ONE - Workshop

ONE Desktop Integration

Prepared for:	v15.4.x
Prepared by:	Ataccama
Dated:	November 2024

Contents of the Document

Introduction

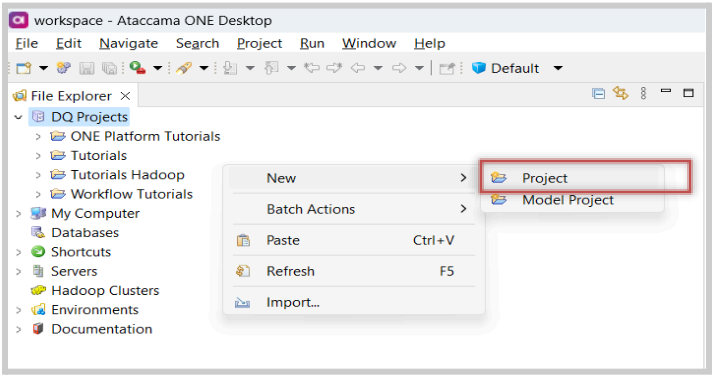
In this workshop, you will explore advanced Data Quality scenarios in the ONE Web application. To implement these scenarios, you will need to integrate it with the ONE Desktop application, as connecting the two applications unlocks many features. The scenarios we will cover in this lab include reading the data of a catalog item, reading the metadata of a catalog item, remotely writing a term, and creating a virtual catalog item.

Tasks

There are 6 tasks in this workshop. Before proceeding with the tasks, please pay attention to the following points:

- ONE Desktop application v15.4 should be installed locally on your machine.
- Completing task 1 is a prerequisite for the remaining tasks.
- For Tasks 2 to 5, you will need a project folder in your file explorer to store the required Desktop plans. If you already have one (e.g., from the ONE Desktop course training), you can use it. Otherwise, you will need to create one. If you are **unsure** how to do this, refer to the ONE Desktop course or follow the **instructions below, otherwise skip**:

  › In the **File Explorer**, right-click on the **DQ Projects**, choose **New** and then **Project**.



  › Choose **Project from template** as **Project content** and select the **Simple project** as a template and give it a name, e.g. "ONE Web Training". If you have a Training project from the ONE Desktop Training course, you can use that.

NOTE *Before doing any transformations or analysis of data, you need to create a project in the Desktop. A **project** is a set of multiple plans, components, data files, workflows, and custom scripts, which are usually focused on solving a specific task and are logically organized into folders. Projects and their hierarchies are physically stored in a workspace folder.*

  › Click on **Finish**. The project will appear under **DQ projects**

  › In order to have a dedicated plans folder, you can right click your newly created project, select **new**, then **folder** and name it **plans**.

  › In order to create a plan, right click the plans folder, select **new**, then **plan** and provide a name.

NOTE *A **plan** file defines the logic and rules to be applied to the input data in order to produce the desired output. Plans are created by placing **steps** and components onto the canvas in ONE Desktop and connecting them together.*

1. Connecting the ONE Desktop to the ONE Web application

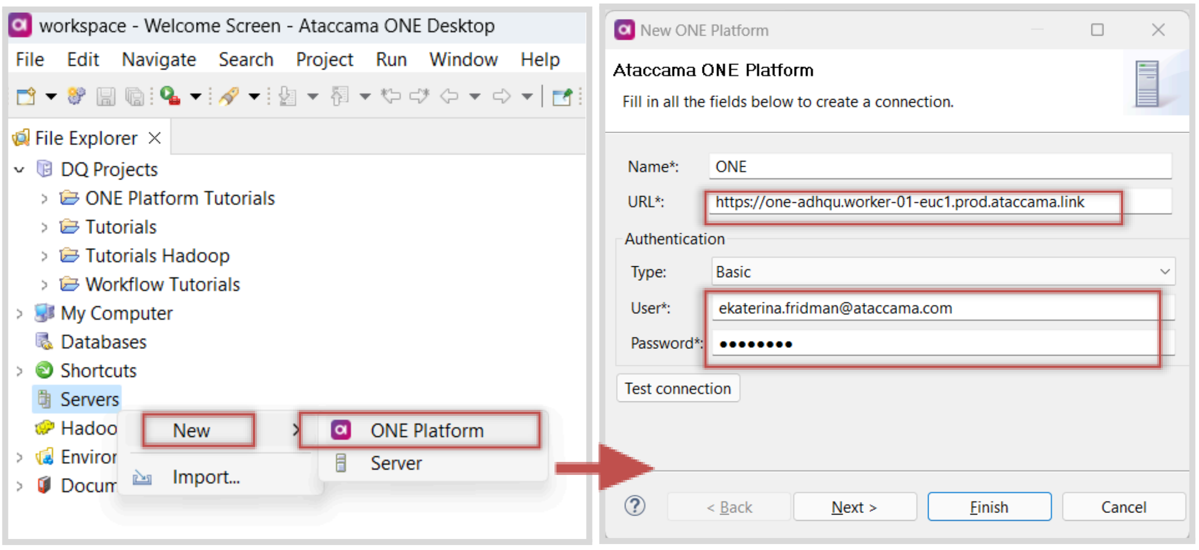
In the first task, you will connect your ONE Desktop application to the ONE Web application:

- Navigate to the root of the [build] folder and open the ONE Desktop application via the file **one-desktop.exe** (or a similar file depending on your OS)
- Once launched, make sure you have ONE **perspective** chosen.

HINT *To switch between the perspectives, use the icon in the upper right corner.*

- In the ***File Explorer*** view, right-click the **Servers** node and press **New**>**ONE Platform**.

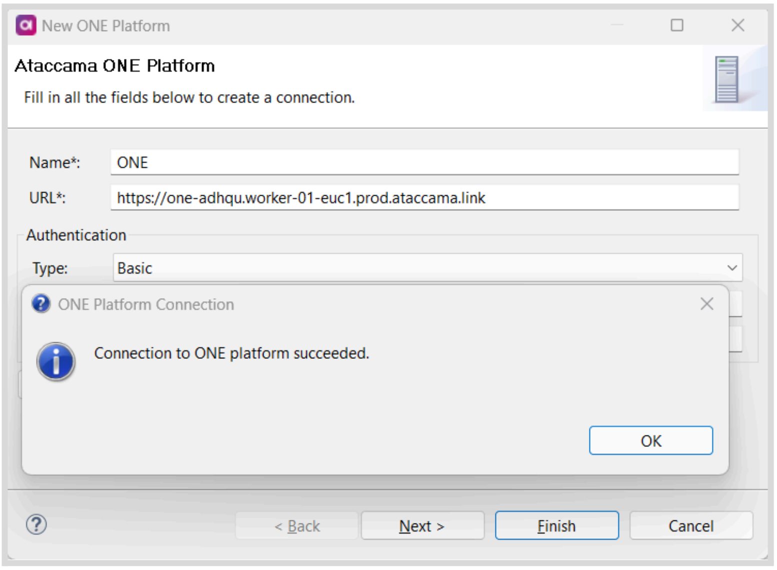
A new window will open for you to specify the preliminary connection details.



Ataccama ONE Platform
Fill in all the fields below to create a connection.

Name*: ONE
URL*: https://one-adhqu.worker-01-euc1.prod.ataccama.link
Authentication
Type: Basic
User*: ekaterina.fridman@ataccama.com
Password*: ••••••••

- Fill in the **URL** as well as the **Authentication** provided to you and give it a **Name**.
- Click **Test connection.**



Ataccama ONE Platform
Fill in all the fields below to create a connection.

Name*: ONE
URL*: https://one-adhqu.worker-01-euc1.prod.ataccama.link
Authentication
Type: Basic

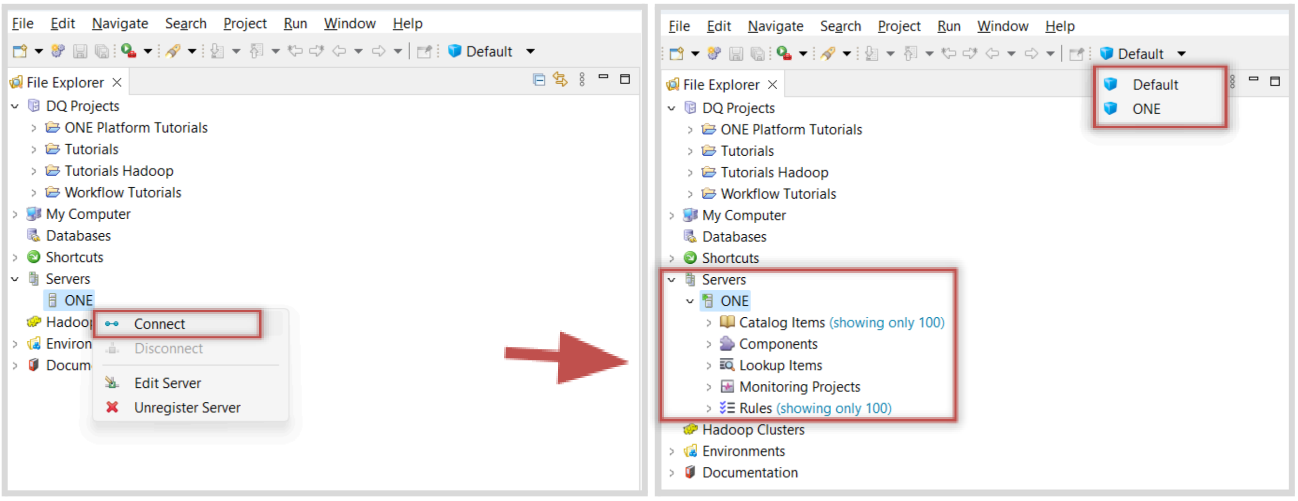
Connection to ONE platform succeeded.

HINT *You can verify whether all has been set up properly via the **Test Connection** button.*

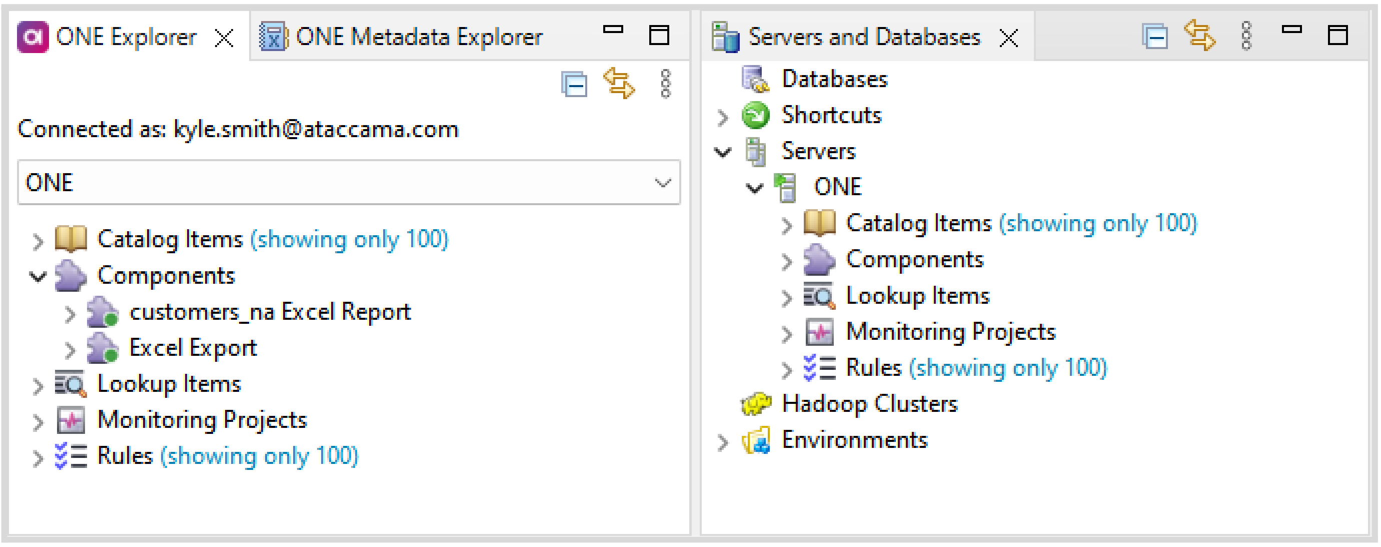
- Complete your server configuration and close the window by pressing the **Finish** button.

Now that the server has been configured, you can access and start using it:

- To connect to the server, right-click on the name of your newly created connection (‘***ONE***’ in this example) and press **Connect**.



- The connection will expand to a hierarchy of items following the structure of your ONE Web Application. You should now see your Catalog Items and other objects from the ONE Web application in the **ONE Explorer** tab.



2. Reading data from a Catalog Item

In this task we will use a special IDE step - **Catalog Item Reader** to read data from a Catalog Item - **customers**. This could be useful when we want to use the data from a catalog item in a ONE Desktop plan.

› Go to your project and create a new plan, e.g., **download_data.plan**.

The first step we need to insert is the **Catalog Item Reader**. It will help us consume the ONE Web application's catalog data and use it in the plan.

