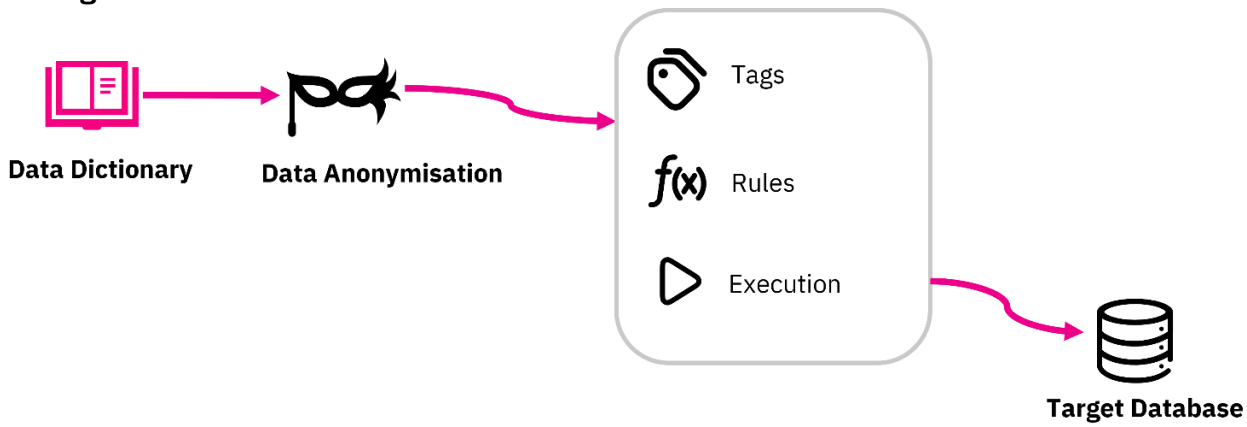
Masking databases with Curiosity Software

Introduction

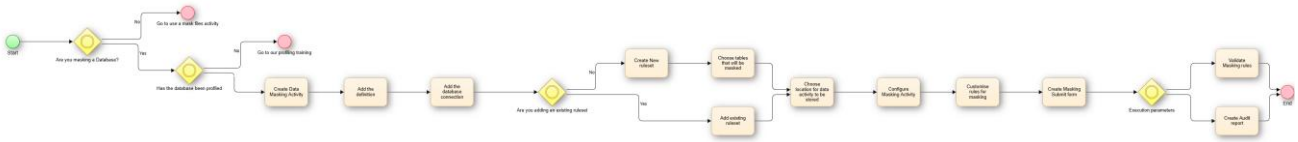
Data masking involves replacing sensitive information in databases with fabricated, yet realistic data, thereby protecting the actual data while still enabling the use of the database system for various tasks such as software testing, development, user training, or data analysis. This ensures sensitive data, including Personally Identifiable Information (PII) and commercially sensitive data, remains inaccessible to unauthorised users, minimising the risk of data breaches especially during non-production activities where full security measures may not be implemented.

Additionally, database masking is crucial for regulatory compliance, as many industries are subject to regulations like HIPAA, PCI DSS, and GDPR that mandate the protection of sensitive customer data. Non-compliance can result in substantial fines and reputational damage, making database masking essential for businesses to meet these requirements and protect stakeholders.

Training overview:



This training course will take you through the journey of creating a data masking activity using a data dictionary and connection which have already been scanned with PII identified. This is shown visually below as a process diagram:



By the end of this self-led training you will be able to:

- Mask a database
- Mask multiple databases keeping referential integrity
- Expose masking routines to be run how you would like

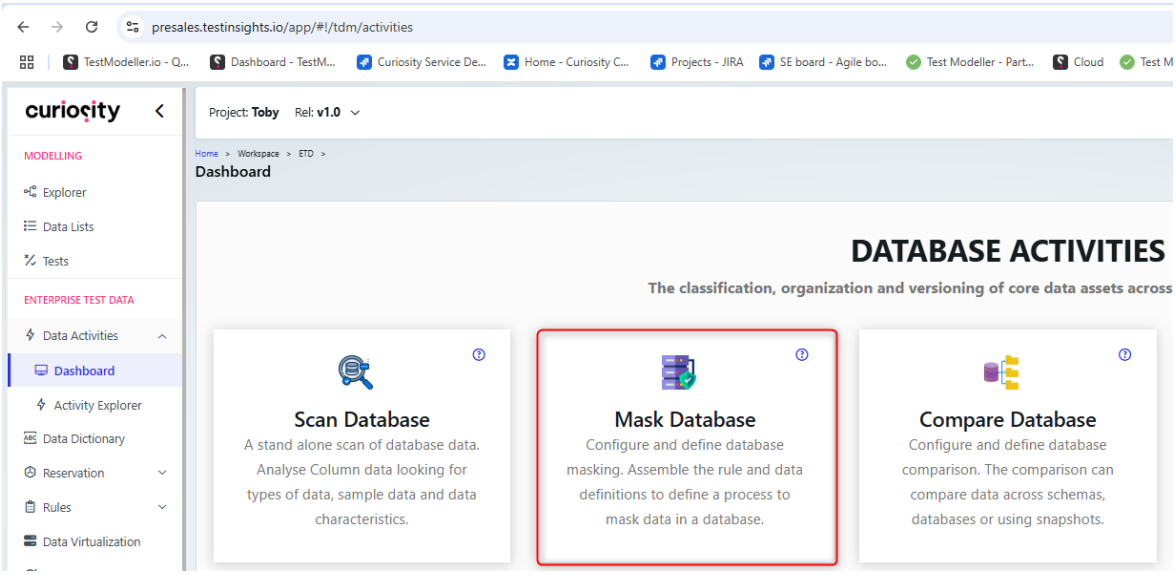
Prerequisites for data masking

For this data activity, you will need:

- A Database Connection
- A Data Definition
- To identify your PII. (Please see the Profiling training in this series for guidance)
- To cross-reference your schema set up

