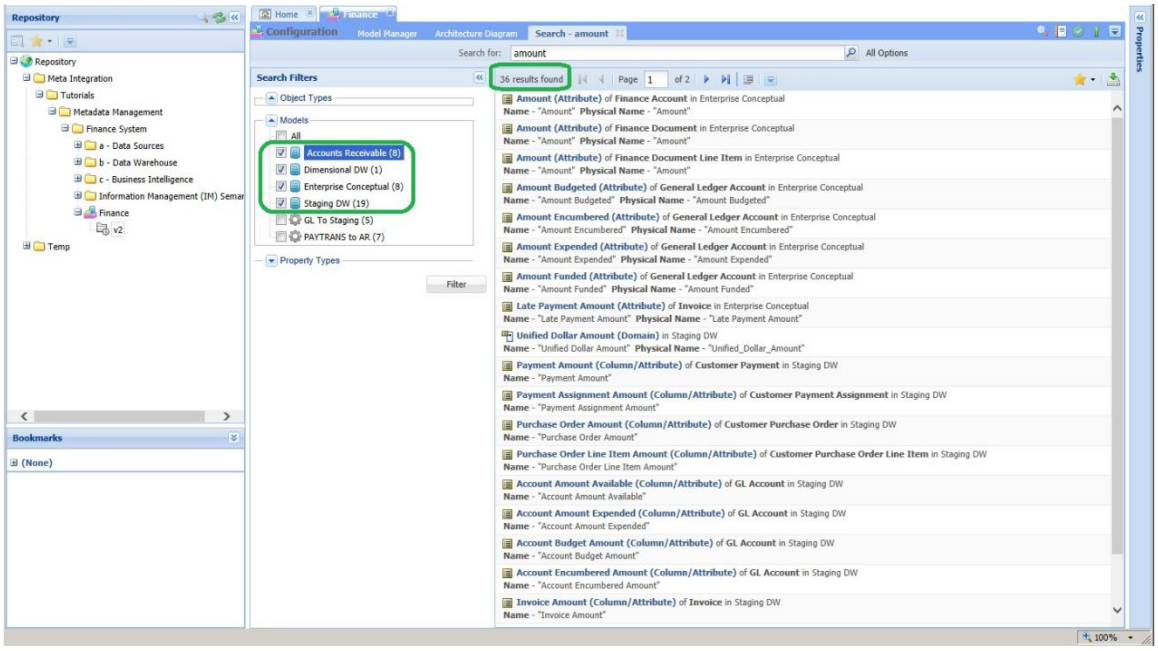


Figure 74 - Filtering search results to data stores only

We are now at 51 results and can stop right here and start viewing the individual models and recommending data type changes.

4.4.2.2 Searching for terminology in a critical model

However, a bit of analysis of the lineage for the Finance Prod configuration suggests another method: one that is opened up by the Meta Integration® Metadata Management (MIMM) lineage analysis capability. A quick glance at the high-level lineage diagram for this configuration shows that all data is collected by the Staging DW PDM. Thus, all we really need to do is search there and then determine all upstream and downstream data elements in the lineage. Such a report will be the basic work order for standardizing all amounts to a common canonical form.

4.4.2.3 Using existing canonical forms

The Meta Integration® Metadata Management (MIMM) is also capable of tracing through *semantic lineage*, which are relationships representing semantic commonality or association instead of data flow mappings. Semantic mappings were created in an earlier section when we harvested the mapping between the Enterprise Conceptual Model and the Staging DW PDM model. Let us open the Finance Prod configuration which includes the semantic models along with the data flow related models, and click on the Architecture Diagram tab. Be sure to open the new published version that we just created.

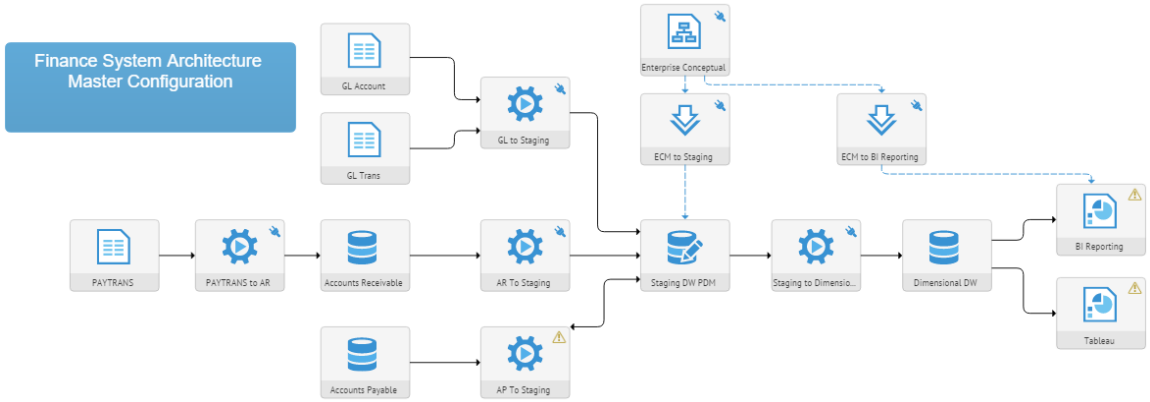


Figure 75 - Architecture Diagram

From this diagram, one can see the warehouse in the middle of the data flow, and also the semantic links from the Enterprise Conceptual Model. In this way, we will be able to:

1. Navigate to the Payment Amount that was updated in the Accounts Receivable model
2. Follow the data flow downstream to any impacted attribute in the Staging DW PDM model
3. Then trace back via the semantic lineage to the Enterprise Conceptual Model and the canonical form for finance amounts fields
4. With this information, we may then trace back ALL of the attributes in the Staging DW PDM model which have been mapped to that canonical form (making good use of all that data administration work)
5. Finally, one may use the dataflow lineage and impact analysis to identify ALL of the data elements in ALL of the systems which are impacted by or impact these fields.

We refer to this process as transitive closure, or the idea that if A→B and B→C, then A→C. In our case, we note that the Payment Amount impacts elements in the Staging DW PDM model, which is to, related via semantic mappings, a canonical form in the Enterprise Conceptual Model (A→B). Then we trace back down to the Staging DW PDM model (B→C) and realize that all of the element related in the dataflow to those "C" elements likely should have the same data type.

Following the above description, we navigate to the Payment Amount that was updated in the Accounts Receivable model. Open the model and enter ".Paymentamount" in the Find text box and press the Enter key. Then click on the resulting element.