This step can be added in two ways:

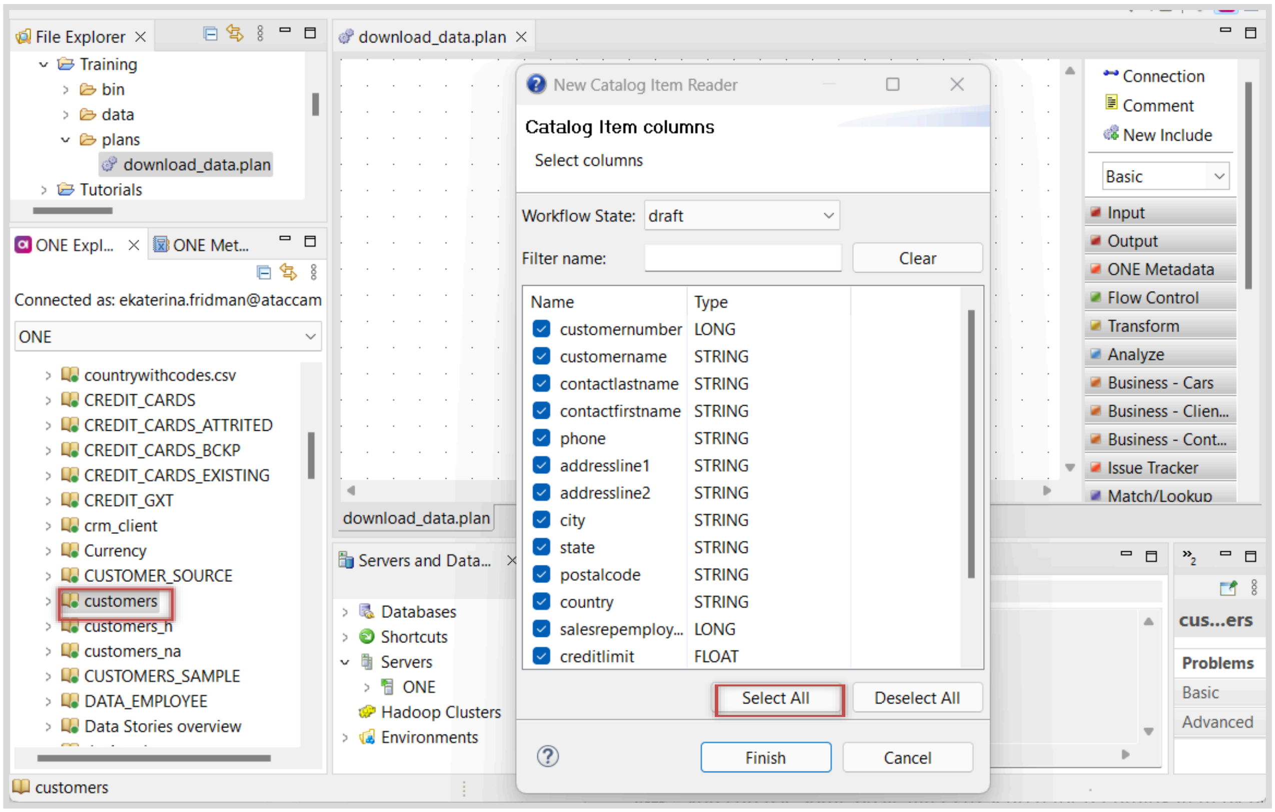
- Insert and configure the step from scratch.
- Drag and drop the desired Catalog Item to the canvas directly from the catalog items list under the server connection/ONE Explorer tab.

Let's choose the second option to make things easier:

› From the **ONE Explorer**, drag the **Customers** Catalog Item and drop it into the plan's canvas.

From the window that appears you can filter out some columns. For now, we'll keep all the columns so that the whole structure can be read.

› Press **Finish** to complete the insertion of the new **Catalog Item Reader** step.



Connected as: ekaterina.fridman@ataccam

New Catalog Item Reader

Catalog Item columns

Select columns

Workflow State: draft

Filter name:

Clear

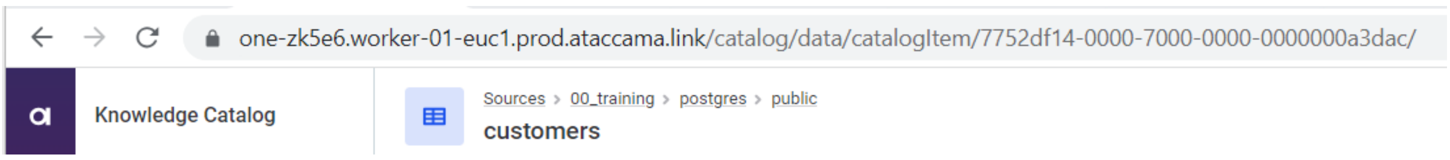
Name	Type
✓ customernumber	LONG
✓ customername	STRING
✓ contactlastname	STRING
✓ contactfirstname	STRING
✓ phone	STRING
✓ addressline1	STRING
✓ addressline2	STRING
✓ city	STRING
✓ state	STRING
✓ postalcode	STRING
✓ country	STRING
✓ salesrepemploy...	LONG
✓ creditlimit	FLOAT

Select All | Deselect All

Finish | Cancel

NOTE

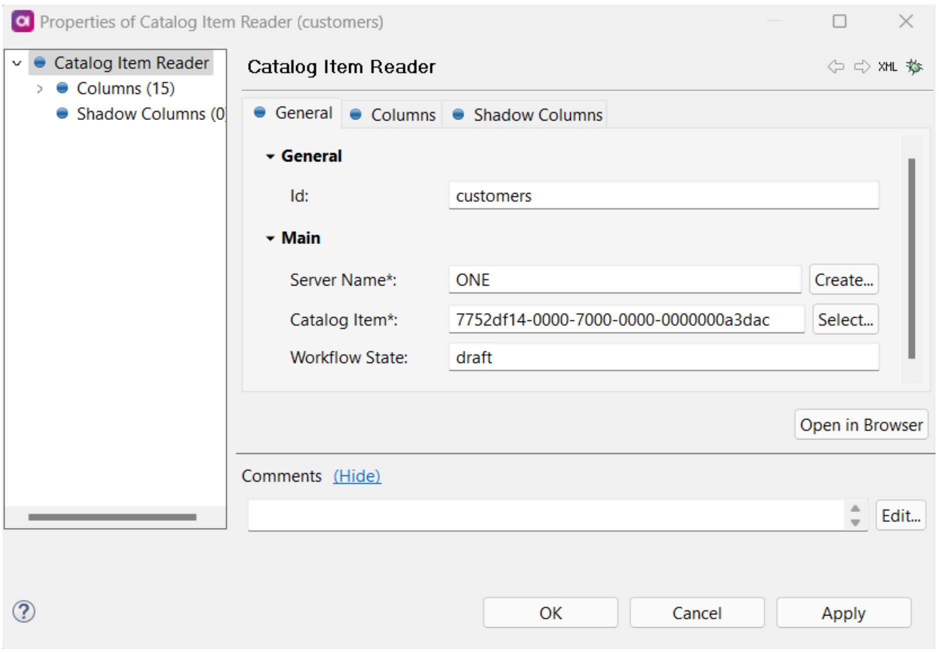
*If you configure the step manually, once you fill in the **Server Name** (e.g. **ONE**), you can search for the catalog item using the button **Select**. A window will open where you can use some basic filters to search for a Catalog Item by name or path and you can also specify whether it is a table catalog item, file, virtual catalog item. etc. After the selection is done, the **unique ID** of the catalog item will be filled.*


one-zk5e6.worker-01-euc1.prod.ataccama.link/catalog/data/catalogItem/7752df14-0000-7000-0000-0000000a3dac/

Knowledge Catalog

Sources > 00_training > postgres > public

customers


Properties of Catalog Item Reader (customers)

- Catalog Item Reader
  - Columns (15)
  - Shadow Columns (0

Catalog Item Reader

General | Columns | Shadow Columns

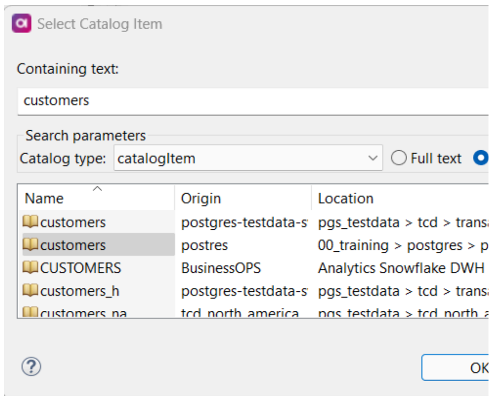
General

Id:	customers

Main

Server Name*:	ONE
Catalog Item*:	7752df14-0000-7000-0000-0000000a3dac
Workflow State:	draft

Comments (Hide)


Select Catalog Item

Containing text: customers

Search parameters

Catalog type: catalogItem

Name	Origin	Location
customers	postgres-testdata-s	pgs_testdata > tcd > trans
customers	postres	00_training > postgres > p
CUSTOMERS	BusinessOPS	Analytics Snowflake DWH
customers_h	postgres-testdata-s	pgs_testdata > tcd > trans
customers_na	tcd_north_america	pgs_testdata > tcd_north_a

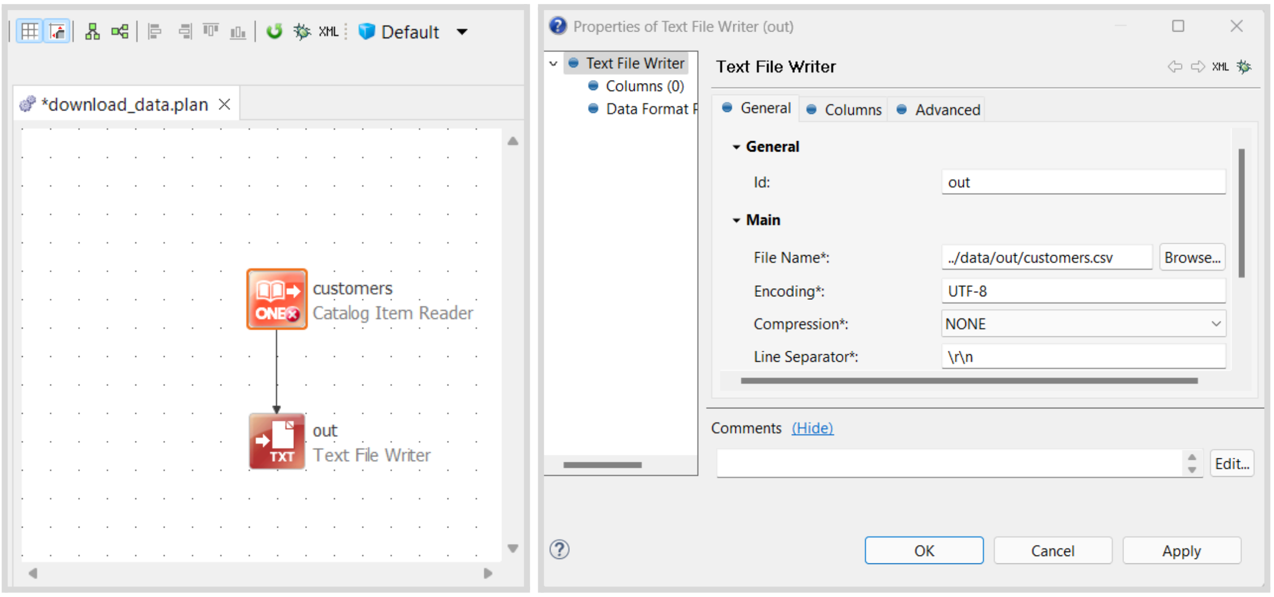
Let’s review the configuration options for the **Catalog Item Reader** step:

- Open the newly created step **customers** and make sure you are on the ***General*** tab.

Notice the **Workflow State** value. By default, it is set to **Draft**, which means that the latest version will be downloaded, even the one in the draft.

We need an output step to save the data. For now, a simple ***.csv*** file output will be enough:

- Add a**Text File Writer** step and connect it to the ***‘out’*** of the **Catalog Item Reader.**
- Fill in the **File name** (e.g., **customers.csv**) and place the file in the **data/out** folder.



Properties of Text File Writer (out)

Text File Writer

General | Columns | Advanced

General

Id: out

Main

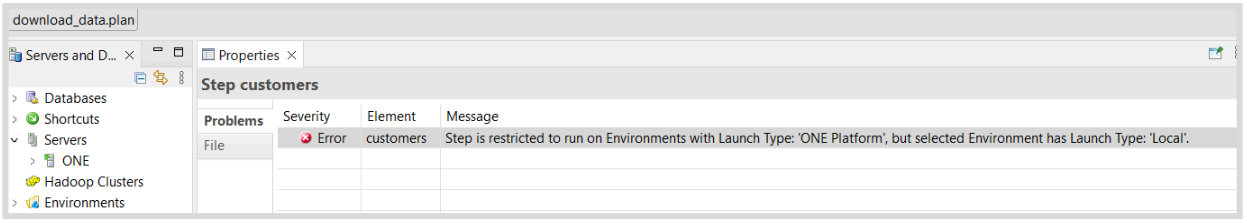
File Name*: ../data/out/customers.csv

Encoding*: UTF-8

Compression*: NONE

Line Separator*: \r\n

There is still an error sign on the **Catalog Item Reader**. If you click on it and look at the **Properties** tab at the bottom of your interface, you will see the following error message.