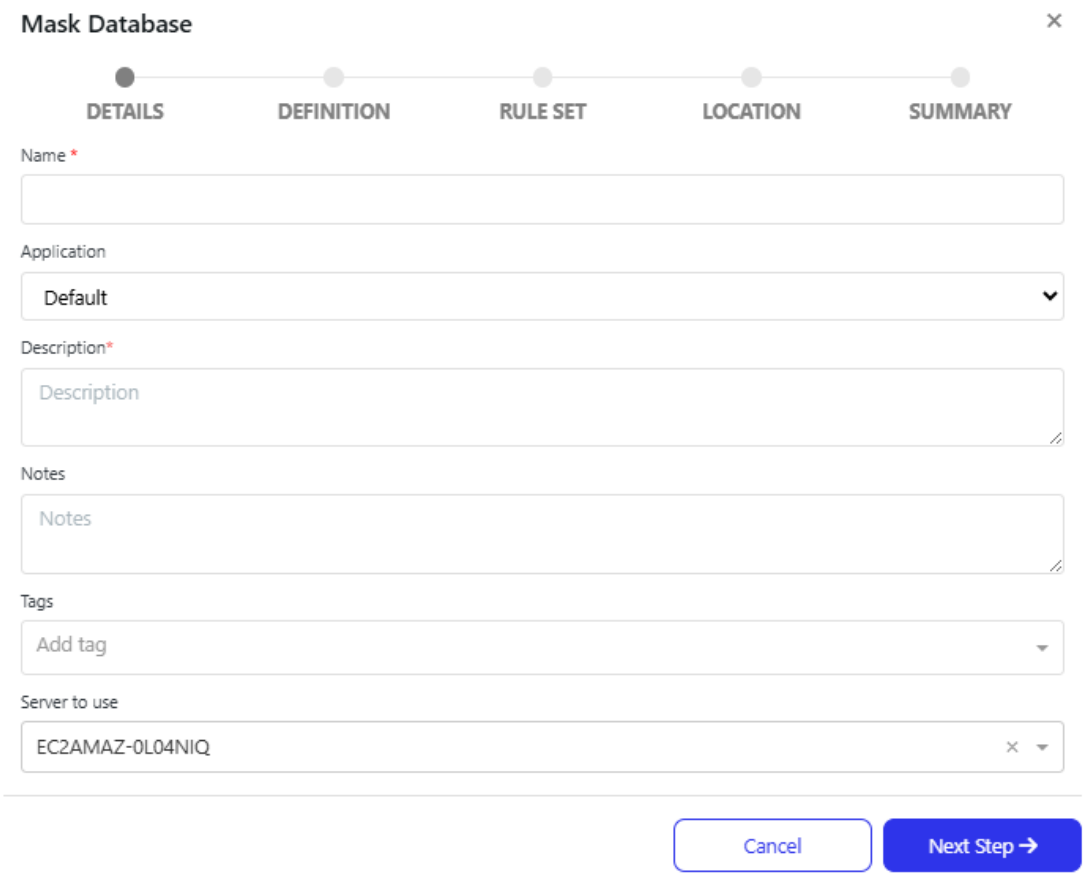
Step 1 - Create data masking activity

The first step to masking your data is to create a new data masking activity. Firstly, navigate to the Data Activities dashboard, then select the 'Mask Database' activity.

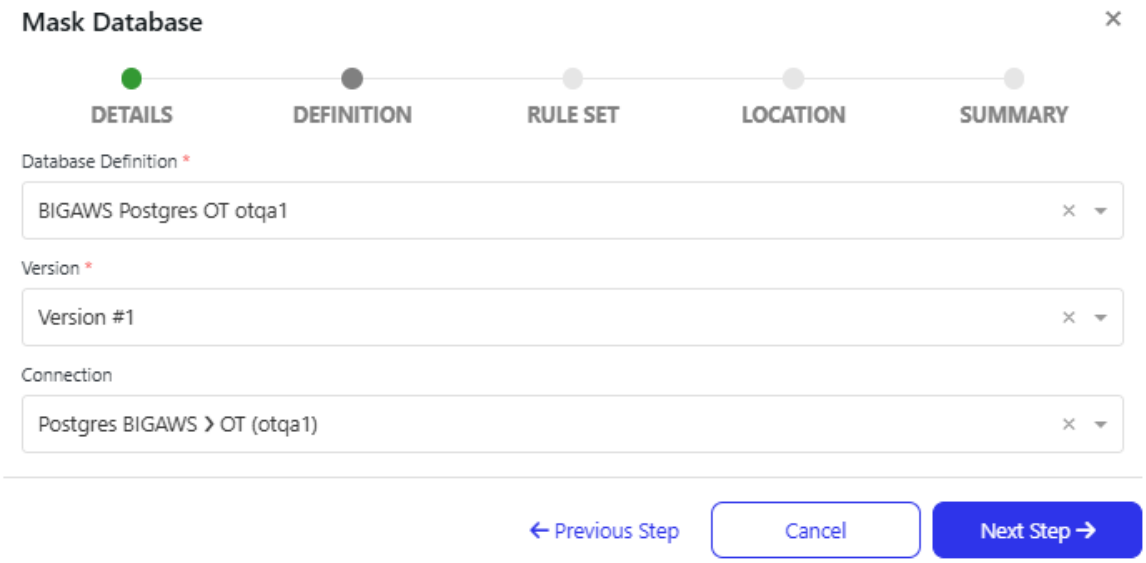


This will launch the wizard for creating the mask database activity. The wizard will guide you through the following steps, click the 'next step' button as you go to progress.

1. **Details** – Here the user can decide the name of the activity and write a description of the activity

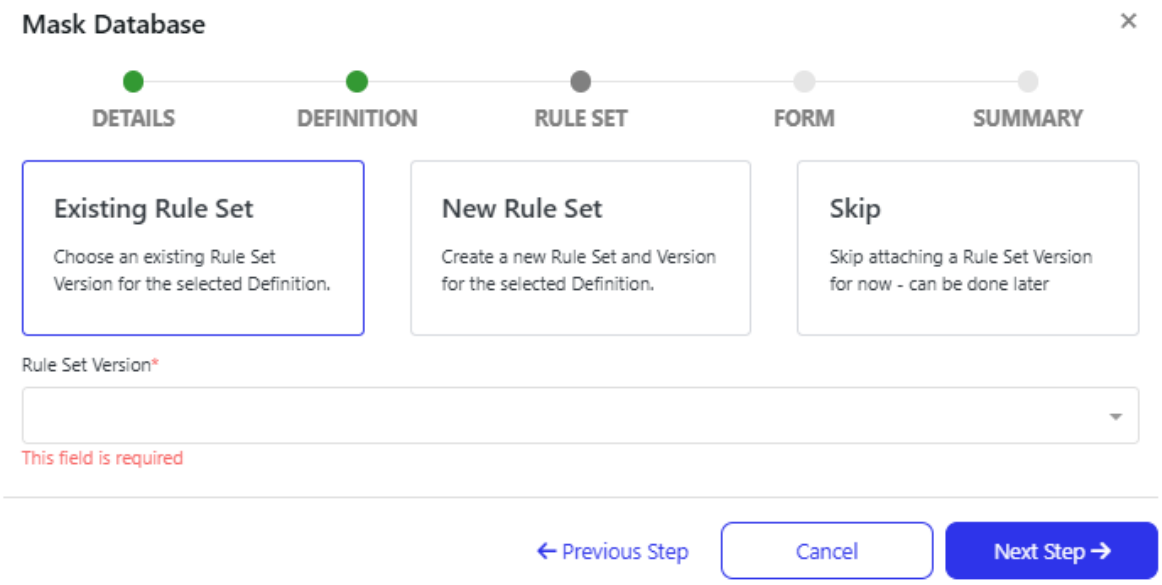


2. **Definition** – here the user can choose the definition and the desired version, it will automatically fill out the related connection for the definition.