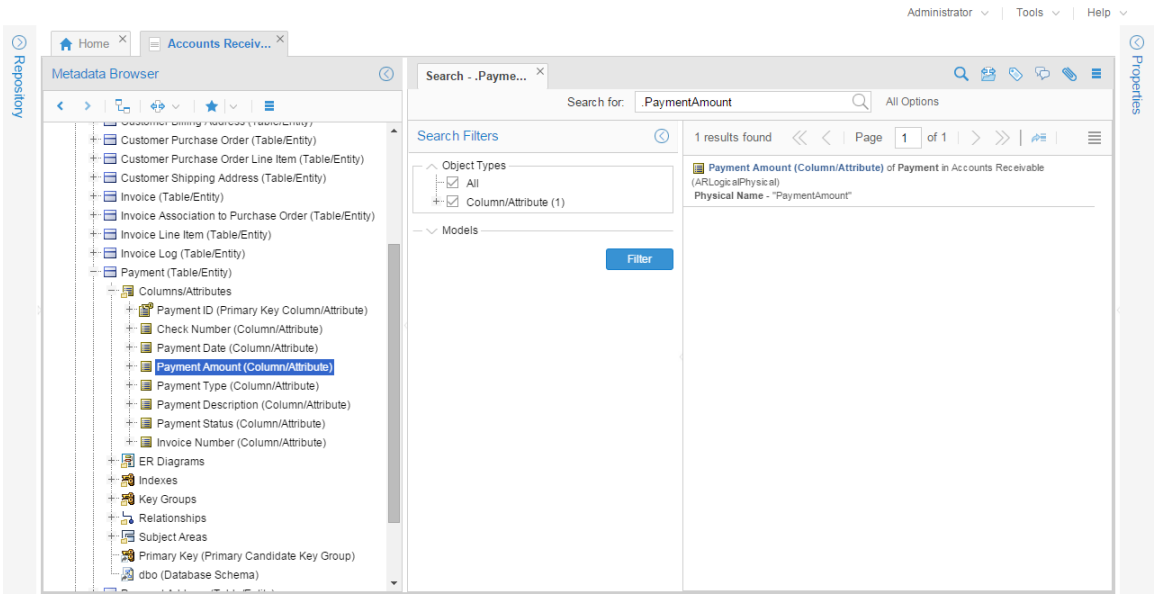


Figure 76 - Changed element in Accounts Receivable model

Step 2 is to follow the data flow downstream to any impacted attribute in the Staging DW PDM model. To do so, right-click on the column and trace the lineage in the Finance Prod configuration. Be sure to use the text report.

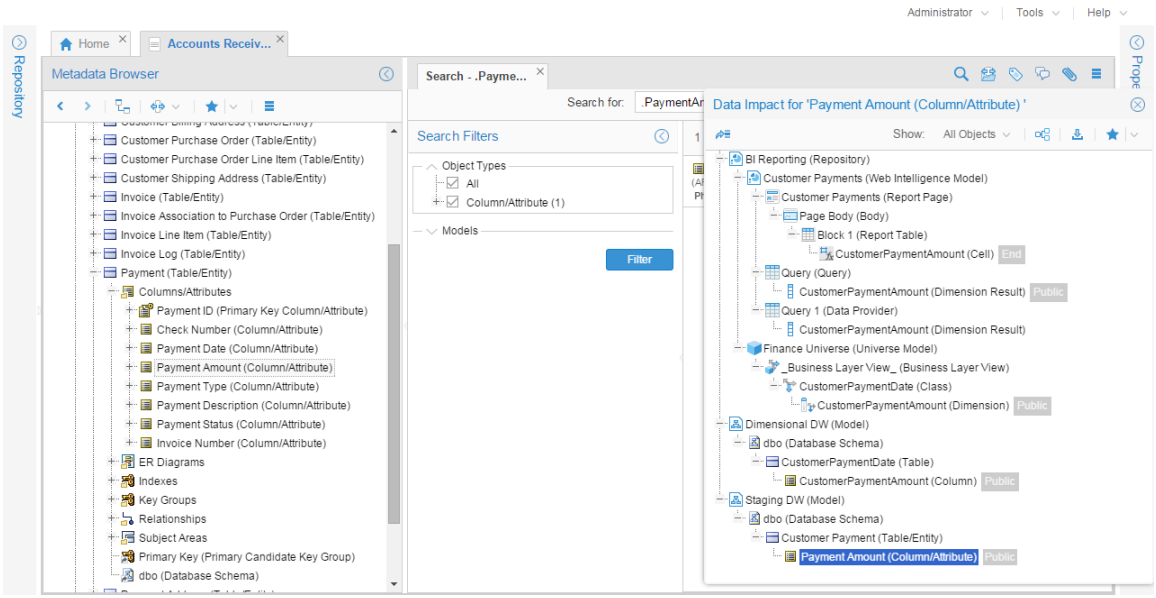


Figure 77 - Data impact for payment amount

Then we right-click on the one of the impacted attributes in the Staging DW PDM model and select Show in Metadata Browser. Then right-click on that element and select Trace Semantic Definition. Be sure to use the text report. This will trace semantic lineage from this point back up to the Enterprise Conceptual Model.

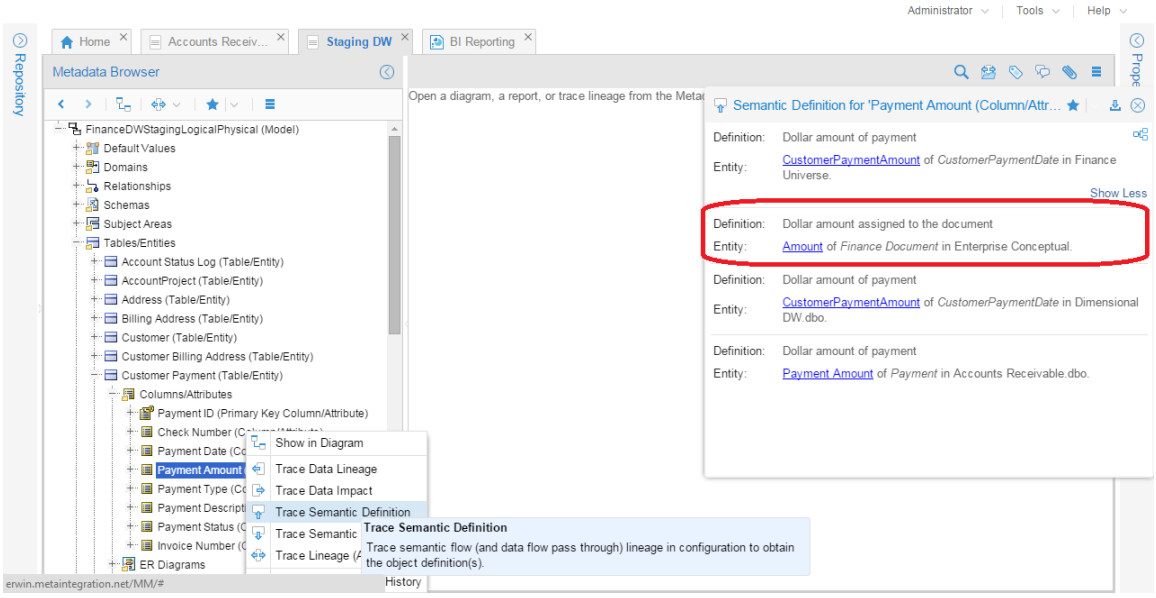


Figure 78 - Trace semantic definition

The result of this semantic lineage trace is a report of those elements in the conceptual model that are *semantically related* to Payment Amount in the Staging DW PDM model.