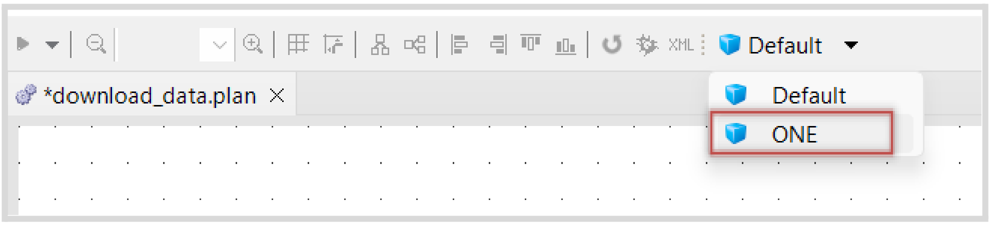


Step customers

Severity	Element	Message
Error	customers	Step is restricted to run on Environments with Launch Type: 'ONE Platform', but selected Environment has Launch Type: 'Local'.

In the first task of this lab, when creating the connection to ONE Web application, we created an **Environment** relevant to the server connection. To run this plan, we need to switch to this environment:

› Click on the arrow next to the **Default** environment and choose the **ONE** instead.



› **Save** the plan, **run** it and check the results in the Project's **data/out** folder.

3. Reading attributes of a catalog item

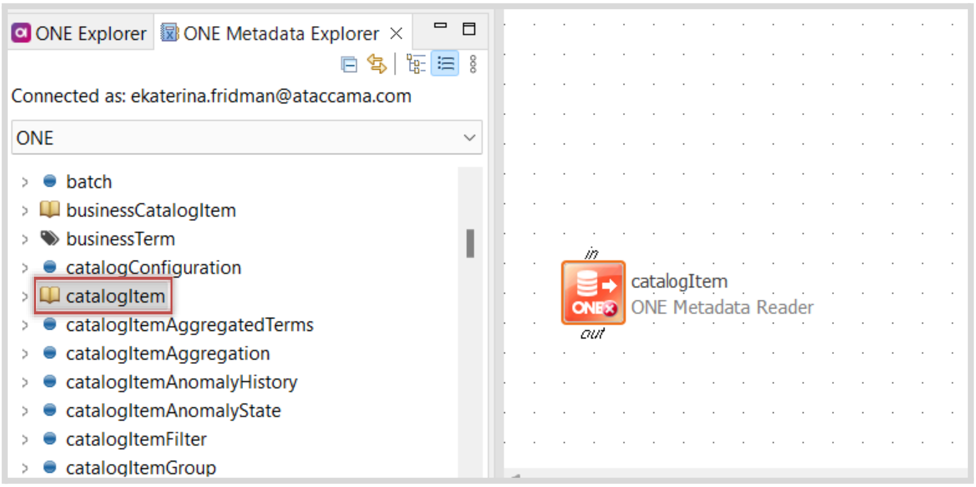
In this task, we are interested in reading and consuming the attributes of our catalog items. We want to download the catalog items and their attributes and save them as a ***CSV*** file. This can be achieved by using a new step - **ONE Metadata Reader**:

› Create a new plan in your project and call it **download_ci_info.plan.**

Like the previous scenario, we need to start with the configuration of the **ONE Metadata Reader** either through directly inserting the step or using the drag-and-drop functionality.

To drag and drop, you can be on either of the tabs (***ONE Explorer*** or ***ONE Metadata Explorer)***

- Find the **catalog Item** entity and drop it in the canvas of your new plan.



ONE Explorer | ONE Metadata Explorer

Connected as: ekaterina.fridman@ataccama.com

ONE

- batch
- businessCatalogItem
- businessTerm
- catalogConfiguration
- catalogItem
- catalogItemAggregatedTerms
- catalogItemAggregation
- catalogItemAnomalyHistory
- catalogItemAnomalyState
- catalogItemFilter
- catalogItemGroup

- Open the new **ONE Metadata Reader** step.

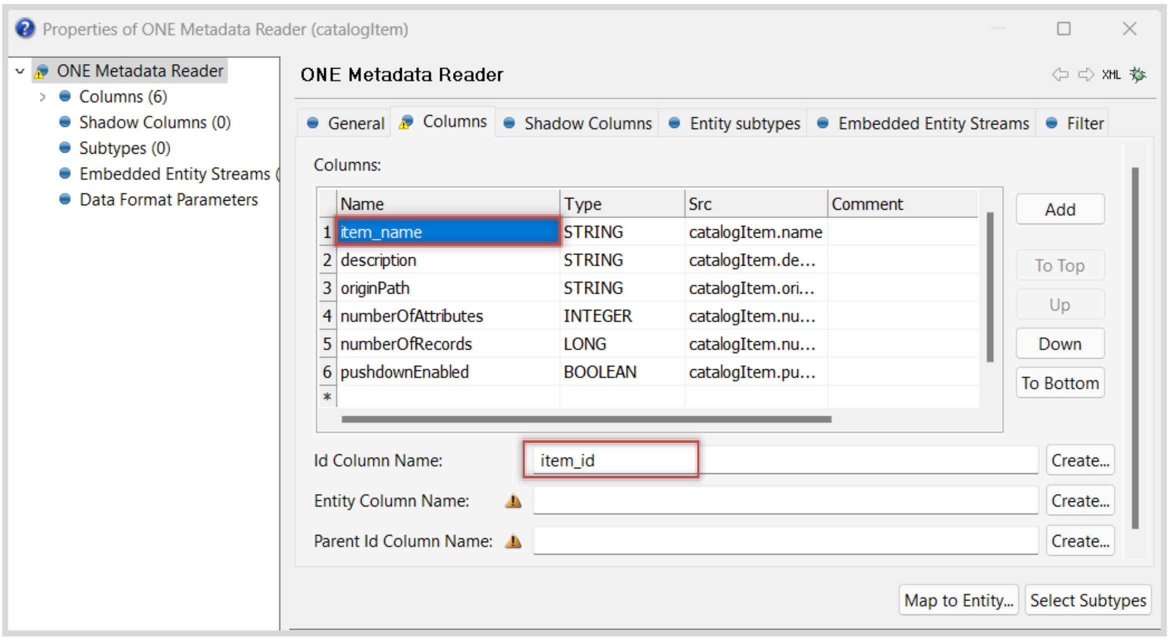
As you can see, some parts of the configuration are already filled in like the **Server Name** value or simple properties in the ***Columns*** tab.

Since we also want to download the attributes, we will rename the column name to item_name to distinguish between the name of the item and the name of the attributes.

- Go to the ***Columns*** tab and change the name of the column **name** to **item_name or CatalogItem_name**.

With a configuration like this, we would be able to download basic metadata about any of our catalog items. However, in this example, we want to modify the plan to go beyond basics - (reading the attributes). For that, we will need to remember each **catalog item id** so that we can then join the attributes with the relevant catalog item.

- While on the columns tab, populate a value for the **Id Column Name:** e.g. **item_id**.



Properties of ONE Metadata Reader (catalogItem)

ONE Metadata Reader

Columns:

	Name	Type	Src	Comment
1	item_name	STRING	catalogItem.name	
2	description	STRING	catalogItem.de...	
3	originPath	STRING	catalogItem.ori...	
4	numberOfAttributes	INTEGER	catalogItem.nu...	
5	numberOfRecords	LONG	catalogItem.nu...	
6	pushdownEnabled	BOOLEAN	catalogItem.pu...	
*				

Id Column Name: item_id

Entity Column Name:

Parent Id Column Name:

Attributes of a catalog item are **Embedded Entities** of the item. Therefore, to extract information about attributes of this catalog item we will create a new Embedded Entity, which will result in another output stream of the **ONE Metadata Reader** step.

- Switch to the ***Embedded Entity Streams*** tab.
- Add an entity stream item and name it **'attributes'**.
- Open its configuration by double-clicking on the first line or expanding the hierarchy tree on the left side.
- For the **Attribute** field, use the **content assist** to select an entity called **attributes**.

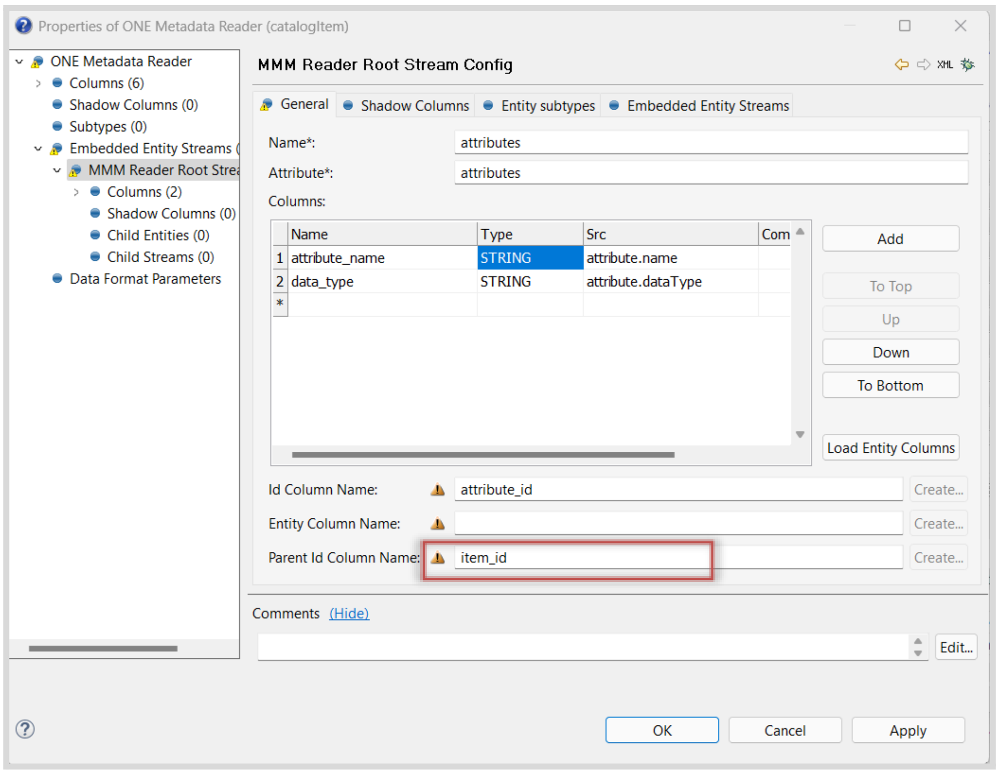
HINT *To display the **content assist,** press the ‚**CTRL**' + ‚**Space**' keys.*

Now we want to fill in columns for the new **attributes** entity stream item. We are interested only in the name of the attribute and its data type:

- Use the button **Load Entity Columns** to add the **name** and **dataType** columns.
- Rename the columns if you want, e.g., **attribute_name** and **data_type**.

If we leave the configuration like this, we will get a list of attributes for all catalog items in our Data Catalog but without the information about the catalog item the attributes belong to. Therefore, we need to capture the *Parent Entity Id*, where the parent entity in this case is the catalog item.

- Fill in the **item_id** attribute as a **Parent Id Column Name**.
- If you want, you can also specify attribute_id as Id Column Name.
- Save the step's configuration and close it with the **OK** button.


Properties of ONE Metadata Reader (catalogItem)

MMM Reader Root Stream Config

Name*: attributes

Attribute*: attributes

Columns:

	Name	Type	Src	Com
1	attribute_name	STRING	attribute.name	
2	data_type	STRING	attribute.dataType	
*				

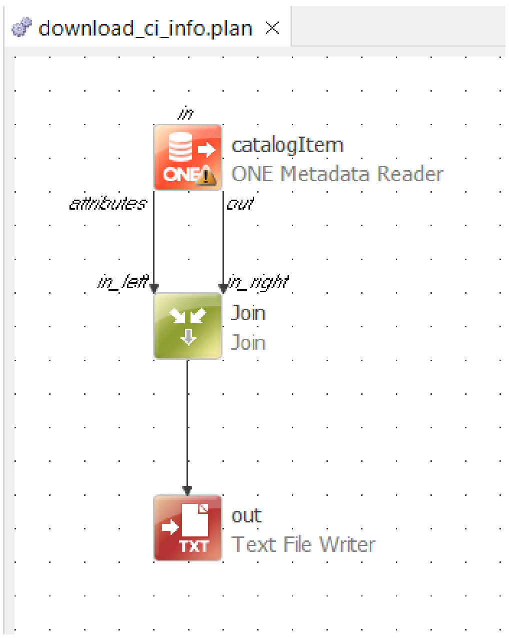
Id Column Name: attribute_id

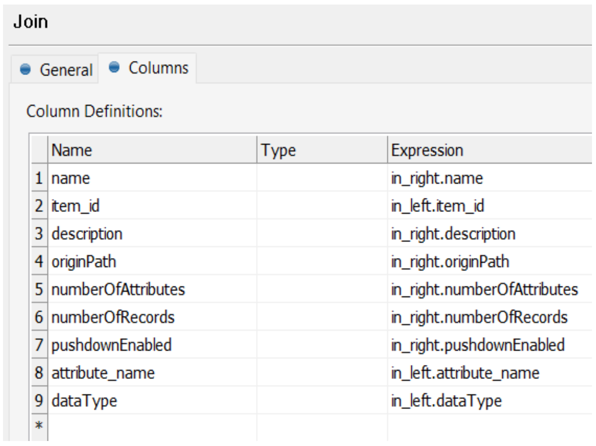
Entity Column Name:

Parent Id Column Name: item_id

Now, we have two output streams from the ONE Metadata Reader step: ***out*** and ***attributes***. The *out* stream contains simple properties of all catalog items, while *attributes* holds a list of all attributes and their data types. To create a **single** CSV file with catalog items and their attributes, we'll **join** the outputs using the ***item_id*** columns as **keys**.