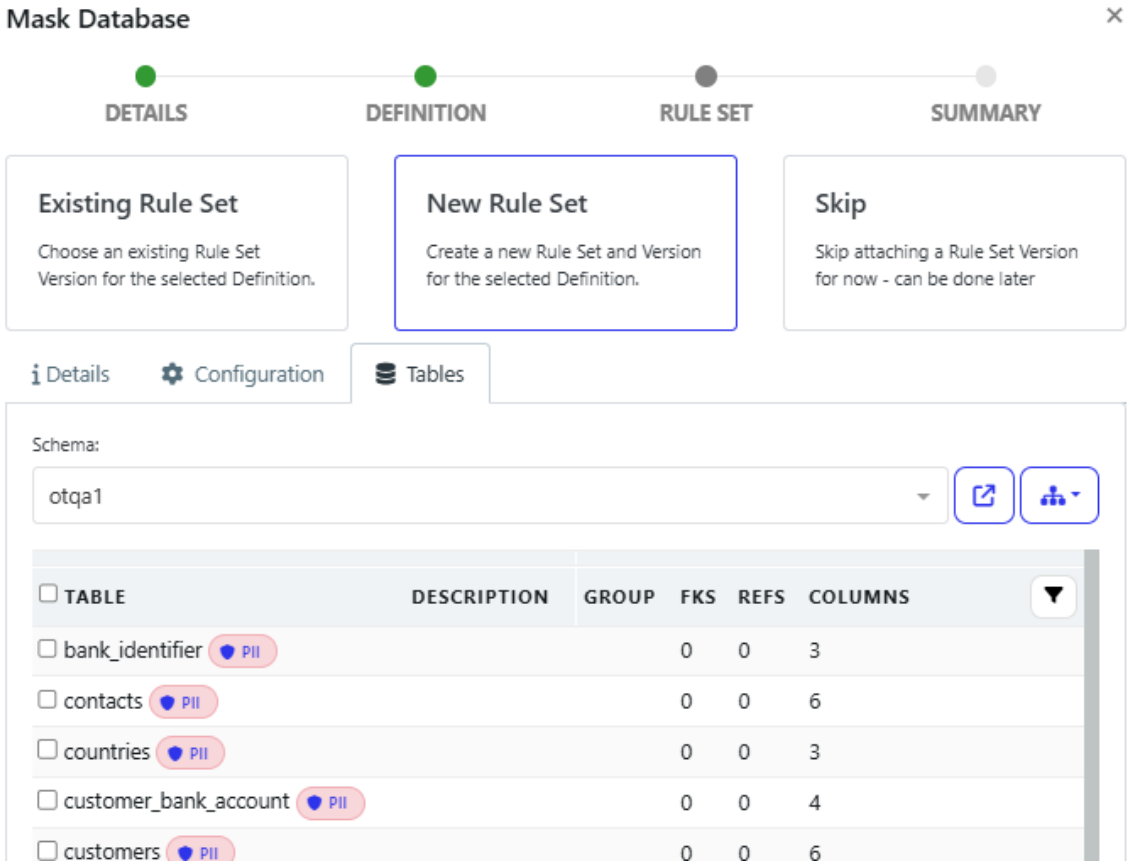


3. **Rule set** – here the user can choose whether to create a new rule set for the activity or choose an already created rule set to use

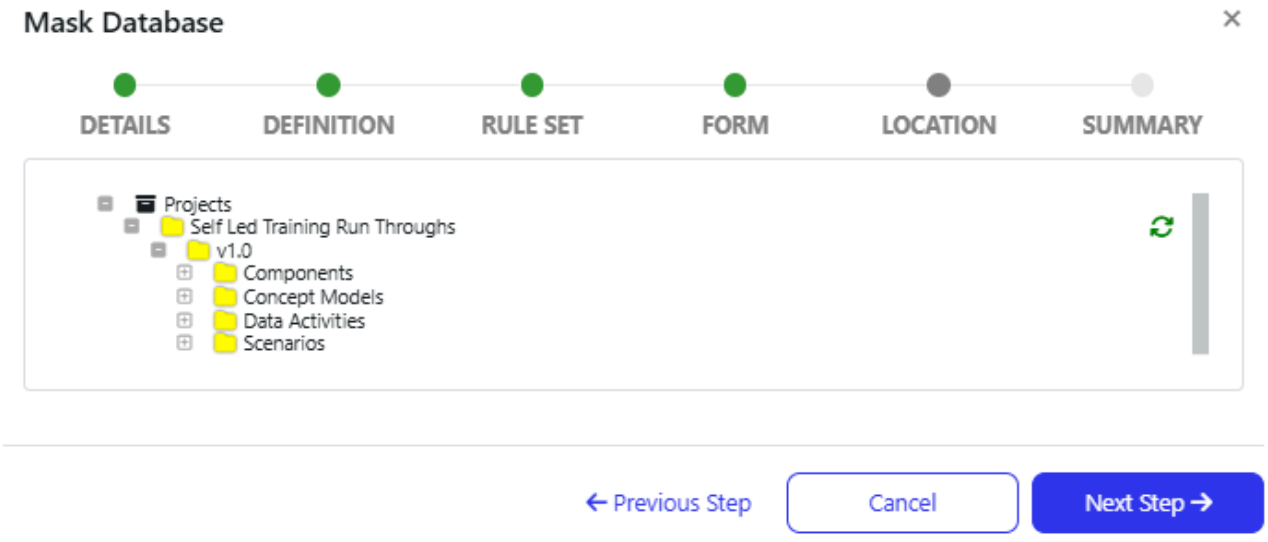
Existing rule set – choose an existing rule set that is associated with this definition



New rule set – Configure which tables are to be masked by this activity. The masking rules will be configured later



4. **Location** – choose where to save the data activity



A new data activity will appear in the chosen location, ready for configuration.

Step 2 - Masking configuration

When a Masking data activity is created, a default configuration will be set. These specify different settings within the data activity. You will want to edit these default configurations to suit your needs. To do this, click the edit button in the top right-hand corner.

The following fields are available to edit:

- **Masking list application ID:** If you're using a seed list, you will need to choose the application that contains those seed lists.
- **Locale:** For some functions the locale will change the result, e.g. addresses. By default, it will be in English.
- **Batch size:** Enter the size of a batch of rows to use in updating the database
- **Xref connection ID:** If you would like to cross reference, choose the database connection you'll be cross referencing against.
- **Include views:** A view is the result set of a stored query, which can be queried in the same manner as a persistent database collection object. Toggle this to on if you would like to include views, otherwise it can be toggled off.

Exercise 1 – Create and configure a data masking activity

a) Use the instructions in Step 1 to create a data masking activity for your OT database (or other training database)

b) Use the instructions in Step 2 to configure your mask database activity

Step 3 – Customise the rule set

A rule set is utilised to determine the appropriate masking rules to be applied to each column in every table of the database. Masking rules define the functions that will substitute values and, in the end, obscure the data.

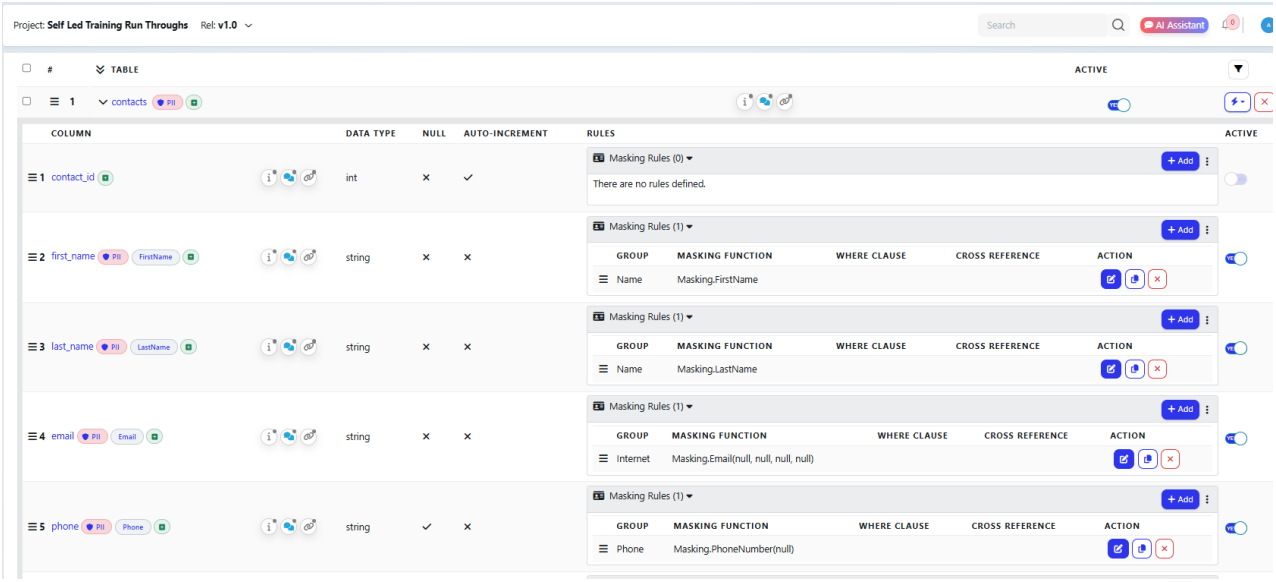
There are two ways to create rules:

1. **Use global defaults**. These are rules that are defined globally. A global rule set is standalone and can be applied to any rule set to copy over the masking rules. This is a fast and powerful way to reuse masking rules between databases.
2. **Add individual masking rules onto each column**. This allows you to specify a masking rule for each individual column.