In this case, we are looking for the canonical form of the Payment Amount, and it is an attribute in the Enterprise Conceptual Model by the name of Amount in the class Finance

Document and thus we would want to trace the lineage back down to ALL of the related columns in the Staging DW PDM model.

Thus, tracing the semantic lineage forward to the physical models by right-clicking and selecting Trace Semantic Usage we have a report including all of the columns which must be updated to reflect the new Decimal(10,2) data type.

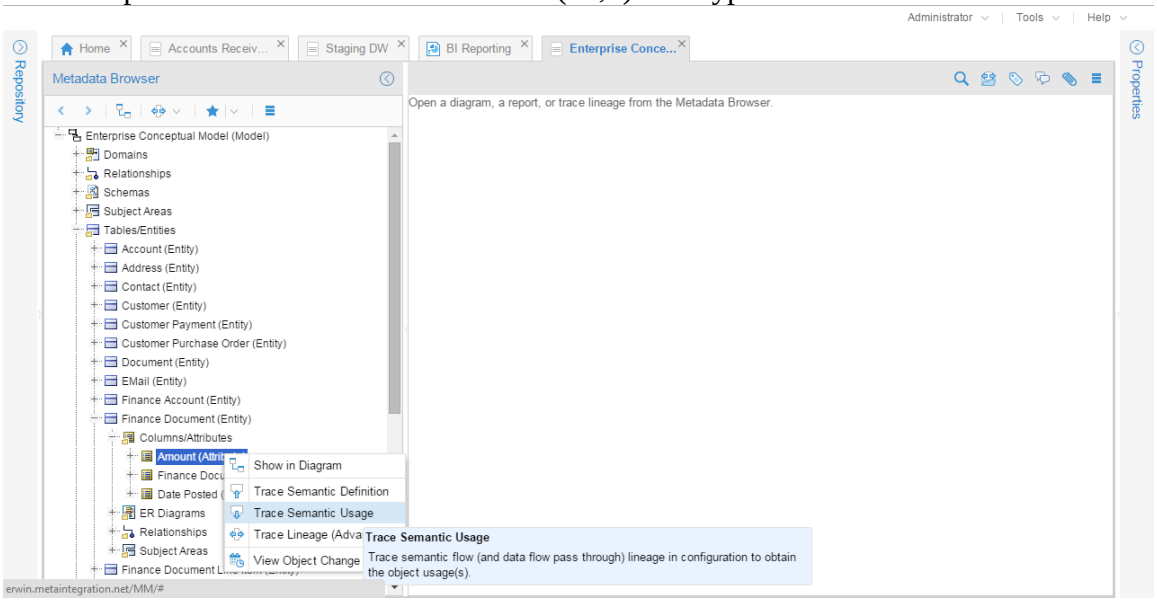


Figure 79 - Trace semantic usage

Be sure to select the correct (latest) configuration:

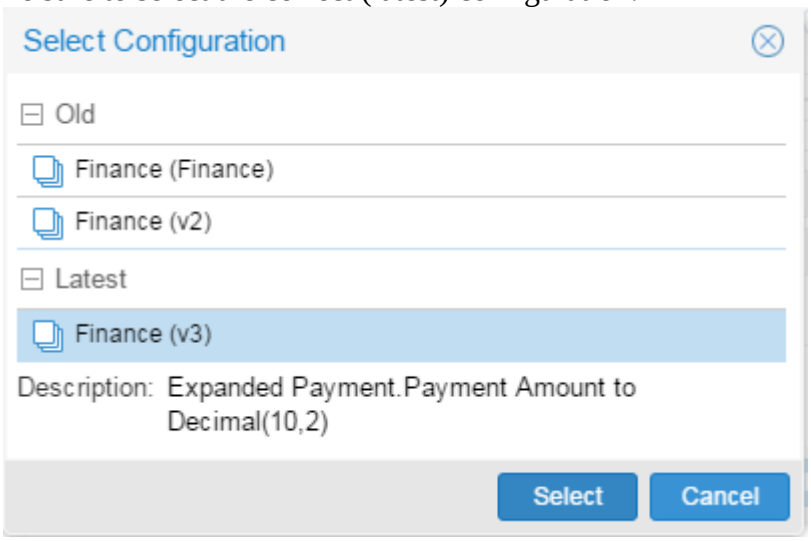
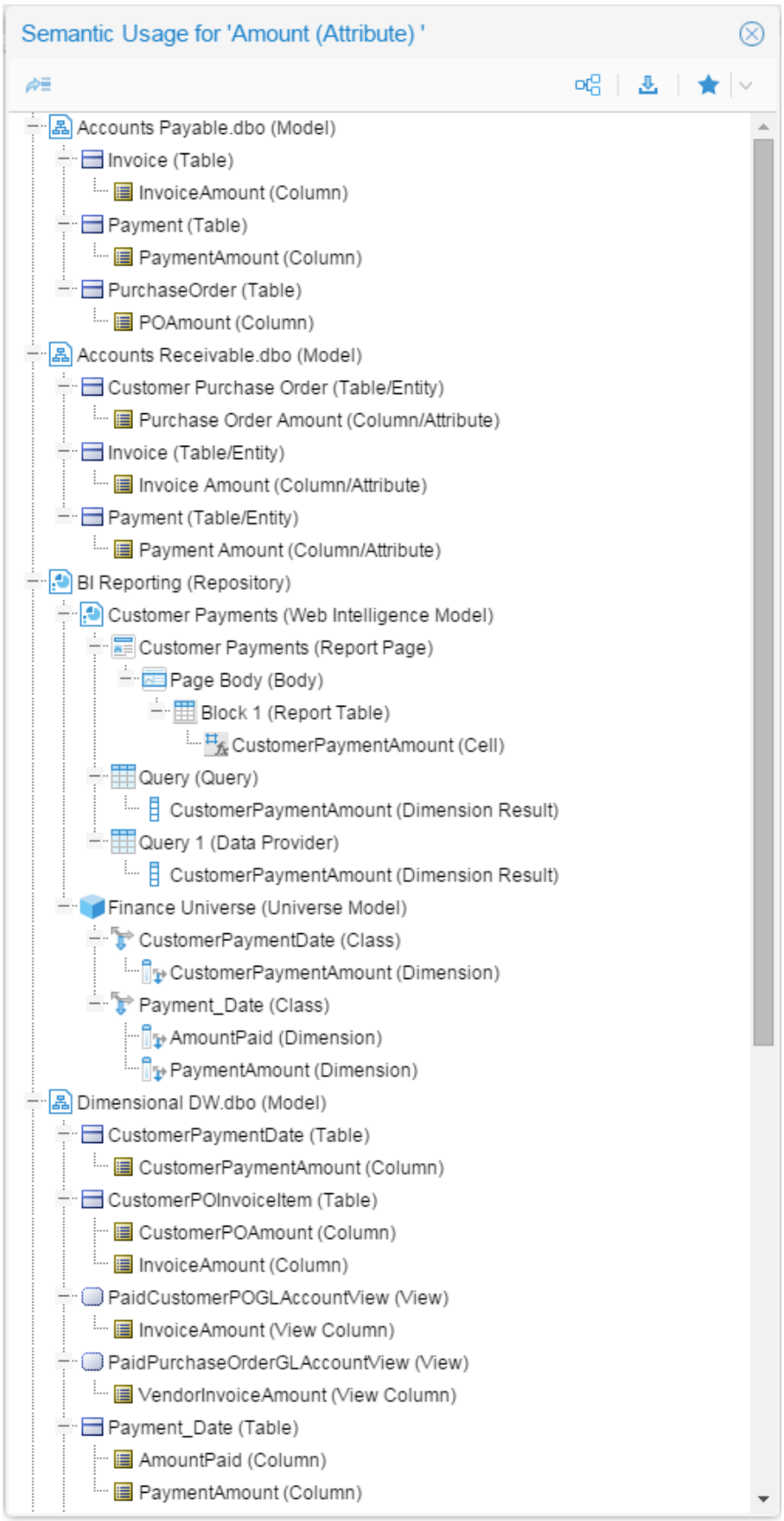