- Insert a join step and configure it to get the desired metadata in a proper order.
- To complete the plan, don't forget to insert and configure a text file writer step.



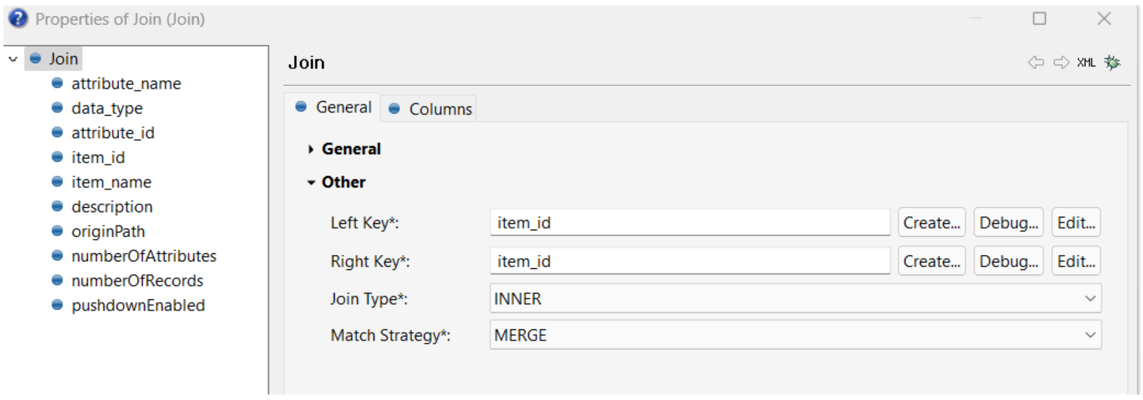


Join

General | Columns

Column Definitions:

	Name	Type	Expression
1	name		in_right.name
2	item_id		in_left.item_id
3	description		in_right.description
4	originPath		in_right.originPath
5	numberOfAttributes		in_right.numberOfAttributes
6	numberOfRecords		in_right.numberOfRecords
7	pushdownEnabled		in_right.pushdownEnabled
8	attribute_name		in_left.attribute_name
9	dataType		in_left.dataType
*			



Properties of Join (Join)

- Join
  - attribute_name
  - data_type
  - attribute_id
  - item_id
  - item_name
  - description
  - originPath
  - numberOfAttributes
  - numberOfRecords
  - pushdownEnabled

Join

General | Columns

General

Other

Left Key*:	item_id
Right Key*:	item_id
Join Type*:	INNER
Match Strategy*:	MERGE

TASK

*Time for a new challenge! Try to add another Embedded Entity Stream for **term instances** that are assigned to catalog items. How would you modify the ONE Metadata Reader step to include this additional output?*

In the end, try to keep generating a unified file containing all the Catalog Items, their attributes and their associated term instances.

4. Creating a term from ONE Desktop

In this task, we will create a new term **Product** using the ONE Desktop environment. Creating terms using ONE Desktop can be useful when we already have a list of terms available in a text file or a database and are looking for a more efficient way to create all of them at once in the ONE Web app. For this purpose, we will need another IDE step - **ONE Metadata Writer**.

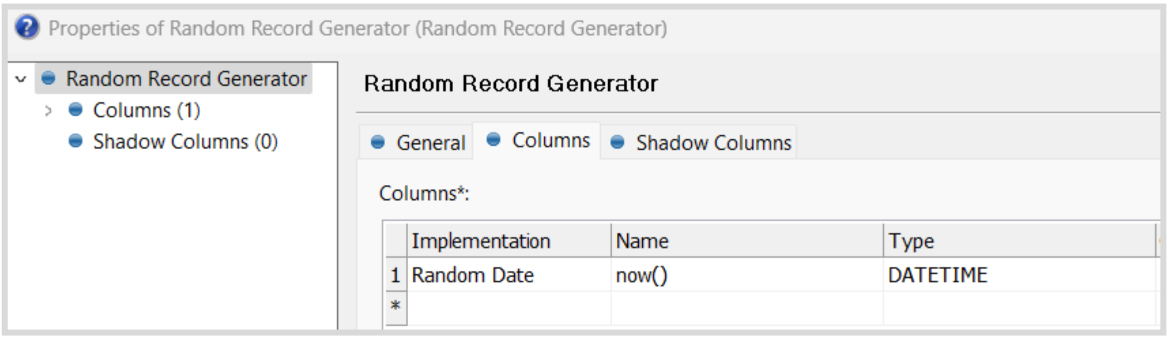
› Create a new plan and name it **create_term.plan**.

To be able to use the **ONE Metadata Writer** step, we need a definition of its '***in***' input. As we currently do not have any specific text file or database of terms, a **Random Record Generator** step can be used to create a single record for the input:

› Add a **Random Record Generator** step.

› Open the **Random Record Generator** step and configure it to generate only a single record:

- On the ***General*** tab, set the **Record Count** value to '1'.
- On the ***Columns*** tab, select an **Implementation** of your choice (e.g., *Pattern, Expression, Random Date* etc.).


Properties of Random Record Generator (Random Record Generator)

Random Record Generator

Columns*:

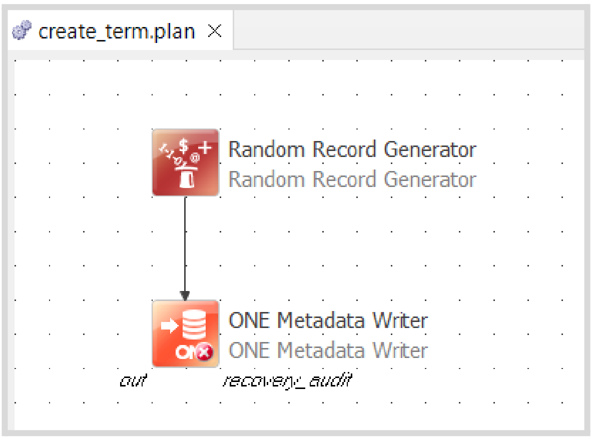
	Implementation	Name	Type
1	Random Date	now()	DATETIME
*			

› Now insert the **ONE Metadata Writer** step.

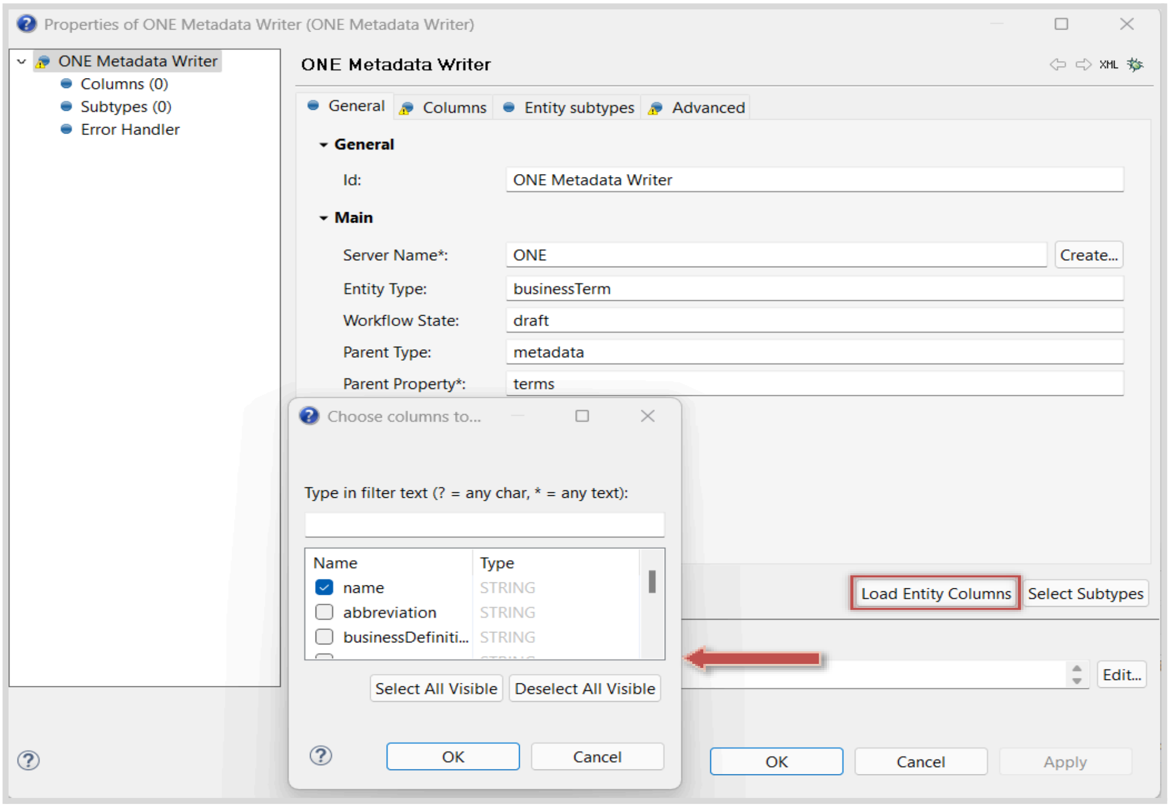
› Double-click to open its configuration and fill in the properties as follows:

• **Server name**:	*ONE*
• **Entity Type:**	*businessTerm*
• **Workflow State**:	*draft*
• **Parent Type:**	*metadata*
• **Parent Property:**	*terms*

NOTE *For all these steps you can use the **content assist** to fill in the information. They will become available after you specify the **Server name**.*



- To define properties of a Business Term, use the button **Load Entity Columns** which will offer you a complete list of business term properties.


Properties of ONE Metadata Writer (ONE Metadata Writer)

ONE Metadata Writer
- Columns (0)
- Subtypes (0)
- Error Handler

ONE Metadata Writer

General | Columns | Entity subtypes | Advanced

General

Id:	ONE Metadata Writer

Main

Server Name*:	ONE
Entity Type:	businessTerm
Workflow State:	draft
Parent Type:	metadata
Parent Property*:	terms

Choose columns to...

Type in filter text (? = any char, * = any text):

Name	Type
✓ name	STRING
abbreviation	STRING
businessDefiniti...	STRING

- Choose which properties to populate – at least a ***name***, which is mandatory – and provide it with a value. Fill in the **'<prefix>_Product'** string as the name of your new term.
- **Save** changes and close the step by pressing the **OK** button.

To function properly, the **ONE Metadata Writer** step requires an output step to be connected to the ***Recovery Audit*** endpoint. As the name suggests, this step can alternatively be used to check the logs of this step for auditing and troubleshooting purposes.

- Add a **Text File Writer** step to the ***Recovery Audit*** endpoint.
- **Save** the step's configuration and close it by pressing the **OK** button.

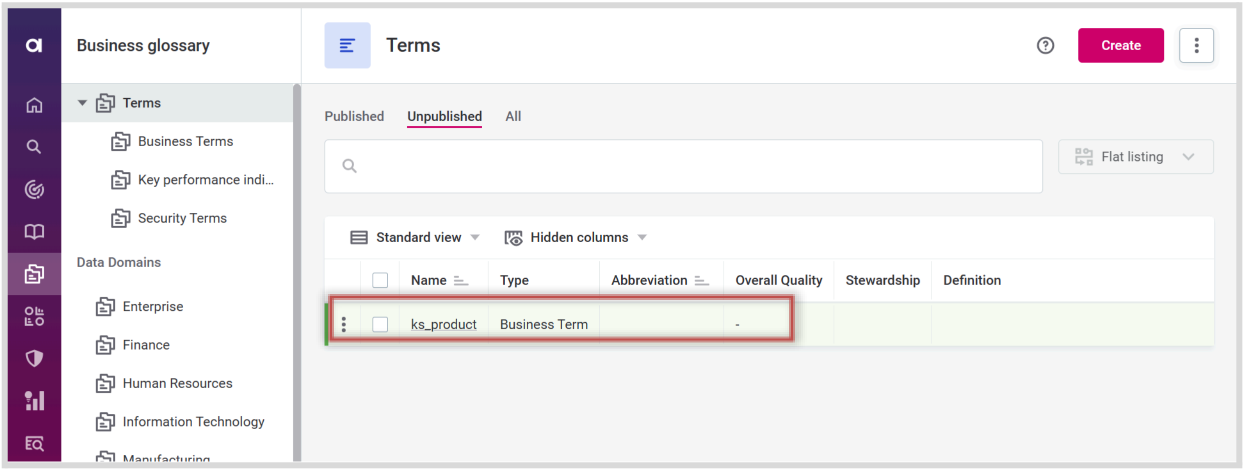
CAUTION

*The **ONE Metadata Writer** step will show the ⚠ symbol due to several warnings. These warnings refer to the missing 'out' output node and optional values that might be populated for future work with the Entity properties. Thus, you can ignore the warnings for now.*

- **Run** the plan.