To edit a rule set, select the **'Modify'** action for your newly created rule set, then click the play button.

Individual masking rules

To add a masking rule to an individual column, locate the column you want to apply a rule to and open the Masking Rules dropdown. This will show any rules that are already in place. To create a new rule, click the **'Add'** button.



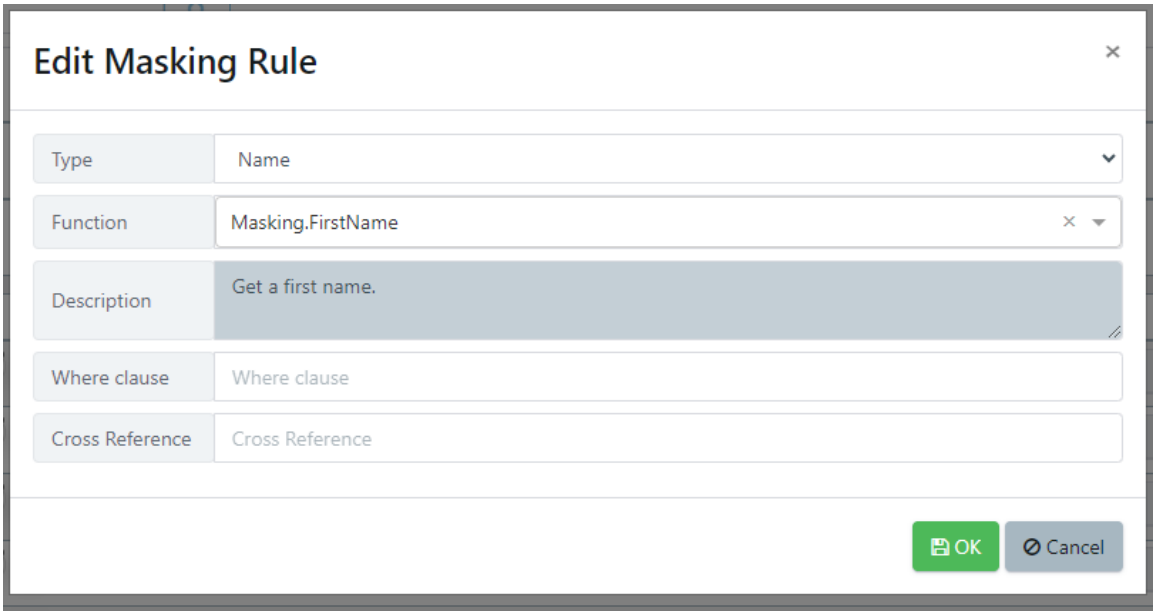
This will open a screen with the following fields:

- **Type:** This is the type of masking rule we want to apply. TDA has many different types of rules built in (e.g. address). To reference a list, select General.
- **Function:** This is the specific masking rule we would use. These are dependent on the type of masking rule chosen above.

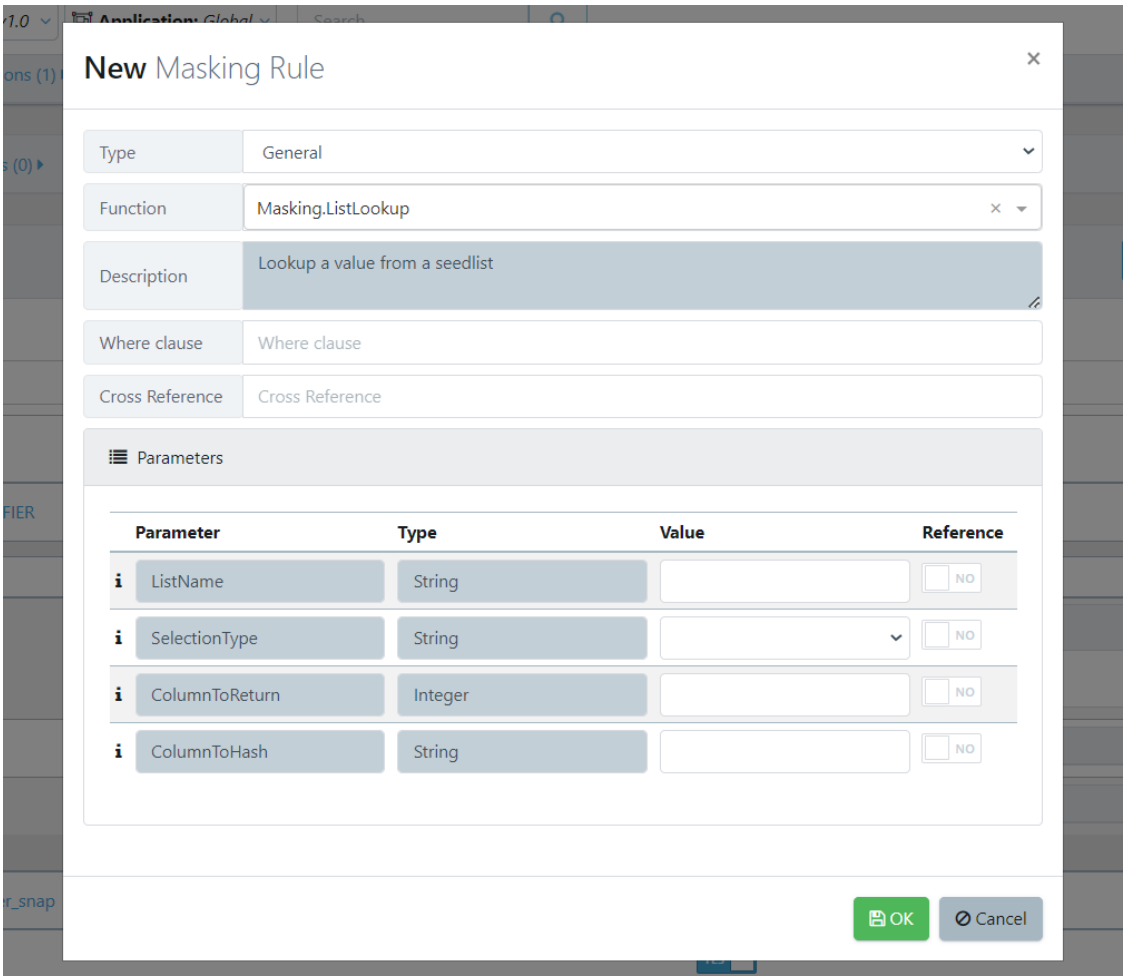
- **Where clause:** This allows you to apply a where clause statement to your rule.
- **Cross reference:** This allows you to enter a database name to cross reference.