Figure 80 - Select Configuration

And you have:



To complete the closure or transitivity exercise, one could then trace the data flow lineage (forward and back) to identify which features in other models must also be updated to reflect the larger amounts. Note, the result above already included the data flow impact and lineage.

Better yet, the analyst for the warehouse group would update the design of the Staging DW PDM database and the Repository PoC for that group would upload the new version. The analyst for the entire configuration would then migrate the configuration and perform a comparison and obtain and impact report, just as in the last section. This report would then be used to determine the cost and schedule the update activities for all impacted systems. The new models would be provided, harvested and then configurations migrated and published.

This activity is left as an exercise for the reader. For the purposes of the tutorial, these updates are in the v4 directories. Copy the contents of

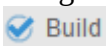
C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v4

into

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\Dev

and you will be ready to harvest the updated models into the Dev section of the Repository, and migrate the impacted configurations. Again, all of the naming and description information for version is provided in the *Summary of Configuration Changes* table at the end of this chapter. These are:

Content Name	Version Name	Version Description
Accounts Payable	v2	Expanded all amount fields to standard Decimal(10,2)
Accounts Receivable	v3	Expanded all amount fields to standard Decimal(10,2)
Dimensional DW	v2	Expanded all amount fields to standard Decimal(10,2)
Staging DW	v3	Expanded all amount fields to standard Decimal(10,2)
AP To Staging	v2	Expanded all amount fields to standard Decimal(10,2)
AR To Staging	v3	Expanded all amount fields to standard Decimal(10,2)
Staging To Dimensional	v2	Expanded all amount fields to standard Decimal(10,2)

Please import all those models and assign Name and Description. Then Migrate the configuration version forward and 

the configuration version. You should rename the configuration version to v4 with the Description of Expanded all amount fields to standard Decimal(10,2).

Be sure to mind the PDM and ensure that the correct version is archived.

You now have a completed configuration version of your architecture after the changes caused by the data type change in PAYTRANS.

Thus, it is time for the Development Team to migrate all their system changes to the Prod environment. To simulate this process, these updates are in the v4 directories. Copy the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v4

into

C:\Temp\Models\MetadataManagement\Finance

Once that is done, we may re-harvest the Finance Prod configuration (right-click on latest version, create a new version, migrate) just as with Finance Dev.

Now, Publish this new Finance Prod configuration version.