Even though there are warnings on the steps, the plan runs successfully and the new term will be created. Let's see the result and confirm in the ONE Web application:

- Navigate back to the ONE Web application and check the result in the **Terms** section:



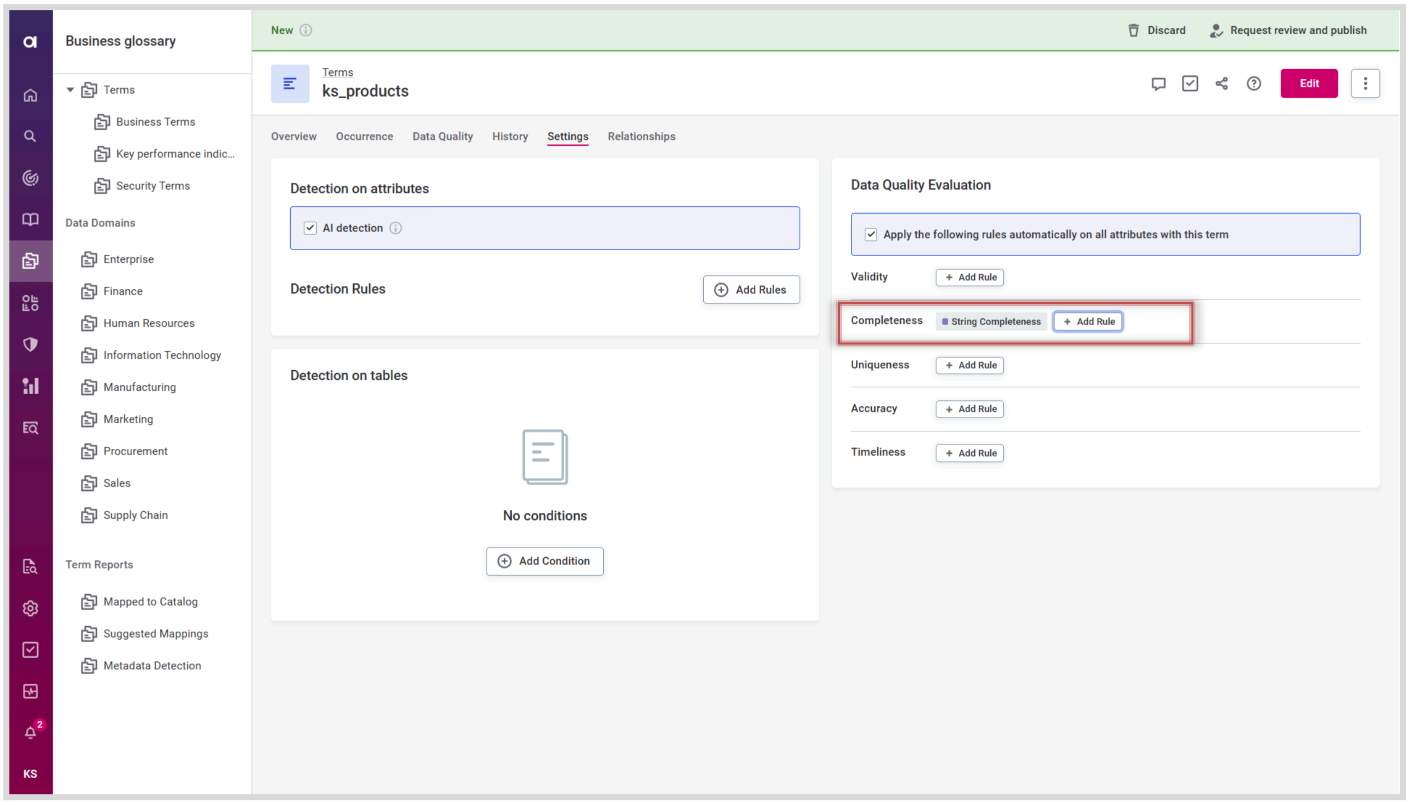
Name	Type	Abbreviation	Overall Quality	Stewardship	Definition
ks_product	Business Term		-		

Your new term should appear in the list of Terms. Let's see if it functions properly.

- Click the new **<prefix>_Product** term to open its details.
- Go to the **Settings** tab and check if you can add any detection rule to it.
- Check to see if you can add any data quality evaluation rule.

CAUTION

*Before adding any data quality rule, make sure "**Apply all these rules automatically for all the attributes with term**" is checked.*



- Click on the rules that you've added to see if you can be directed to the rule's page.
- After making sure the created term is functioning properly, **Publish** the changes.

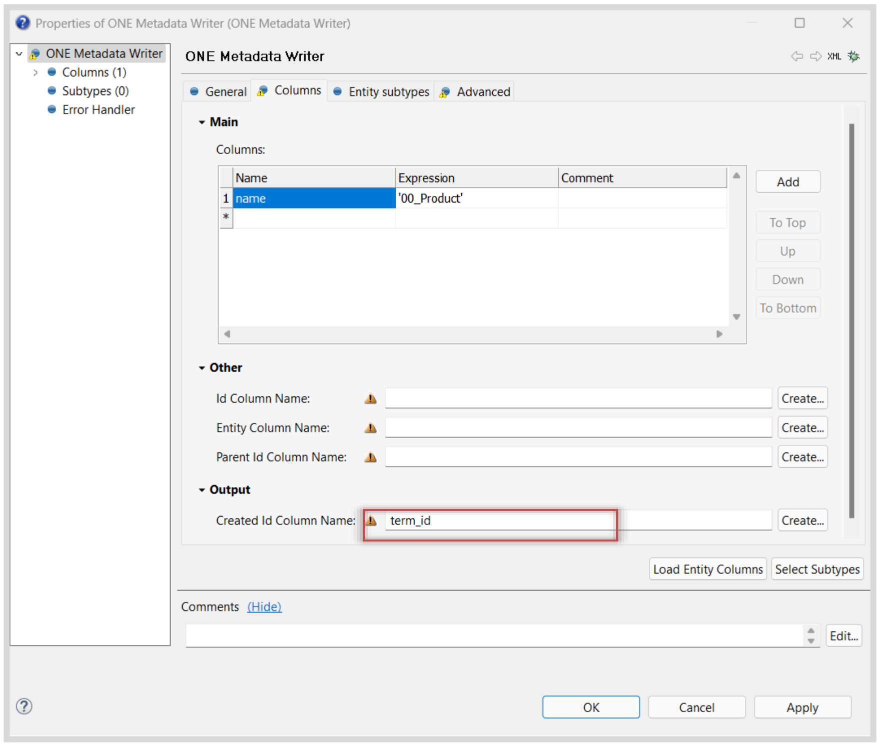
5. Creating term from ONE Desktop_complementry (optional)

Imagine if the term we created remotely wasn't functioning as intended, specifically in the aspect of being unable to add a DQ (Data Quality) evaluation rule. Consequently, we would find it necessary to adjust our plan, allowing us to associate Validation rules with the new term. This scenario could serve as a valuable lesson, as it's highly likely that when generating any form of asset remotely, certain properties might not perform as expected. Thus, embracing this concept could be beneficial!

The first thing to do is to capture the id of the term itself:

› Open the configuration of the **ONE Metadata Writer** step.

› Switch to the ***Columns*** tab and fill in the **Created Id Column Name** with a value – **term_id**:



Properties of ONE Metadata Writer (ONE Metadata Writer)

- ONE Metadata Writer
  - Columns (1)
  - Subtypes (0)
  - Error Handler

ONE Metadata Writer

General | Columns | Entity subtypes | Advanced

Main

Columns:

	Name	Expression	Comment
1	name	'00_Product'	
*			

Other

Id Column Name:	
Entity Column Name:	
Parent Id Column Name:	

Output

Created Id Column Name:	term_id

Comments (Hide)

This way we can remember the Id of the newly created term.

Now, we need to add another **ONE Metadata Writer** step to enable the Validation rules. This has to be done in a separate step after the new term has been created.

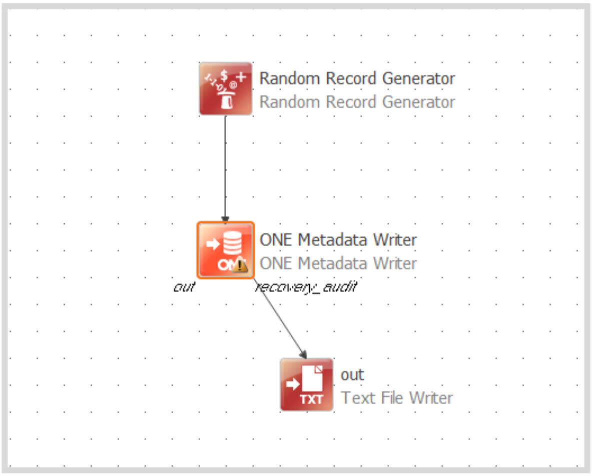
› Get back to your plan and add a second **ONE Metadata Writer** step. Connect the ***out*** endpoint of the existing one to the input of the new one.

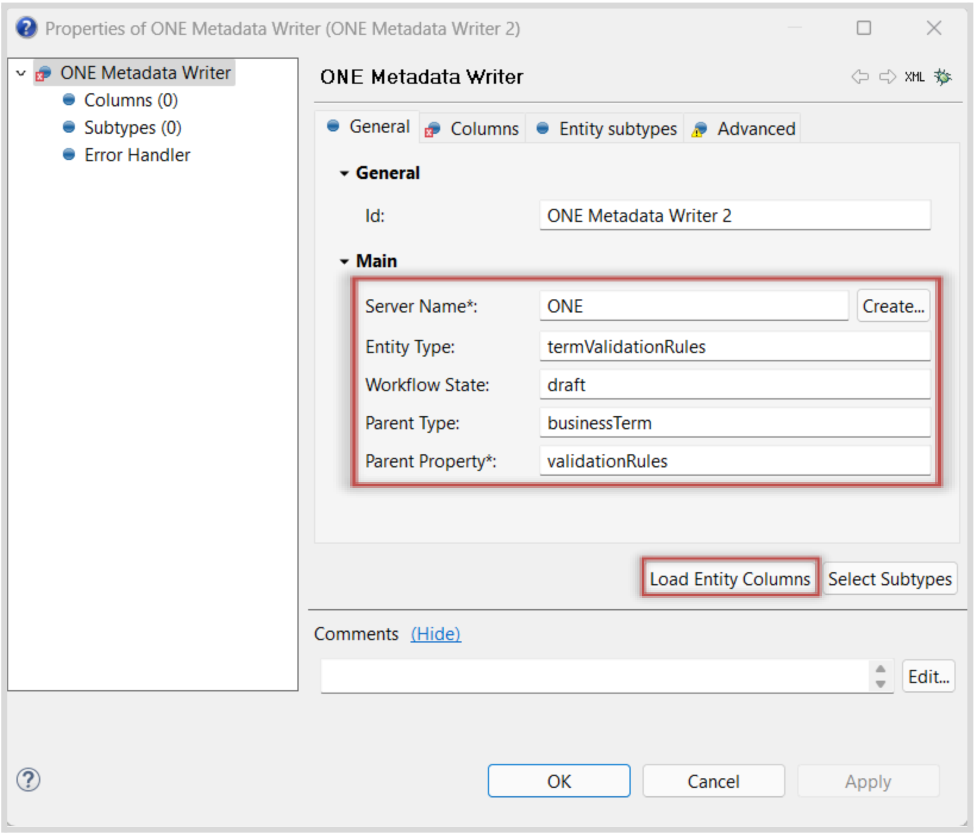
› Open the configuration of the second one and fill in the following values:

• **Server name**:	*ONE*
• **Entity Type:**	*termValidationRules*
• **Workflow State**:	*draft*
• **Parent Type:**	*businessTerm*
• **Parent Property:**	*validationRules*

HINT

*Similar to previous ONE integration steps, you can use **content assist** to fill in the information. This will become available after you specify a **Server name**.*




Properties of ONE Metadata Writer (ONE Metadata Writer 2)