Depending on the type and function you've chosen, you may see a drop down with parameters.

Let's look at a simple example which applies a masking rule to replace the Name column with a random first name, by using the 'Masking.FirstName' data generation function.



Another example below uses a List lookup, to lookup data from a seed list.

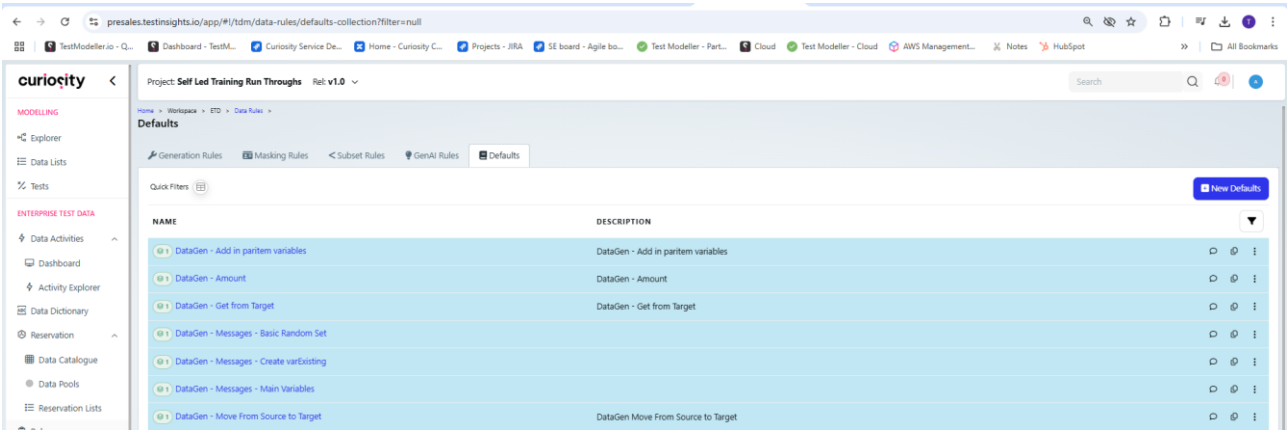


- **ListName**: This refers to the name of the seed list you want to use.
- **SelectionTypes:** This allows you to choose the way in which values from the seed list will be chosen.
    - For a given column, Random selects a row from the database at random
    - For a given column, Sequential will go down each of the roles sequentially and start from the beginning again once exhausted.
    - Hashing is the process of transforming any given key or a string of characters into another value. Hashing will take a column value, hash it to provide a large number, then modify that number by the number of rows in the list to tell you which row to use. This provides the same column value, same hash algorithm and same rows in the list to provide the same result wherever the hashing is run.
- **ColumnToReturn:** This is the column index in the list to use. This defaults to 1, but you can have a multi-column list.
- **ColumnToHash:** This is the column to hash if you use hashing to select the row. This defaults to the column being masked.

Creating global defaults

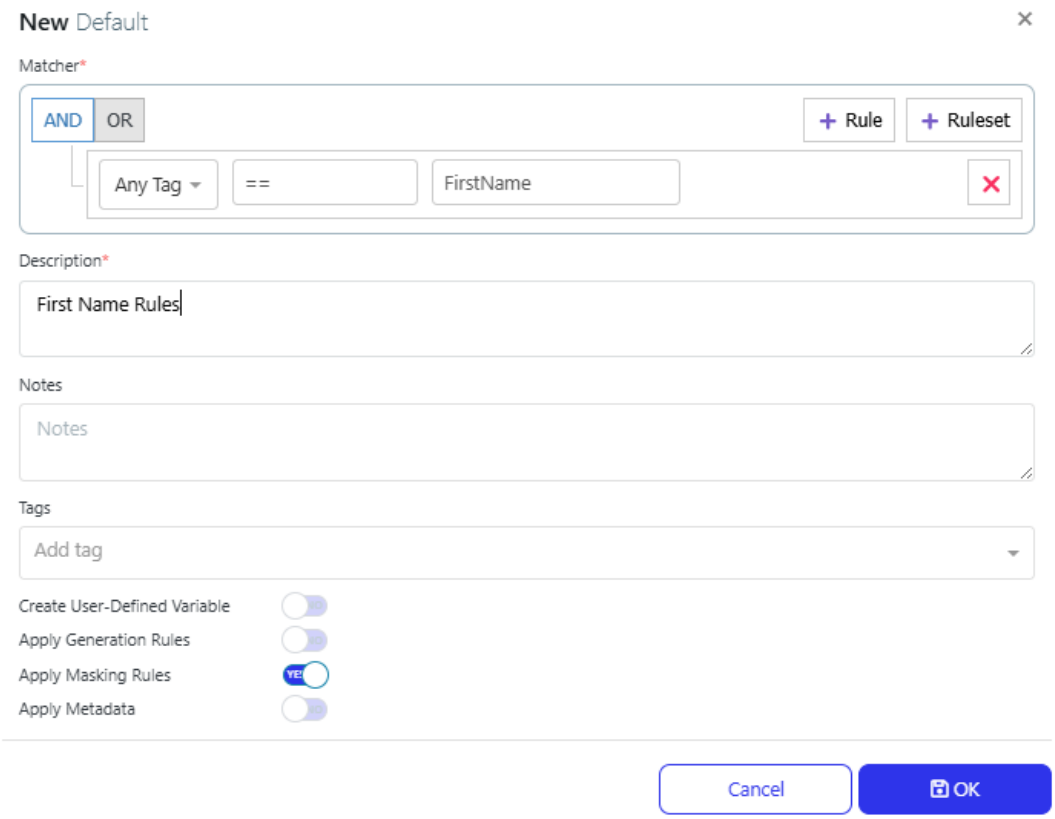
Global defaults refer to universal masking rules that can be integrated into any masking rule set. This enables the creation of universal rules that can be implemented collectively across various rule sets. This feature is particularly beneficial when masking multiple databases.

To view the global defaults in your space, navigate to **Enterprise Test Data → Ruleset Explorer → Defaults**. This page will tell you whether a tag is applicable to use for global defaults in synthetic data generation, masking, or both.

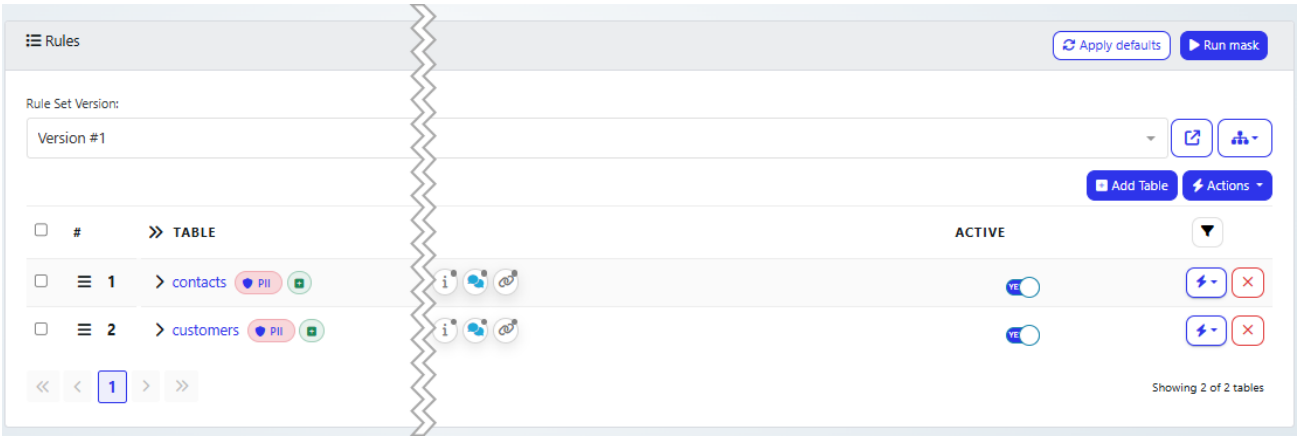


To edit a global default, click the '**Edit**' button. This will allow you to configure your tags and values.

