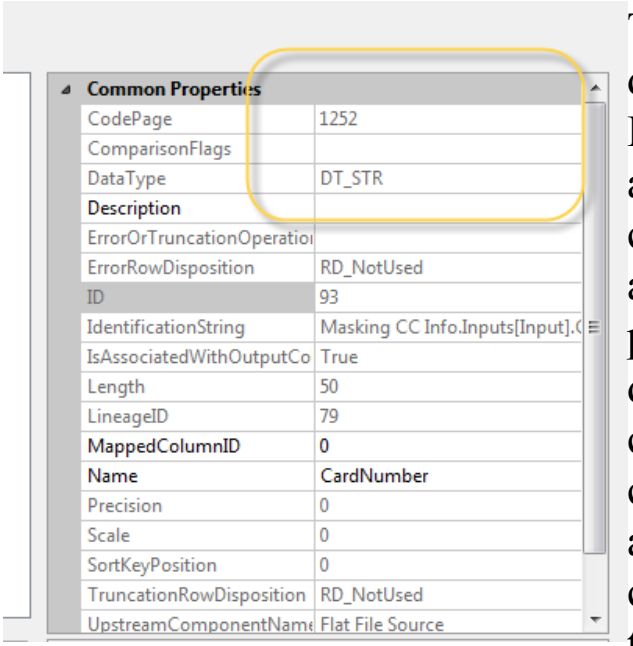
HUSH HUSH

MASKING SSIS COMPONENTS MANUAL

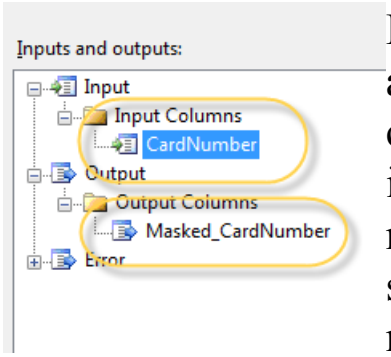
Table of Contents

INTRODUCTION

Hush Hush is a collection of SSIS data flow transformation components that de-identify data entities while they retain the format and the rules specific to these entities in the output. The components use industry-standard types of algorithms of data-de-identification and some of them utilize pre-populated dictionaries. The components follow the standards of SSIS data flow components in the rest of their properties.
The majority of the components expects **string type** data on input and provides **string type** data on output.



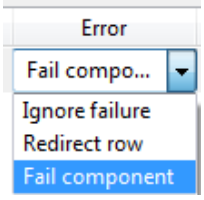
The exceptions to “string data type” rule are the **Masking Decimal Number** and **Date of Birth** components. Using numeric and date type data as strings proved challenging as these data required complicated conversion syntax, so these components use any numeric and date types. See the corresponding sections of the manual for instructions on how to handle them.



Every component requires at least one and allows multiple fields on input. When the component receives the input field and identifies its name, type, and other metadata, it creates another field of the same data type and length that has the masked value in it.

The name of the new column is a concatenation of the word “Masked”, underscore, and the input field’s name. Each component is strongly typed. It expects specific types of data. Upon input, it will confirm the data type and the rules of the data element formation based on the algorithm of the component. If input rules do not conform to the expected rules of element formation, the component will fail. There are a handful of algorithms that are based on the ISOs or accepted practices, while the rest are just substituting the values.

In case of the input or transform error, there are three types of error handling that allow for a “fail component”, “redirect row”, and “ignore failure”.



ALGORITHMS

The components employ the following major algorithms:

Random substitution algorithm substitutes an input value with the random value suitable for that element. Random components replace data in a truly random fashion. This algorithm is safest in the sense of the re-identification; however, output values could create collisions for the elements with the uniqueness requirements. For example, the random transform of a social security number "111-11-1111" can generate the number "387-08-0973", and if another social security number has already generated this value, the values will collide in case social security values are unique in the table.

Substitution Preserving Referential Integrity algorithm is a proprietary substitution algorithm that substitutes given input value with the same value consistently and repeatedly across the enterprise, no matter the source of data. This algorithm adequately secures sensitive data from the re-identification efforts. At the same time, it both maintains uniqueness of such elements as social security and credit cards (which are unique per individual) and allows for disturbances of statistical distribution of the zip codes, names, cities and other elements with existing known statistical distributions.

Shuffle algorithm moves, or "shuffles", data values along the column. This algorithm is often necessary to use when preserving the value of the aggregate or in other use cases when one needs to keep the data set the same.

Number and Date Variance add values to the given input value within the specified interval of confidence. For example, Birthday Date component would create birth dates within 2 day interval of the original date if the practitioner limits the variance to two days. August 1st in such case may be replaced with July 30th, July 31st, August 1st, August 2nd and August 3rd.

Character Substitution Algorithm exchanges values of the alpha numeric characters in the given values with either random or in consistent manner, depending on being random or Dynamic.

Majority of algorithms use the key in calculating mapping tuples and will change the substitution values when the practitioner changes the key using the "Key Utility".