- ONE Metadata Writer
  - Columns (0)
  - Subtypes (0)
  - Error Handler

ONE Metadata Writer

General | Columns | Entity subtypes | Advanced

General

Id:	ONE Metadata Writer 2

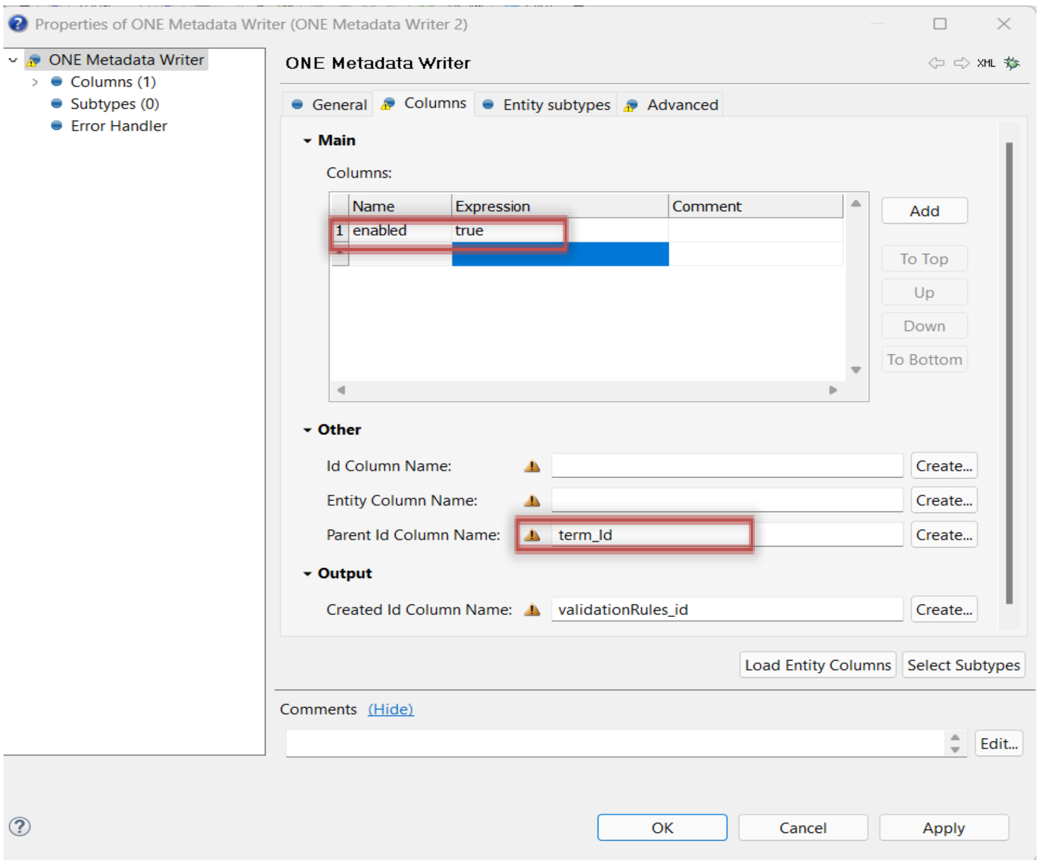
Main

Server Name*:	ONE
Entity Type:	termValidationRules
Workflow State:	draft
Parent Type:	businessTerm
Parent Property*:	validationRules

After filling in the general information:

- Switch to the ***Columns*** tab of the configuration.
- Use the **Load Entity Columns** button and add the **enabled** column. Set it as '**true**'.

- Fill in the **term_Id** attribute as a **Parent Id Column Name**.
- If you want, you can also specify validationRules_id as Created Id Column Name.



Properties of ONE Metadata Writer (ONE Metadata Writer 2)

ONE Metadata Writer

Main — Columns:

Name	Expression	Comment
enabled	true	

Other

Id Column Name:

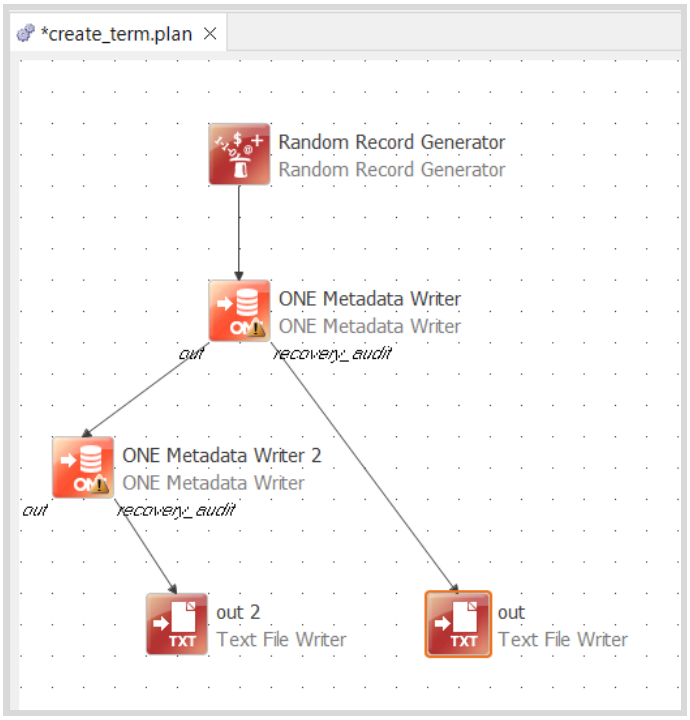
Entity Column Name:

Parent Id Column Name: term_Id

Output

Created Id Column Name: validationRules_id

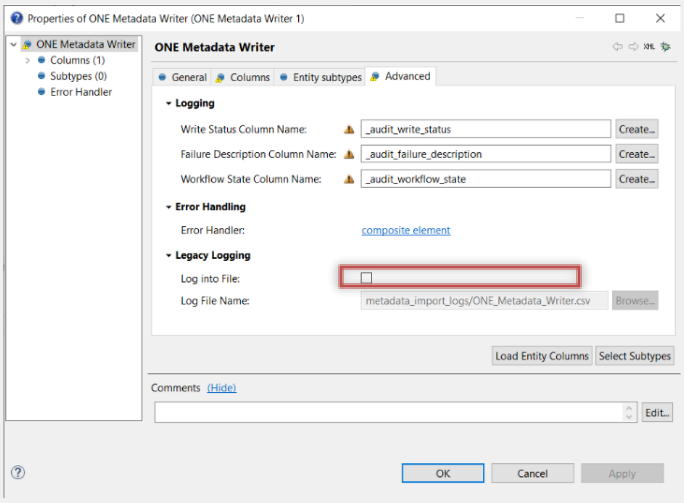
When all is finished properly, the resulting plan should look like this:



HINT

*If you created the second **ONE Metadata Writer** step by copy-paste action of the first one, it is probably showing an error now. This is because it has a pre-generated filename for its **Legacy Logging** which reports a filename clash even if disabled.*

*Double-check this on the step's **Advanced** tab and update the **Log File Name** if necessary:*



Properties of ONE Metadata Writer (ONE Metadata Writer 1)

ONE Metadata Writer

General | Columns | Entity subtypes | Advanced

Logging

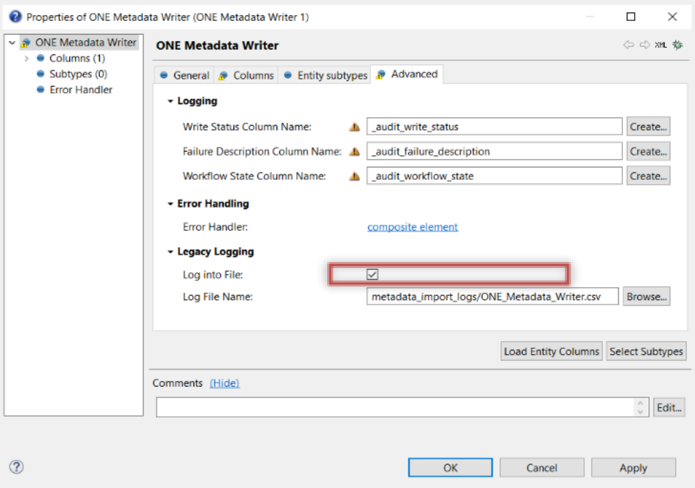
Write Status Column Name:	_audit_write_status
Failure Description Column Name:	_audit_failure_description
Workflow State Column Name:	_audit_workflow_state

Error Handling

Error Handler: composite element

Legacy Logging

Log into File:	☐
Log File Name:	metadata_import_logs/ONE_Metadata_Writer.csv



Properties of ONE Metadata Writer (ONE Metadata Writer 1)

ONE Metadata Writer

General | Columns | Entity subtypes | Advanced

Logging

Write Status Column Name:	_audit_write_status
Failure Description Column Name:	_audit_failure_description
Workflow State Column Name:	_audit_workflow_state

Error Handling

Error Handler: composite element

Legacy Logging

Log into File:	✓
Log File Name:	metadata_import_logs/ONE_Metadata_Writer.csv

TASK

*Think about how you would add some other more complex properties to the new term. For instance - how would you create a **Parent-Child relationship** between this '**Product**' term and another existing term (e.g. the term **'Product department'** that was created in the Business Glossary workshop)?*

You can find some hints at the end of this document.

6. Creating a Virtual Catalog Item (VCI)

It's often the case that profiling the raw data directly from a given source isn't ideal. Typically, a series of transformations is required to prepare the data before analysis. Additionally, there may be instances where the data comes from sources or types not natively supported by the ONE Web Application, such as data accessed through API calls. To address these data preparation needs, the concept of **Virtual Catalog Items (VCIs)** is introduced.This allows you to:

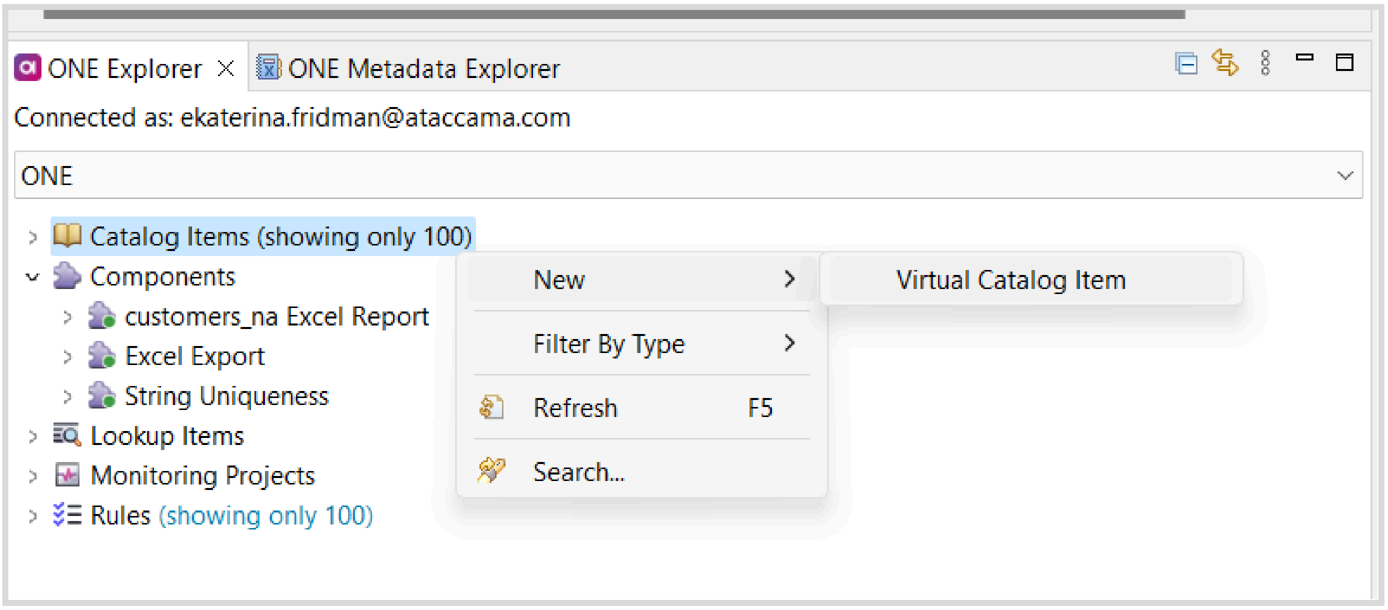
- Use an **existing** Catalog Item (and perform some transformations to it).
- Create a completely new Catalog Item (and provide its definition).

For our training, we will take already existing Catalog Item **orders** and create a new column **processing_time** to tell us how long it took to finish an order. We will use the columns **orderdate** and **shippeddate** to calculate the time difference.

Note that VCIs are always created using the **ONE Desktop** application. Therefore, you need to be in the ONE Desktop application and make sure you are connected to the ONE Web application.

› In ONE ***Explorer***, right-click on the **Catalog Items** section to open the context menu.

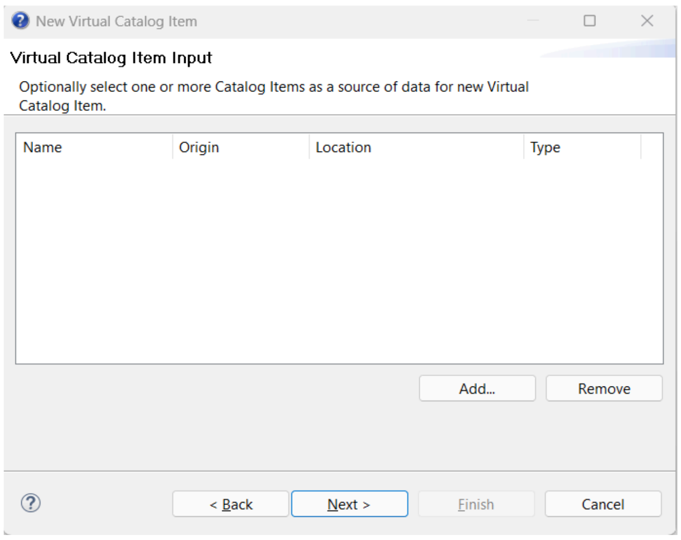
› Choose the '**New'** and '**Virtual Catalog Item'** options.