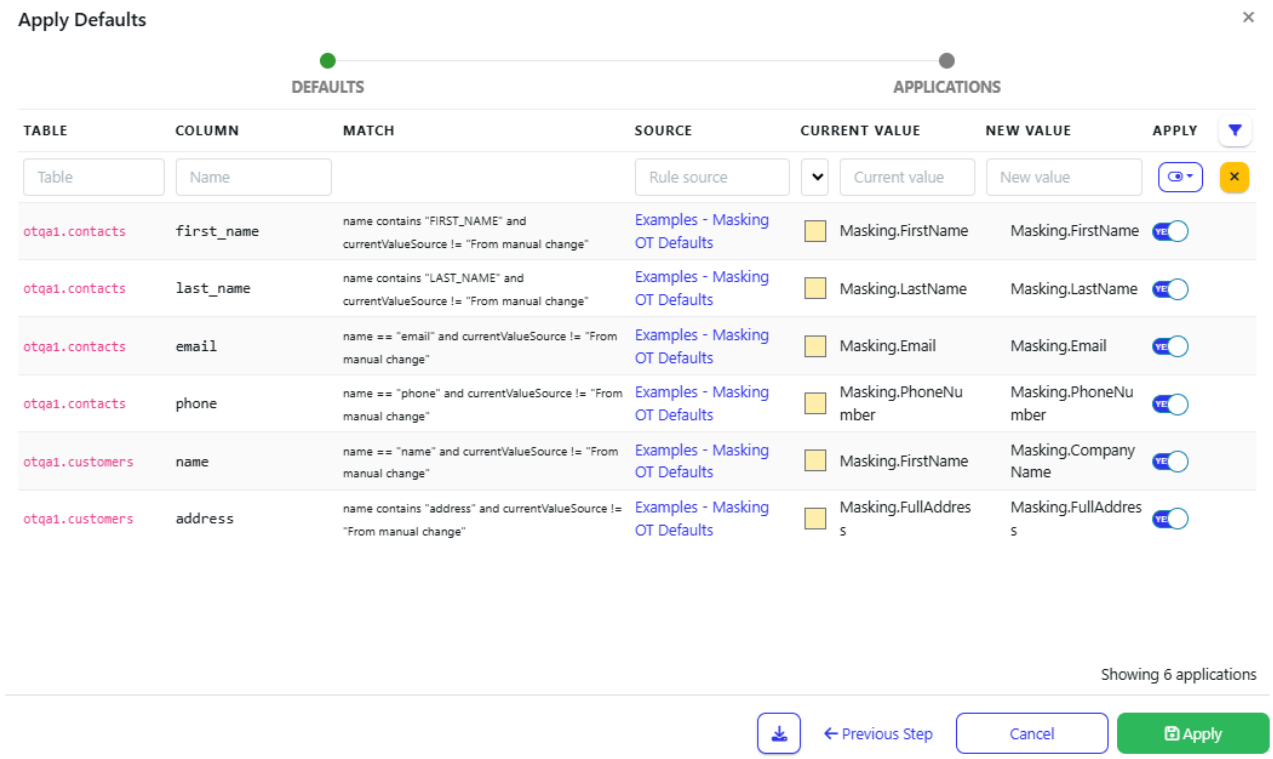
Make sure you click on the toggle for '**Apply Masking Rules**', so it turns blue.



To apply the global defaults to your masking Data Activity, click on the **'Apply defaults'** button.



This will bring up a screen that shows which columns in your database will be affected by a global masking rule. It provides the option to apply the rule. Click '**Apply**' once you're finished to apply the rules.



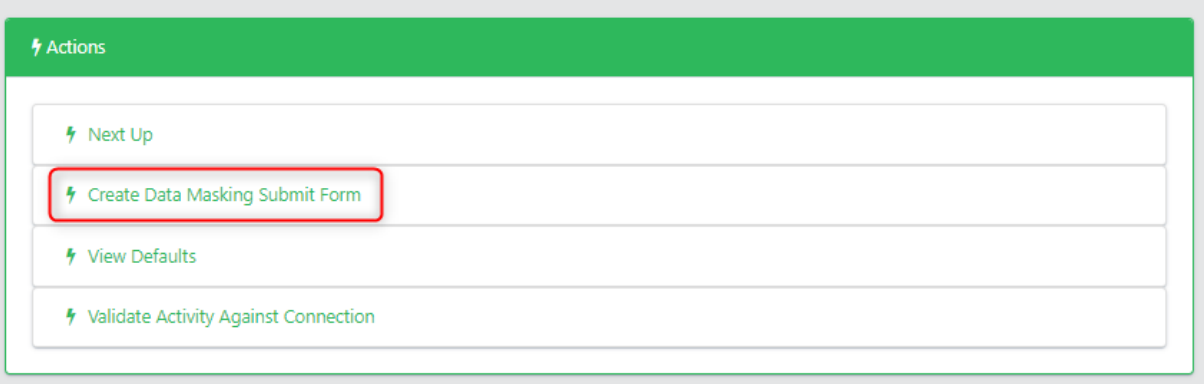
Exercise 2 – Create masking rules

a) Create a masking rule using one of the built-in masking functions such as first name
b) Create a masking rule using a seed list
c) Create a masking rule that uses the value from another column

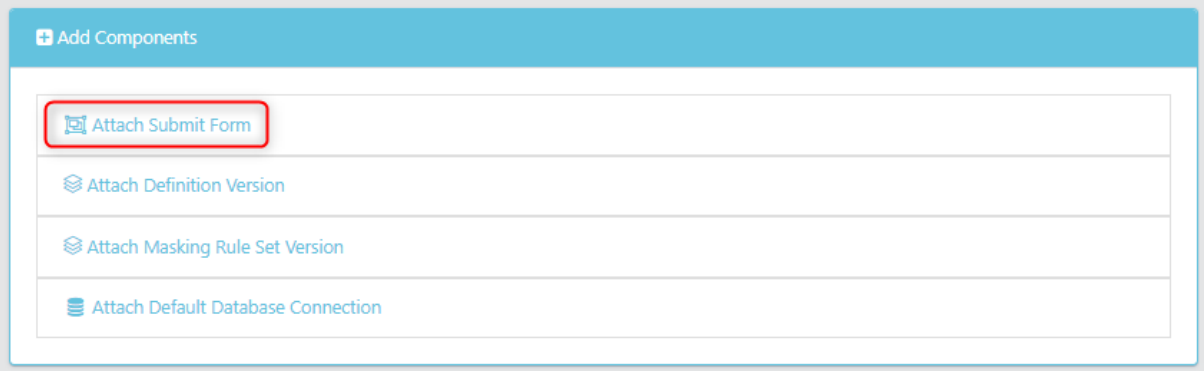
Step 4 - Create masking submit form

A submit form is a reusable form which can be embedded into the self-service portal for future use. Whenever any user wants to perform the defined masking activity, they can do so using the created form.