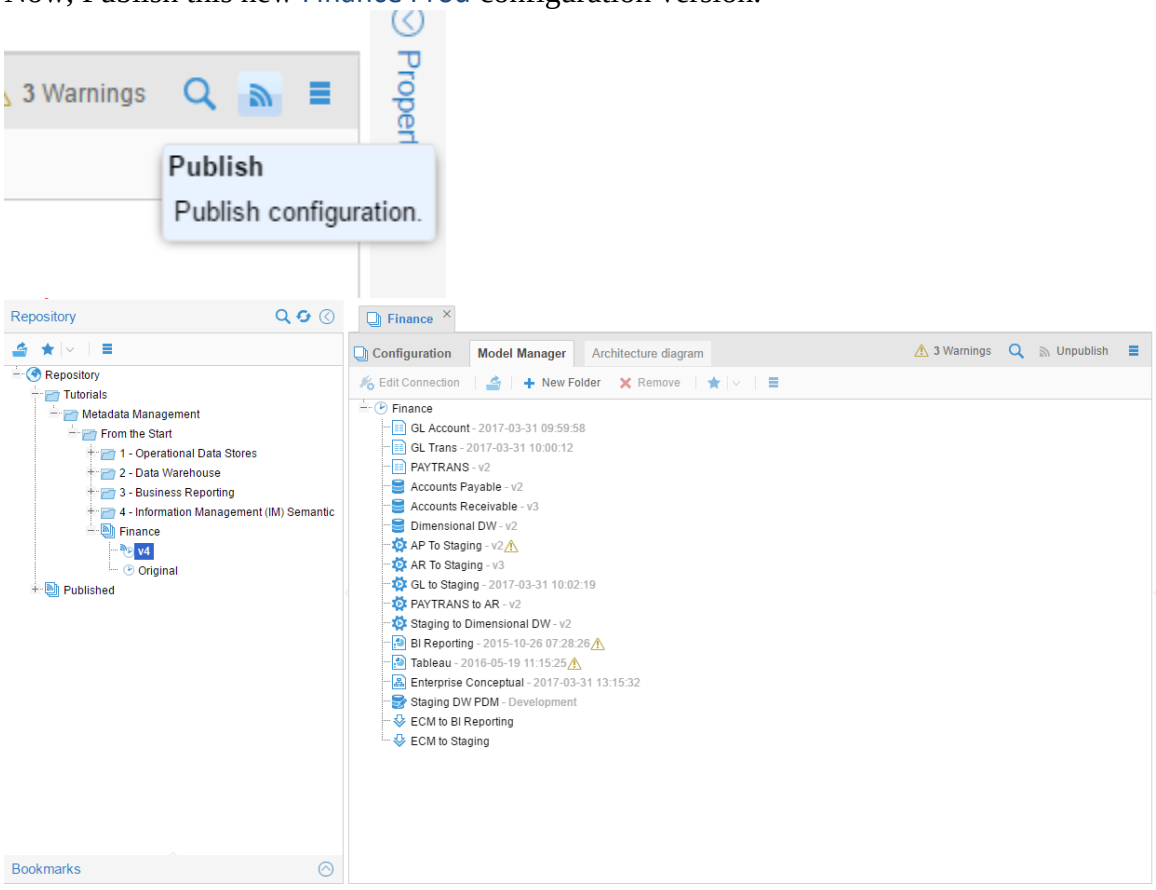


Figure 81 - Published configuration version

4.4.3 Create Historical Archive Version

From here on in this Tutorial document, we will work solely in the Finance Prod environment.

Now, as this is likely to be a milestone, let's archive a version for historical reference as we did at the beginning of this chapter.

Expand the Finance configuration in the Repository panel, right-click on the v4 version and select Create A Copy.

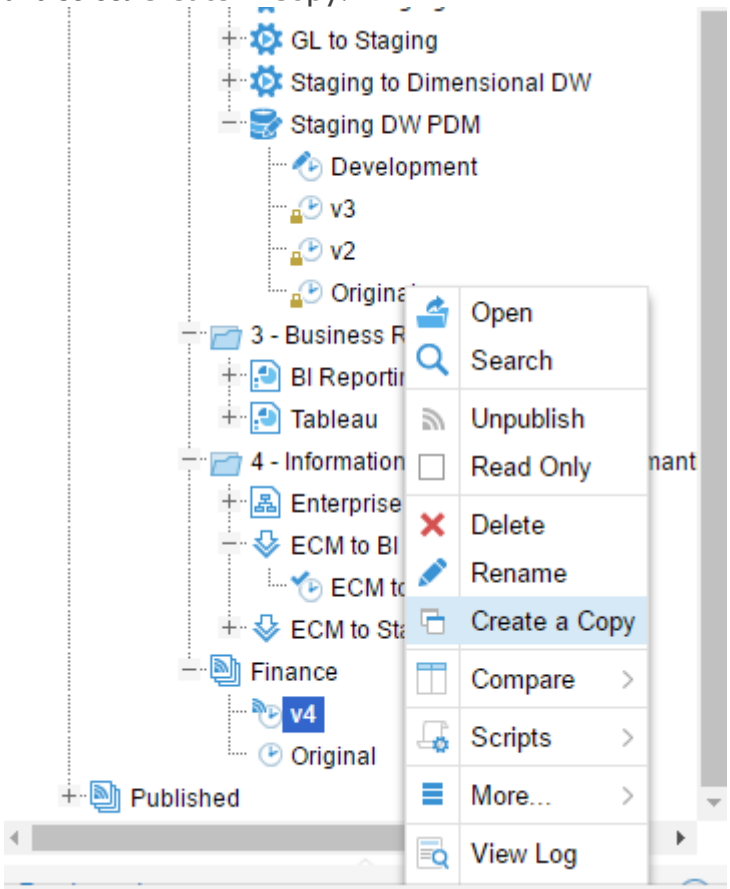


Figure 82 - Create new version of a configuration

Name this version "Development". This new version should be described by " Archived after Expanded all amount fields to standard Decimal(10,2)".

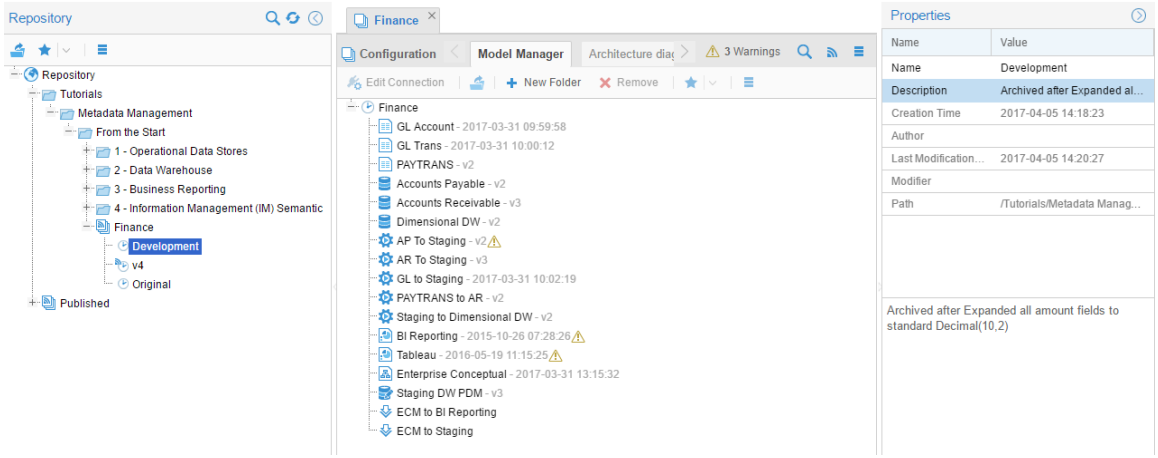


Figure 83 - Development configuration version

We already have a read-only (frozen) version of the Staging DW PDM. So, we can simply drag the v3 version of the Stating DW PDM (that you just created earlier by harvesting its lightweight model) into the v4 version of the Finance configuration.