› Populate a **Catalog Item Name**, e.g., **'<prefix>_Orders_transformed'**, optionally provide some description, then press **Next**.

Since we have decided to provide data from an already existing Catalog Item, we will select the source of input in the next step:

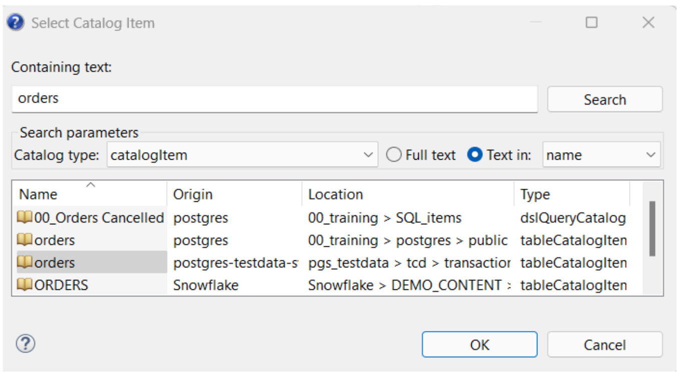
› Click the '**Add...'** button and search for the **orders** Catalog Item using the inbuilt Search engine. Make sure your filter is set to the '*catalogItem*' type of results:


New Virtual Catalog Item

Virtual Catalog Item Input

Optionally select one or more Catalog Items as a source of data for new Virtual Catalog Item.

Name	Origin	Location	Type


Select Catalog Item

Containing text: orders

Search parameters

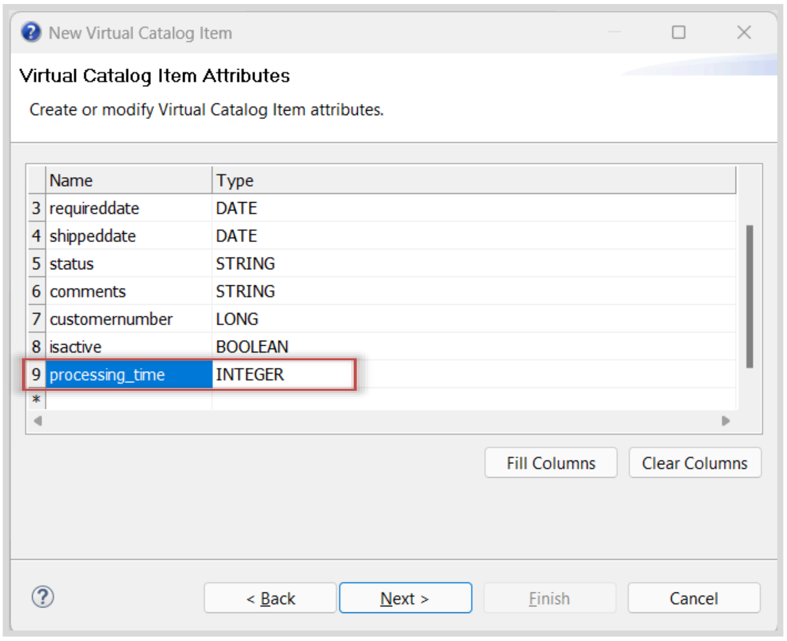
Catalog type: catalogItem [✓ Text in:] Full text name

Name	Origin	Location	Type
00_Orders Cancelled	postgres	00_training > SQL_items	dslQueryCatalog
orders	postgres	00_training > postgres > public	tableCatalogIten
orders	postgres-testdata-s	pgs_testdata > tcd > transactior	tableCatalogIten
ORDERS	Snowflake	Snowflake > DEMO_CONTENT :	tableCatalogIten

If you want to create a brand-new catalog item without using any of the existing ones, you need to skip this part.

In the next part, you can modify the list of VCI's attributes. There, you can leave out some of the columns or add new ones. In this example, we want to calculate the total number of days for processing duration and store this information in a new attribute:

- Press the **Next** button to advance to the list of VCI's attributes.
- Scroll to the end of the list and create a new column **processing_time** as the type of Integer.


New Virtual Catalog Item

Virtual Catalog Item Attributes

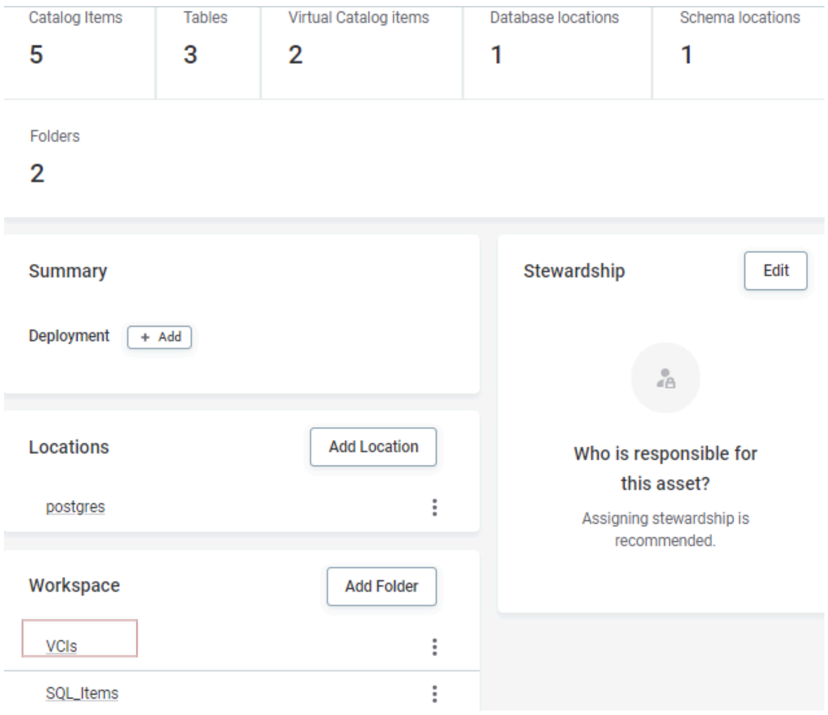
Create or modify Virtual Catalog Item attributes.

	Name	Type
3	requireddate	DATE
4	shippeddate	DATE
5	status	STRING
6	comments	STRING
7	customernumber	LONG
8	isactive	BOOLEAN
9	processing_time	INTEGER
*		

The last step is to choose a location to store the VCI; this should be a folder within a source. where a **Workspace** (folder) will be created. This is the folder/physical location for the new VCI.

- Under the **training** source, create a new folder and name it e.g., **VCIs**.
- Click **Finish** to complete the process of creating the new VCI.

If you now go to the web application and open the data source **<prefix>_training**, you will see the new **Workspace** folder in the overview tab.



Catalog Items 5 | Tables 3 | Virtual Catalog items 2 | Database locations 1 | Schema locations 1

Folders 2

Summary

Deployment + Add

Stewardship Edit

Who is responsible for this asset?

Assigning stewardship is recommended.

Locations Add Location

postgres

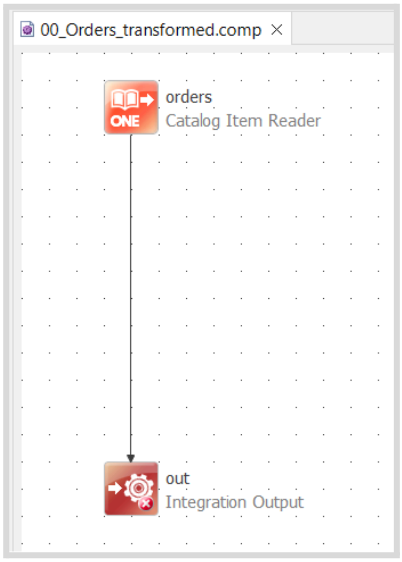
Workspace Add Folder

VCIs

SQL_Items

Although already included in the count of your source catalog items, the new VCI is now just a draft; you need to implement the logic. If you go back to the ONE Desktop, the transformation component for your new VCI should be open with two steps already preconfigured:

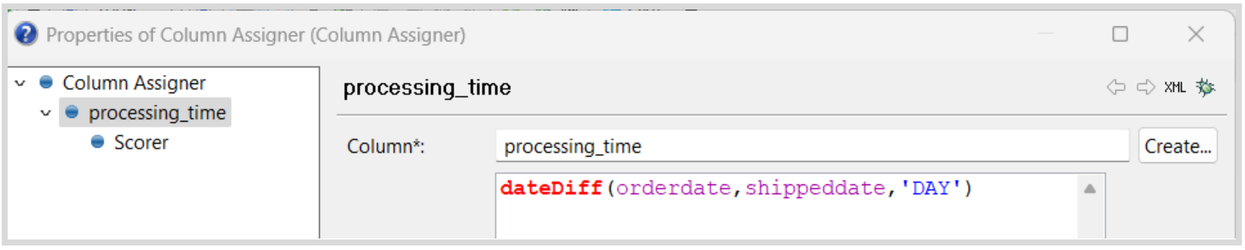
- **Catalog Item Reader: 'orders'** - It consumes data of your existing **orders** Catalog Item.
- **Integration Output** – It contains the definition of the columns of your new VCI.



Let's now create the transformation logic for the newly added column **processing_time**:

› Add an **Alter Format** step to create a new column **processing_time**. Position it after the **Catalog Item Reader** step.

- Add a **Column Assigner** to create an expression to return the number of days between the two DATE type attributes: **orderdate** and **shippeddate**.

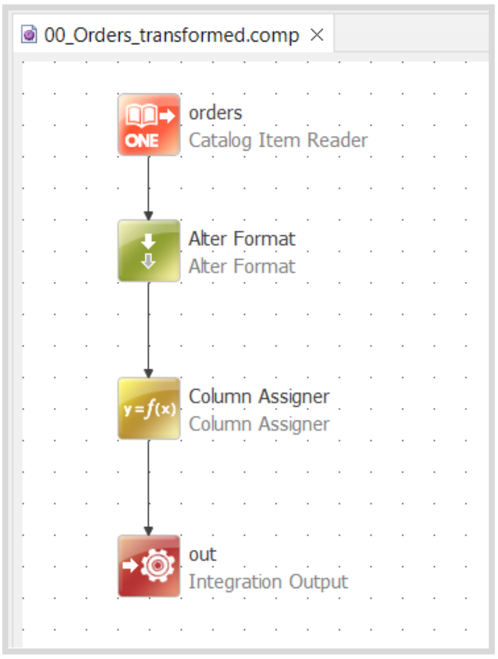


Column*: processing_time

```
dateDiff(orderdate,shippeddate,'DAY')
```

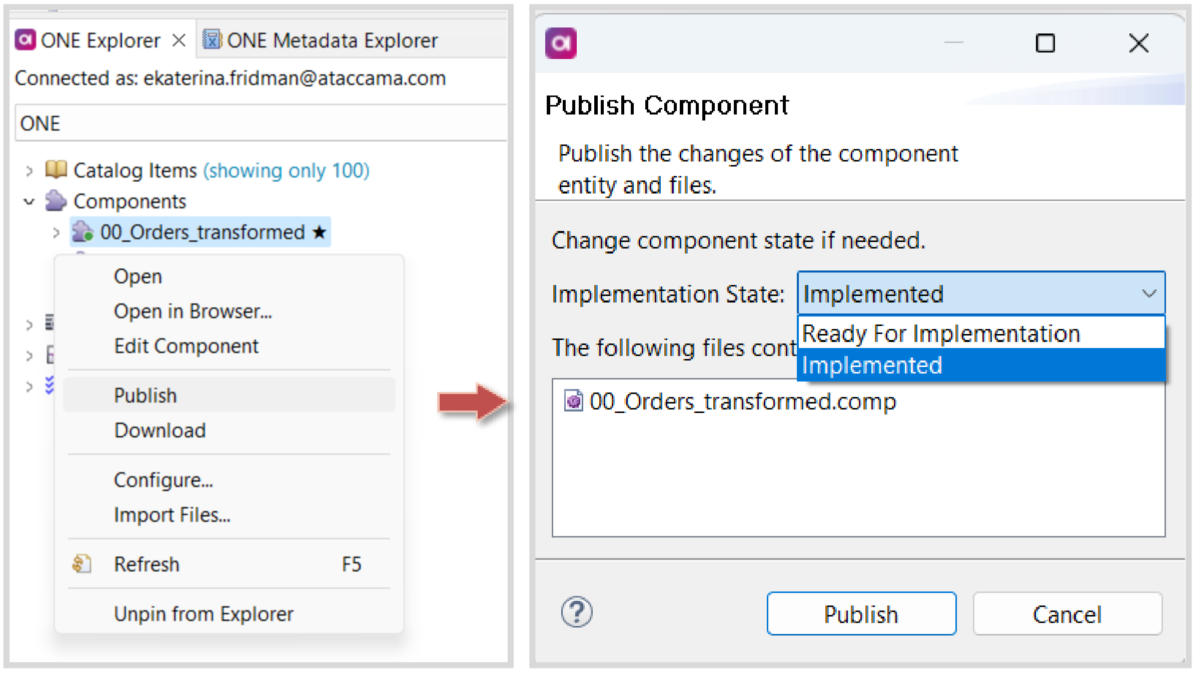
Expression: dateDiff(orderdate,shippeddate,'DAY')