From Actions, choose **'Create Data Masking Submit Form'**.



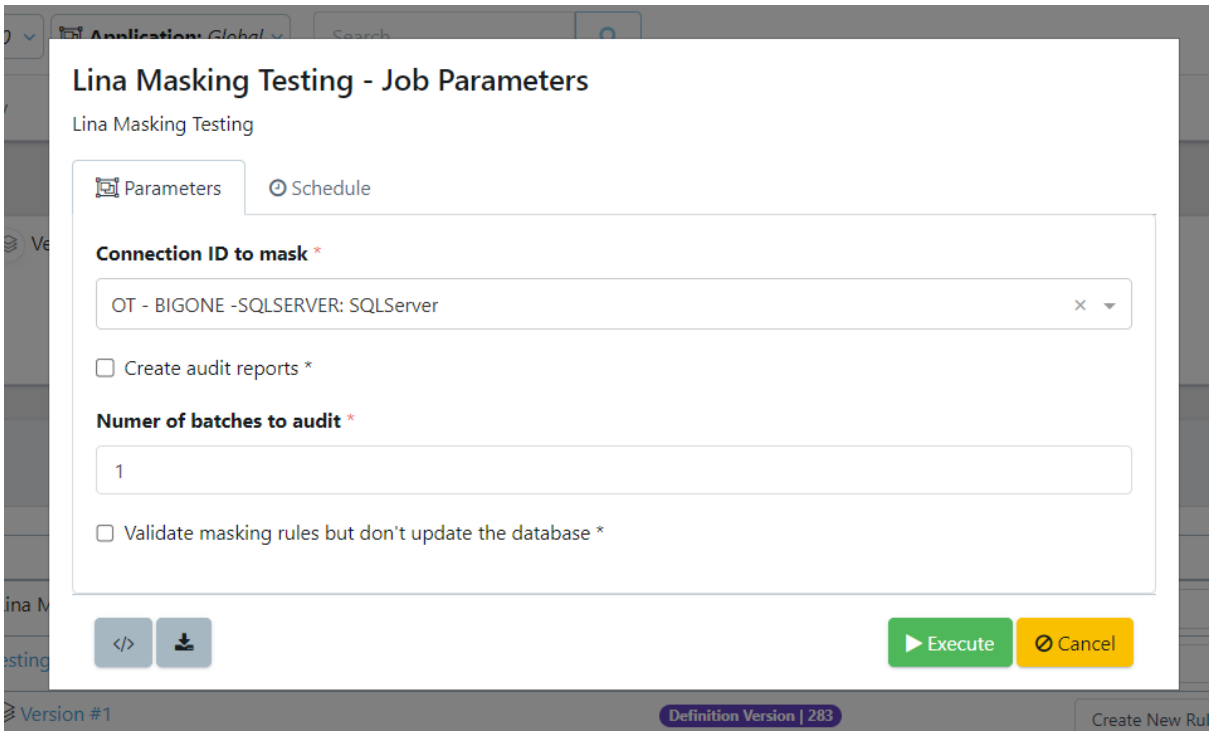
To use a form that has already been created, select **'Add Components'** > **'Attach Submit Form'.**



Step 5 - Mask database

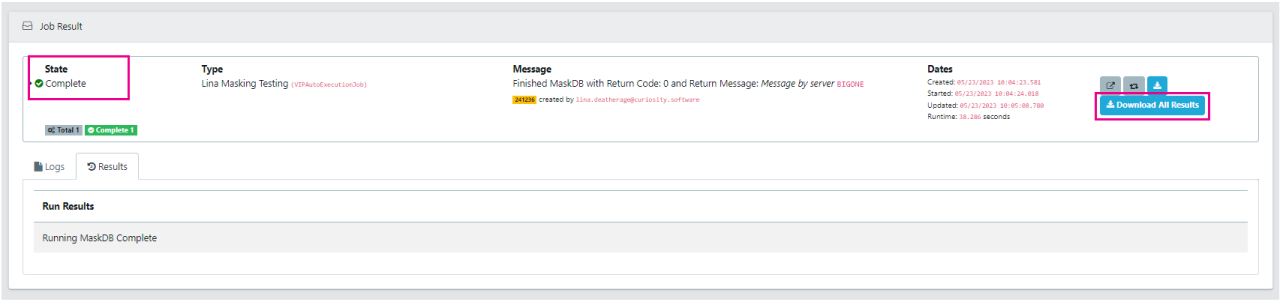
Now we can execute our masking job. From the top component, choose **Execute** as the Action, and press the play button.

A form will appear to input your job details and parameters.



1. Select the Database Connection ID to mask
2. Selecting Create audit reports will create excels which can be downloaded at the completion of a job. This will show the changes made to the database.
    a. **This will contain the data you are masking, so this is recommended only under special circumstances.**
3. The number of batches to audit: Enter the size of a batch of rows to use in updating the database
4. Validate masking rules but don't update the database will check if your rules are working as intended without altering your database

Your job will now run, and if the audit reports option was selected, provide a zip file to download.



The database will be masked with the rulesets specified applied. If you have enabled auditing you will see an AuditResults.xlsx file which contains the pre-masking and replaced post-masking values.

Below is an example report where the first_name, and last_name column values have been replaced with synthetic values.

	A	B	C	D
1	Pre/post	id (PK)	first_name	last_name
2	Pre	5	Marcellus	Wehner
3	Post		Joan	Wunsch
4	Pre	72	james	Smith
5	Post		Miller	Romaguera
6	Pre	76	Nicole	Smith
7	Post		Kurt	Abbott
8				

Exercise 3 – Create and execute submit forms

a) Create a submit form for masking

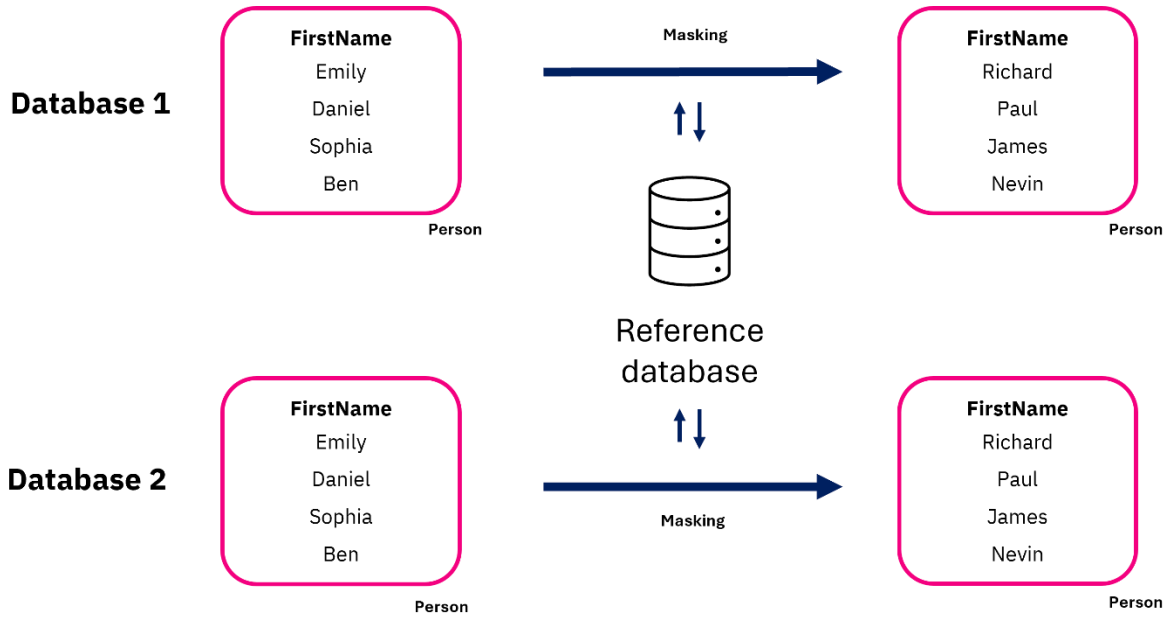
b) Execute form with the validate option to test the execution

c) Execute form with validate and audit report checked to see the results of the masking