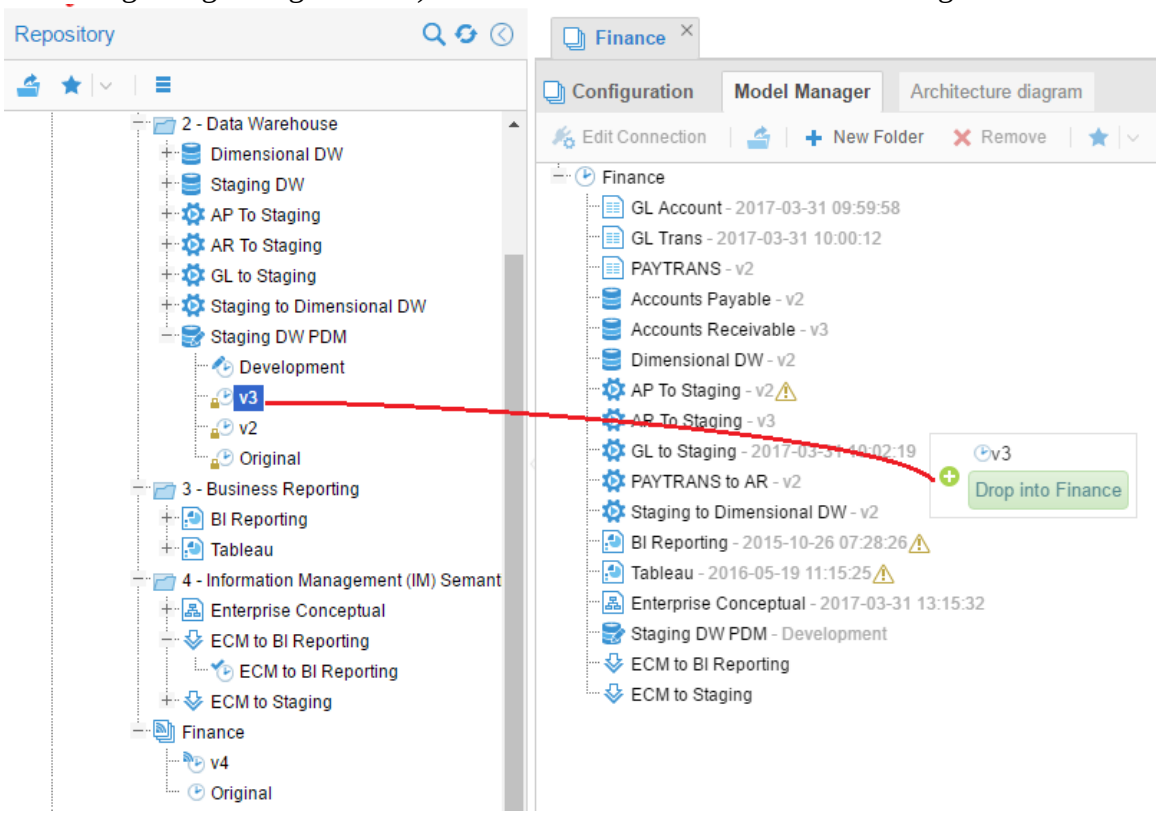


Figure 84 - Drag into the configuration

You will see the following dialog:

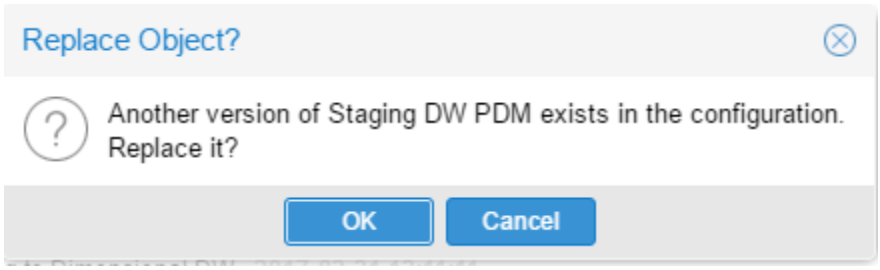


Figure 85 - Replace object

As we are replacing the Development version (editable version) of the Staging DW PDM with this archive version, Meta Integration® Metadata Management (MIMM) simply wants confirmation. Now, click on the OK button.

We now have an historical configuration recording the v4 state of the configuration. Be sure to right-click on this v4 version and select Read Only.

4.5 Inconsequential Impacts

Often times, changes which are captured through harvesting have no impact on downstream system. An example like this is provided in the scenarios outlined in the Summary of Configuration Changes at the end of this section.

For the purposes of the tutorial, these updates are in the v5 directories. Copy the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions \v5

into

C:\Temp\Models\MetadataManagement\Finance\

and agree to the replacement of files. Now import the impacted models into Meta Integration® Metadata Management (MIMM) using either the manual process or automatic scripts and assigning names and descriptions as defined in the table at the end of this section.

Content Name	Version Name	Version Description
Dimensional DW	v3	Nothing Substantive
Staging DW	v4	Nothing Substantive

Now, migrate the Complete System Configuration and compare this new version with the previous version. A little investigation will show that not substantive changes have been made. This is an example of an inconsequential impact, and the new configuration version may simply be published so that the new descriptive information is available to the Explorer UI.

Nevertheless, we migrate and publish this configuration version (named Development) as it is the latest valid architecture.

4.6 Architecture Extensions

The v6 updates in the Summary of Configuration Changes table at the end of this section contains a true architecture change. In this case, a new component of the Payroll System is added, one which the Repository is unaware of. In this case it is necessary to extend the configurations representing the Finance System, by first creating new models, harvesting the new metadata and then dragging these new models into the configurations and stitching as appropriate.

For the purposes of the tutorial, these updates are in the v6 directories. Copy the contents of

C:\Temp\Models\MetadataManagement\Finance_ConfigurationVersions\v6

into

C:\Temp\Models\MetadataManagement\Finance

and agree to the replacement of files. Now import the impacted models into Meta Integration® Metadata Management (MIMM) using either the manual process or automatic scripts and assigning names and descriptions as defined in the table at the end of this section.

Content Name	Version Name	Version Description
GL Category	v1	Added Account Category Entities/Attributes
Accounts Receivable	v4	Added Account Category Entities/Attributes
Dimensional DW	v4	Added Account Category Entities/Attributes
Staging DW	v5	Added Account Category Entities/Attributes
AP To Staging	v3	Added Account Category Entities/Attributes
AR To Staging	v4	Added Account Category Entities/Attributes
Staging To Dimensional	v3	Added Account Category Entities/Attributes
GL To Staging	v2	Added Account Category Entities/Attributes

First, create a new model content name GL Category in the General Ledger [folder].