When finished, your plan should look like this:



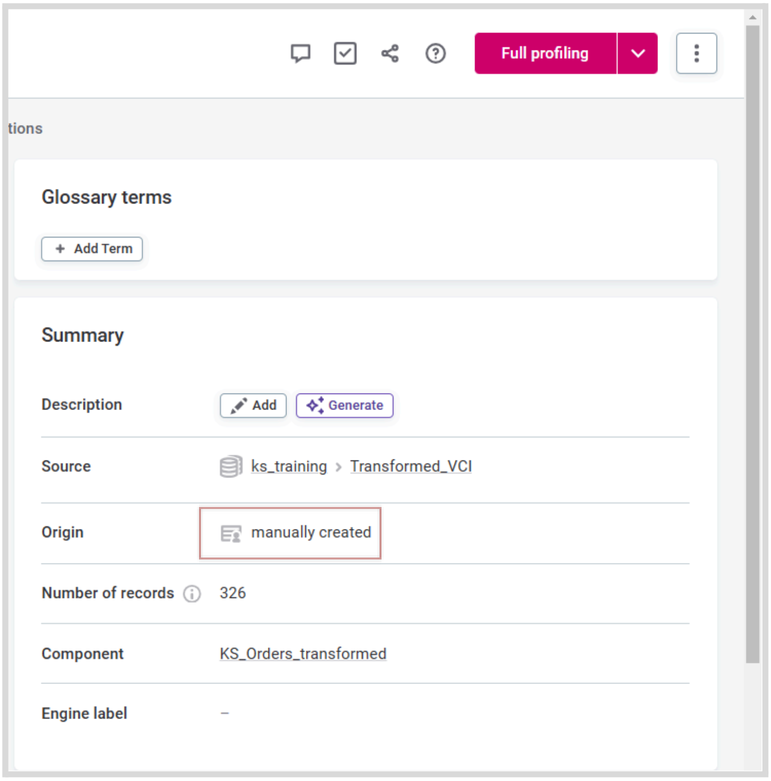
Now we need to publish the component to apply the changes:

- In the ***ONE Explorer*** tab, under the **Components** section, right-click on the new component and choose **Publish**.
- In the window that opens, choose the '**Implemented'** value as the **State** and complete the **Publish** action.



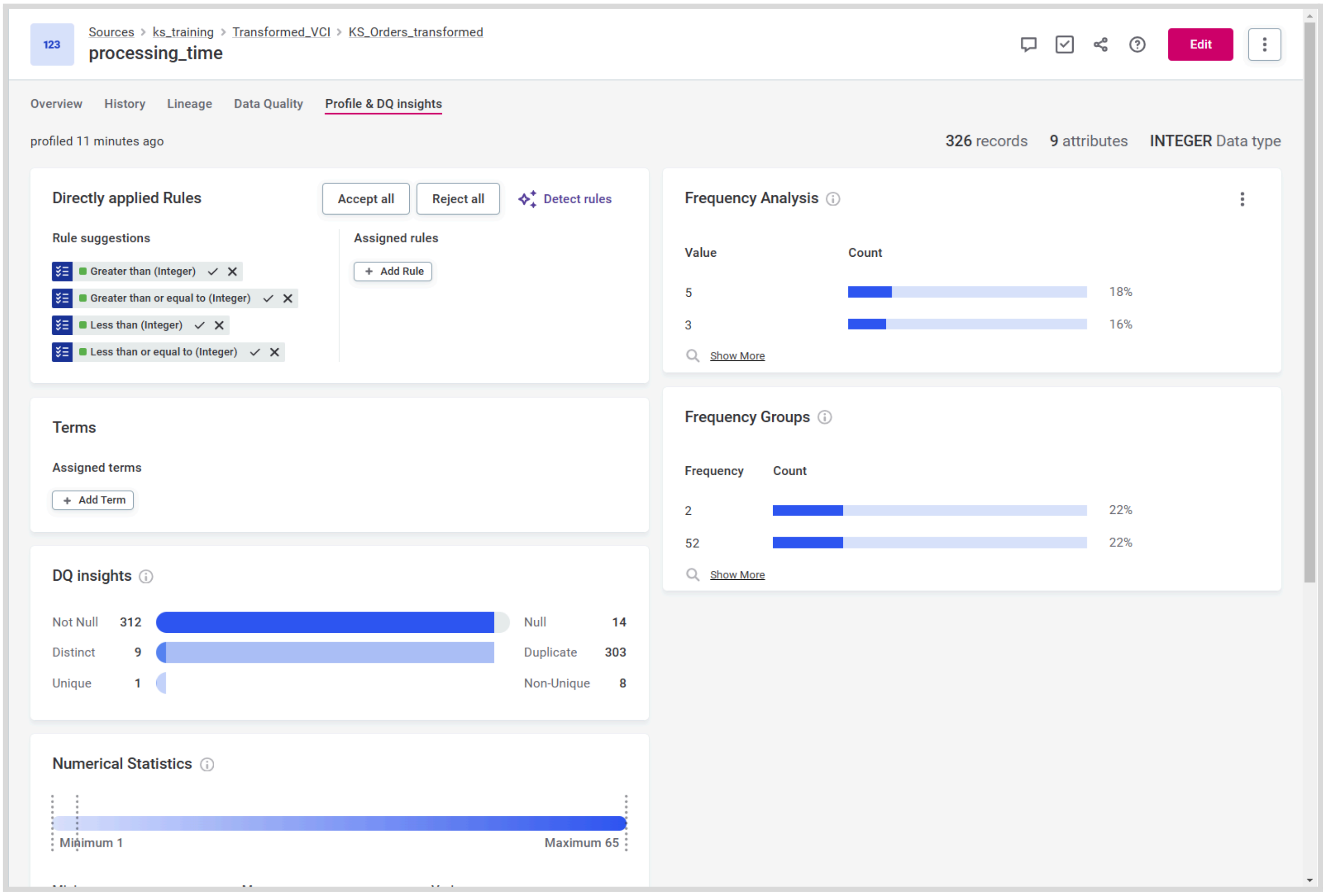
- Switch to the **ONE Web Application** and go to the **Catalog Items** section.
- Find your new VCI (**<prefix>_Orders_transformed**) and run the **Full profiling** option on it.

Notice the **Origin** path which says **manually created**. This indicates that the item is a VCI.



Description	
Source	ks_training > Transformed_VCI
Origin	manually created
Number of records	326
Component	KS_Orders_transformed
Engine label	–

› Once the profiling is finished, open the details of your new **processing_time** column and review the profiling results.



Sources > ks_training > Transformed_VCI > KS_Orders_transformed

processing_time

Overview | History | Lineage | Data Quality | Profile & DQ insights

profiled 11 minutes ago

326 records 9 attributes INTEGER Data type

Directly applied Rules

Rule suggestions:
- Greater than (Integer)
- Greater than or equal to (Integer)
- Less than (Integer)
- Less than or equal to (Integer)

Assigned rules

Frequency Analysis

Value	Count
5	18%
3	16%

Frequency Groups

Frequency	Count
2	22%
52	22%

Terms

Assigned terms

DQ insights

Not Null	312	Null	14
Distinct	9	Duplicate	303
Unique	1	Non-Unique	8

Numerical Statistics

Minimum 1 Maximum 65

*When looking at the profiling results of the **processing_time**:*
What can you tell about your orders?
How long does it usually take to finish the order?
Are there any strange values? How would you explain them?

Conclusion

We have come to the end of this workshop!

We learned how to connect our ONE Desktop application to the Ataccama ONE web application and how to use it for some integration tasks. We went through the special IDE steps and learned how to configure them. Lastly, we learned how to transform existing Catalog Items and create a brand-new one using Virtual Catalog Items.

In the next workshop, we will explore validation rules and how to use ONE Desktop to create more complex rules.

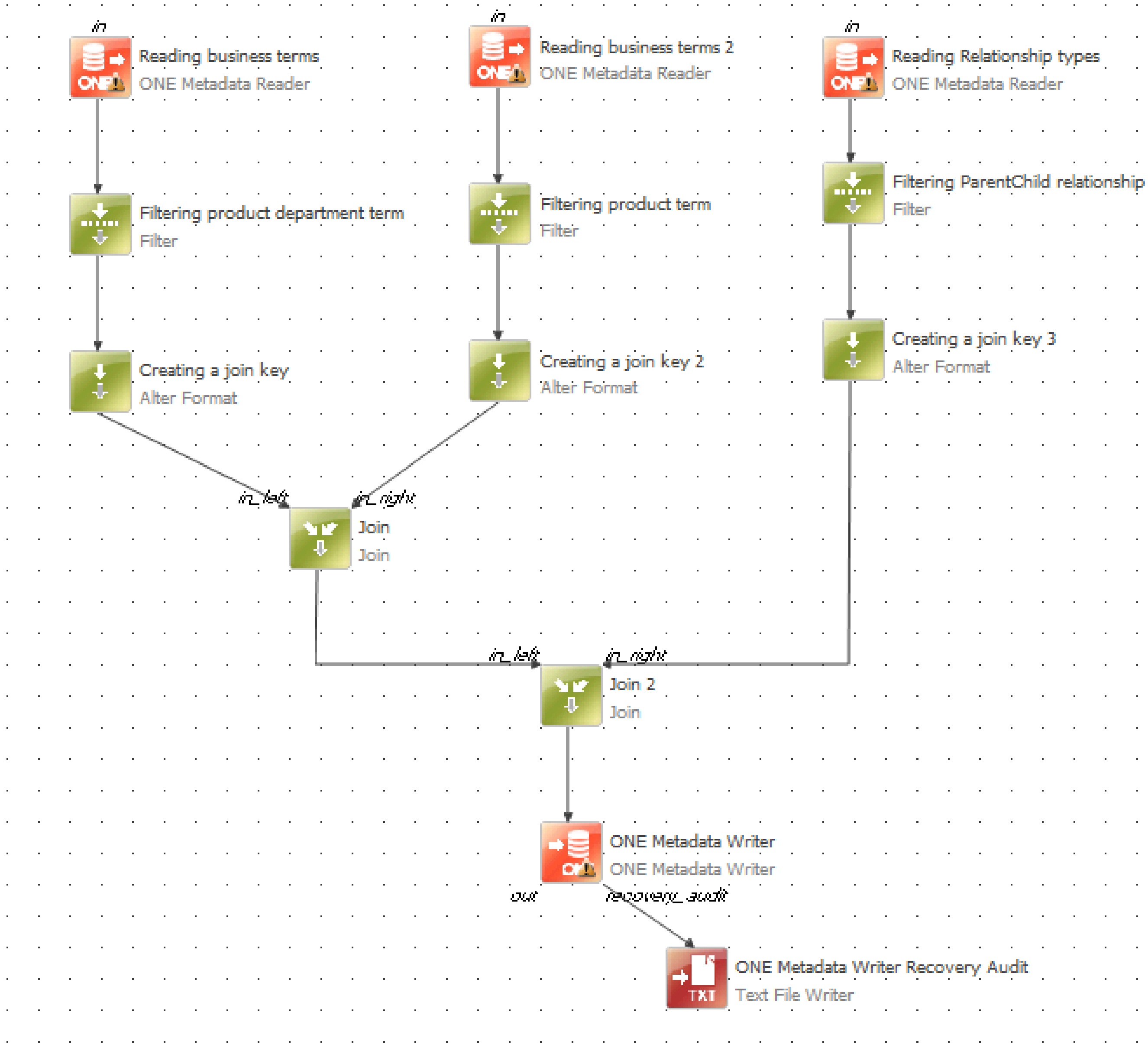
Hints & Solutions

Tak 4 optional assignment hints:

In order to create a relationship between two terms, you need a Metadata Writer step. However, in order to do that, some necessary information relevant to them such as their unique **ID**s as well as the ID of the ParentChild relationship is required. In this regard, you need to insert **Metadata Reader** steps to download all **relationships** and **business terms** separately and then filter the results by your desired type (**parentChild**) and term names(product, product department), respectively. Once you have all this information, you can create the relationship using a **Metadata Writer** step:

- As an Entity **Type** you would choose **termRelationship**.
- Subsequently, required columns are the **source** (=parent ID), **target** (=child ID) and **type** (ID of the Parentchild relationship).

The final plan can look for example like this one:



You can check the result in the web application. When you open the term (either Product or Product Department), under Relationships you should see the newly created relationship.