Consistent masking and conditional masking

Consistent masking a cross-reference table

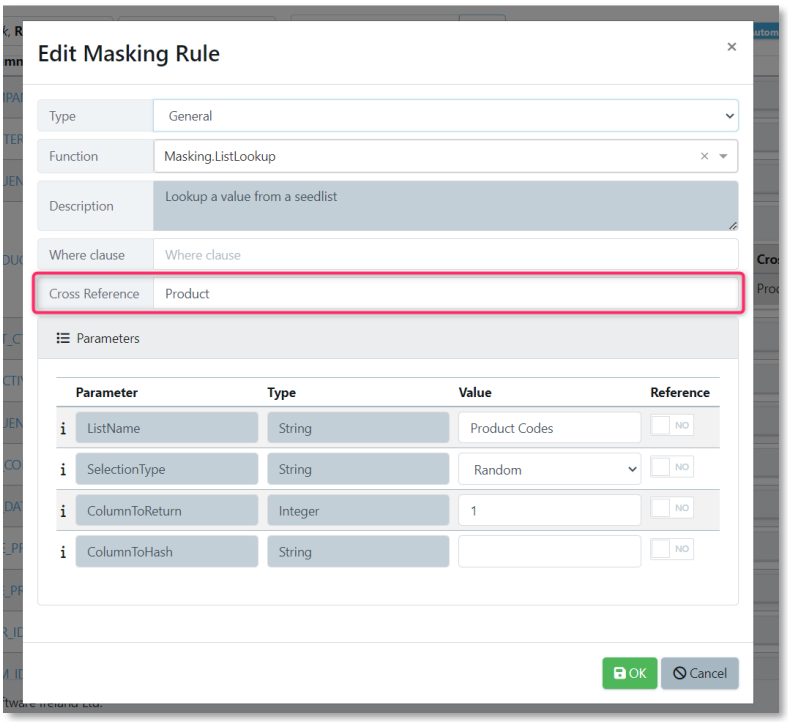
At Curiosity we use a reference database to enable our cross-reference functionality which is core to our ability to mask across multiple systems and keep referential integrity.



The cross-reference database comprises a table featuring columns for a cross-reference identifier, a pre-masked value (hashed), and a post-masked value. During the masking process, a lookup is performed based on the identifier and pre-masked hash. If a matching entry is located, the corresponding post-masked value is applied. In cases where no match is found, a new row is appended to the table.

FirstName		Hash Value		Masked FirstName
Emily		12345		Shannon
Daniel	Hash →	15423	Mask →	Dave
Sophia		43215		Michelle
Ben		51423		Finn
Benjamin		12432		Laura

To use the cross referencing ability during your masking routines you need to make sure the cross reference connection has been set up and configured in the masking configuration section. You can then select a name in the cross reference section as shown below:



Note: For each unique name the referencing will be different. For instance if I am masking the first name John and the name for one masking ruleset has First Name in the Cross reference section and the other has Name in there then they will be masked to different values.

Exercise 4 – Use cross-reference functionality

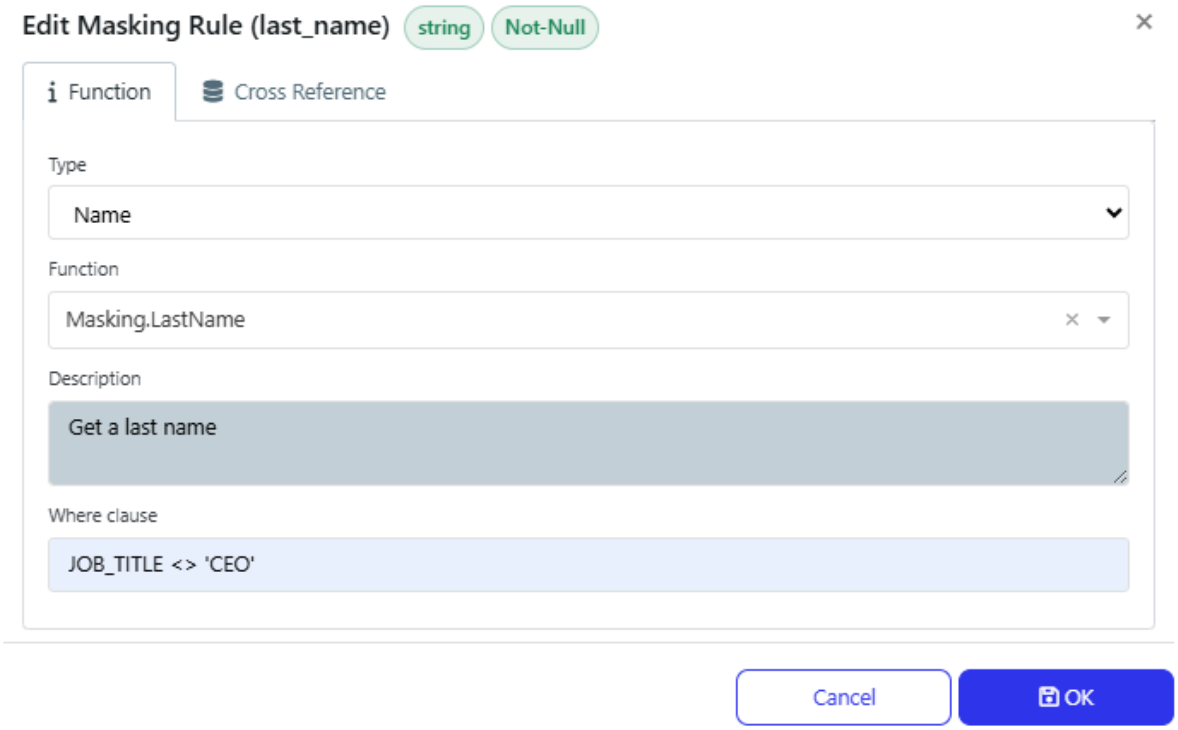
a) On one of the columns that you have previously set up masking rules for add a name for your cross reference section and then run

Consistent masking using hashing

Another method that can be used to achieve referentially integral masking is by using the hashing feature mentioned in List lookup function. By using this feature the tool converts a value, such as First Name, to a hashed value which relates to the row ID of the seed list. For instance 'Toby' could be taken and given a hashed value for the row ID of 7 in a seed list that could point to the value 'Harry'. Thus every time Toby needs to be masked, the value of Harry will be used.

Conditional masking

You can also choose when to apply masking rules to your data set using the **'where'** section in the masking ruleset, such as below:



The syntax is as follows [columnName] [Operator] [Value] such as the above JOB_TITLE <> 'CEO' making sure that any string values are enclosed in single quotes

Exercise 5 – Implement a conditional masking rule

a) Choose one of your masking rulesets and choose a column to implement a conditional rule