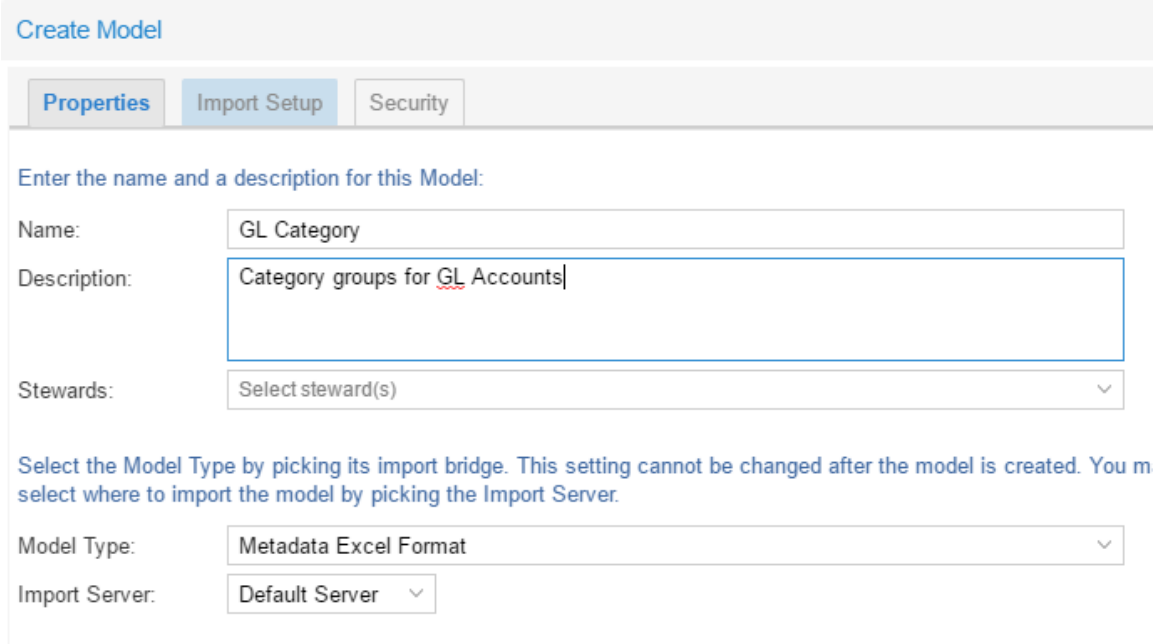


Figure 86 - Adding the GL Category model content

Now complete the import bridge parameters on the Import Setup tab.

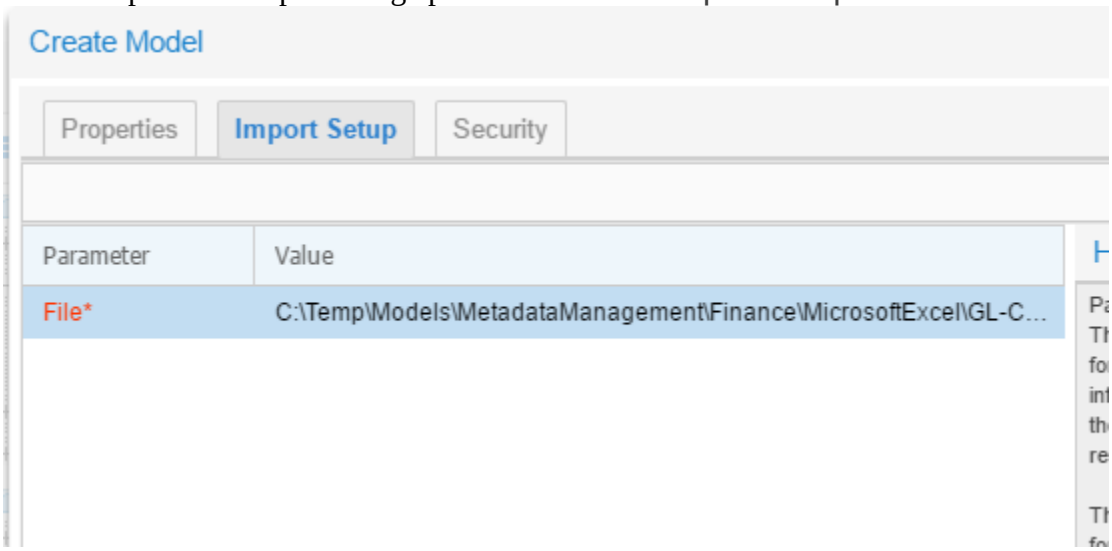


Figure 87 - Import parameters for the General Ledger Account Category model content

The location is C:\Temp\Models\MetadataManagement\Finance\MicrosoftExcel\GL-Category.xlsx

Now, import the model.

Opening the Finance Configuration and dragging the new model version into it and then stitching it to the existing ETL model (General Ledger to Finance Data Warehouse Staging ETL) would appear to be the next step.

However, one cannot define a connection because one is not specified in the source ETL, as the ETL model is the OLD model which only reads from the GL Account and GL Transaction data.

Thus, the proper steps would be to:

1. Create new model content and import model for first time (already done)
2. Harvest the other models which have been updated due to the architecture changes (see the table at the end of this section of a complete list)
3. Migrate the Development configuration version so it contains the updated models
4. Add the new model to the Development version of the configurations, stitch to the new ETL model (that was just harvested) and save those updates
5. Publish when ready.

Now you have a new architecture represented in the latest version of the configuration in your Repository. You may now simply migrate as new versions are harvested until another true architecture change occurs.

Be sure to stitch the new connection:

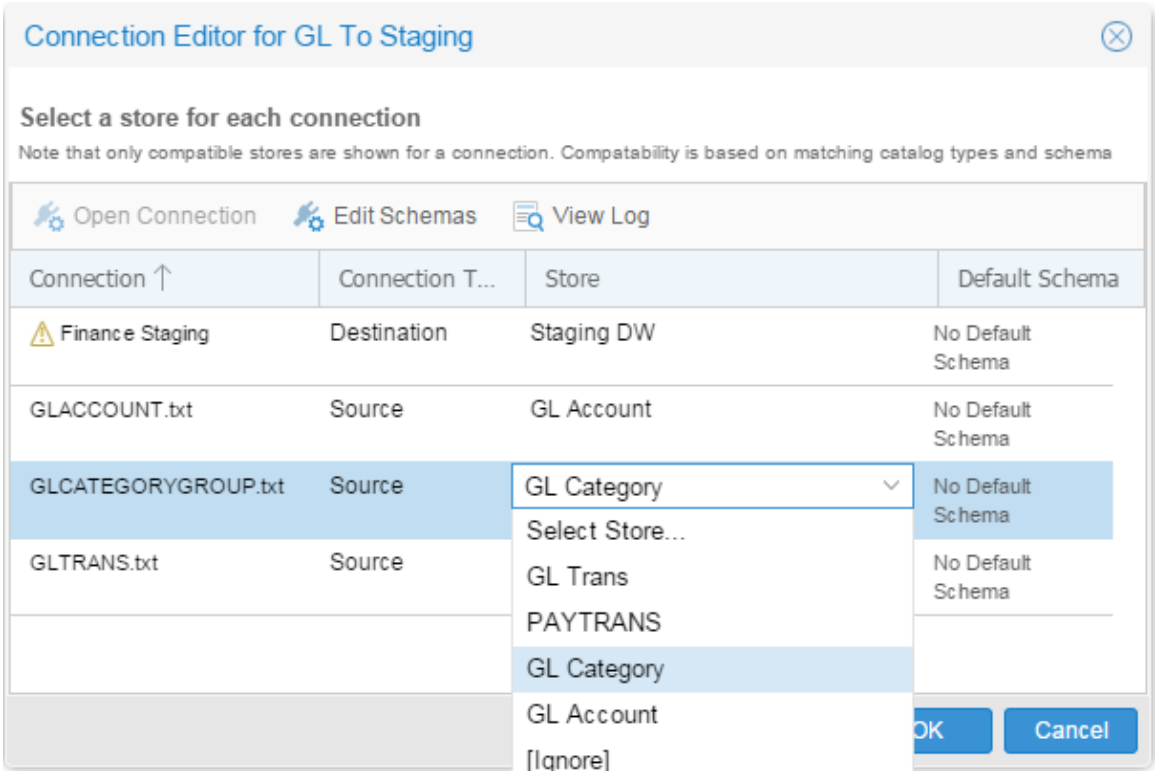


Figure 88 - Stitching connection to account category

Here is the final result:

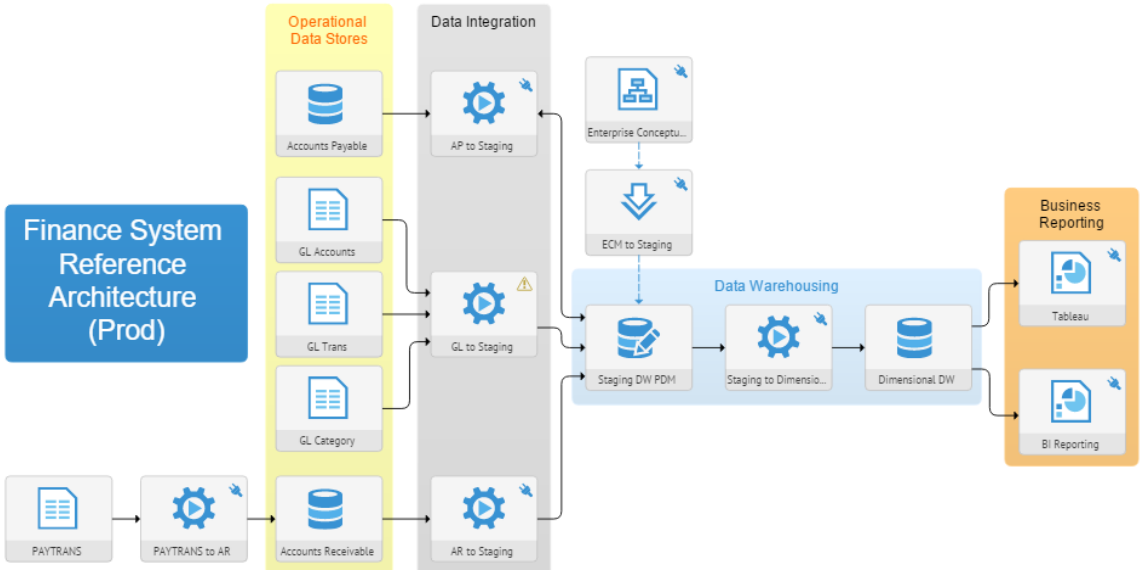


Figure 89 - Final result of this section in the repository

That is the end of the version management chapter. Please create one more version (v6) that is a read-only historical version of the "Final result after the version management chapter", and be sure it has the latest read-only version of the Staging DW PDM in it and also be sure to right-click on this v6 version and select Read Only.

However, publish the Development version.

4.7 Summary of Configuration Changes

The following matrix shows the configuration versions, the version of the models to import and the reasons for the changes.

Summary of Configuration Changes							
Red "X" indicates new version of model for that version of the configuration	**Original Version**	**Expanded PTAMT to Decimal(10,2)**	**Expanded Payment.Payment Amount to Decimal(10,2)**	**Expanded all amount fields to standard Decimal(10,2)**	**Added Descriptions to GL Entities and Attributes**	**Added Account Category Flat file**	**Added Partial Over Payment Type**
	v1	**v2**	**v3**	**v4**	**v5**	**v6**	**V7**
Data Warehouse Staging Versions							
v5 - Added Account Category Entities/Attributes						X	X
v4 - Added Descriptions to GL Entities/Attributes					X		
v3 - Expanded all amount fields to standard Decimal(10,2)				X			
v2 - Expanded CustomerPayment.Payment Amount to Decimal(10,2)			X				
v1 - Original	X	X					
Data Warehouse Dimensional Versions							
v4 - Added Account Cetegory Entities/Attributes						X	X
v3 - Added Descriptions to GL Entities/Attributes					X		
v2 - Expanded all amount fields to standard Decimal(10,2)				X			
v1 - Original	X	X	X				

AR Versions							
V5 - Added Partial Over Payment Type							X
v4 - Added Account Category Attributes						X	
v3 - Expanded all amount fields to standard Decimal(10,2)				X	X		
v2 - Expanded Payment.Payment Amount to Decimal(10,2)			X				
v1 - Original	X	X					
AP Versions							
v2 - Expanded all amount fields to standard Decimal(10,2)				X	X	X	X
v1 - Original	X	X	X				
BI Versions							
v1 - Original	X	X	X	X	X	X	X
GLAccount							
v1 - Original	X	X	X	X	X	X	X
GLAccountCategory							
v1 - Original	X	X	X	X	X	X	X
GLAccountTransaction							
v1 - Original						X	X
PAYTRANS Versions							
V3 - Added Partial Over Payment Type							X
v2 - Expanded PTAMT to Decimal(10,2)		**X**	X	X	X	X	
v1 - Original	X						

PAYTRANS2AR Versions							
v3 - Added Partial Over Payment Type							X
v2 - Expanded Customer Payment.Payment Amount and PTPMT to Decimal(10,2)			X	X	X	X	
v1 - Original	X	X					
AR2Staging Versions							
v4 - Added Account Category Attributes						X	X
v3 - Expanded all amount fields to standard Decimal(10,2)				X	X		
v2 - Expanded CustomerPayment.Payment Amount to Decimal(10,2)			X				
v1 - Original	X	X					
AP2Staging Versions							
v3 - Added Account Category Attributes						X	X
v2 - Expanded all amount fields to standard Decimal(10,2)				X	X		
v1 - Original	X	X	X				
GL2Staging Versions							
v2 - Added Account Category						X	X
v1 - Original	X	X	X	X	X		
DWS2DWD Versions							
v3 - Added Account Cetegory Entities/Attributes						X	X
v2 - Expanded all amount fields to standard Decimal(10,2)				X	X		
v1 - Original	X	X	X				