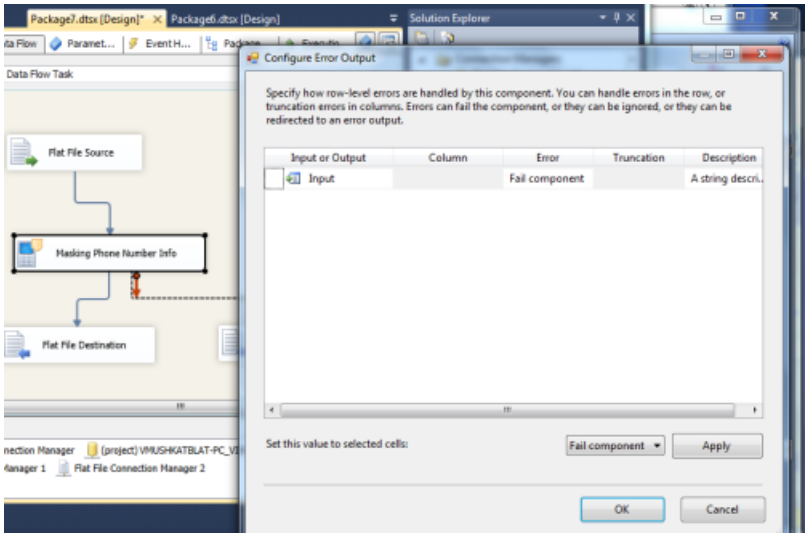
NAMING AND OTHER CONVENTIONS

All the components use the value's entity type in the title. For example, for the components that randomly transform social

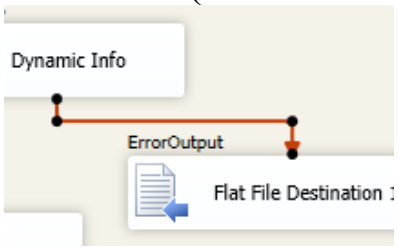
security numbers, the name will be “Masking SSN Info”. The components that consistently substitute an output value based on the same input value; the name will contain the word “Dynamic” in the title. For social security numbers, for example, it will be “Masking SSN Dynamic Info”.
Social Security Numbers, Credit Card numbers, US Phone numbers, and Phone Numbers, and Generic Alpha-Numeric dynamic components guarantee that unique data sets will map to a corresponding unique data set. The rest of the elements such as Names, Addresses, City Names, Country Codes, URLs, Dates of Birth, Decimal, and Zip Codes do not guarantee unique data set on output but provide sufficient amounts of output values to test upon for various KPIs.

ERROR DISPOSITION

Error disposition works in all the components. It allows you to fail components, to re-direct rows, and to ignore failures. In the first case, the failed component stops the flow. In the case of re-directing, the component directs erroneous rows into the destination that one created to store erroneous content. Ignoring failure will only transfer successfully validated values, without attending to the errors. The “Ignore failure” option might result in the loss of data.
Error disposition rules are limited to the rules of the entity:
checks whether numeric components’ (SSNs, Credit Cards, Phones and Zips) input values consist of digits;
checks that email has a format of xxx@xxx.xxx;
checks (in a very limited fashion) URL format;
checks Luhn calculation in Credit Cards.
It is important to notice that all the numeric components assume that non-alphanumeric characters are likely the separators and preserves them intact.
To set error disposition, connect your component with the following component in the flow with the precedence constraint:



The Configure Error Output window opens and contains the "Error" Column. That should be set to the desired behavior. Default SSIS architecture does not allow you to open the window again to change set error handling behavior. In order to change it, one has to delete and re-create the precedence constraint (the red arrow)



CREATING TEST SETS FROM SCRATCH

Random components allow you to create test sets when the data in production does not exist. In text-based components such as names, one simply can provide a "dummy" value in the necessary fields, such as "John" or "abc" – and the component will create a random set of values.

In this case, use SQL query to create the templated value. We suggest the following:

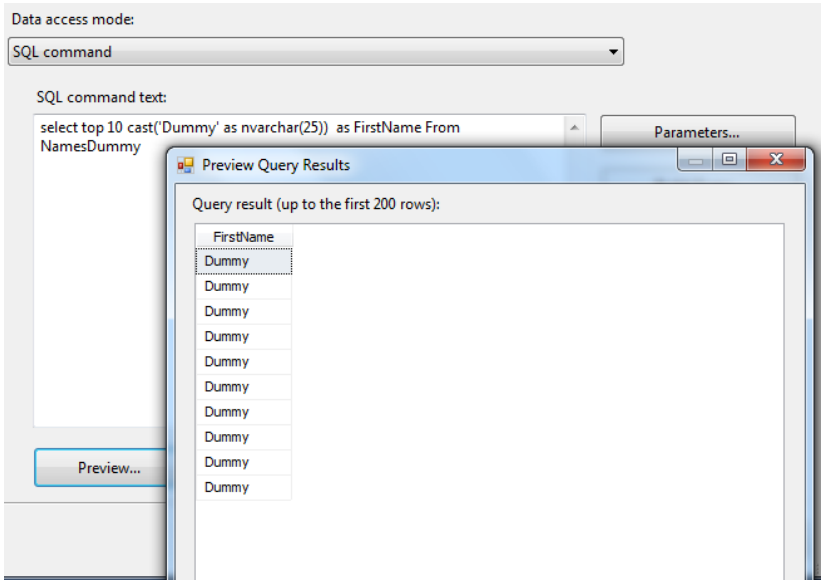
Use any value for input for the strings type components.

Cast or convert it to the one of the character data types with the pre-defined length, i.e. varchar(10) or nvarchar(10).

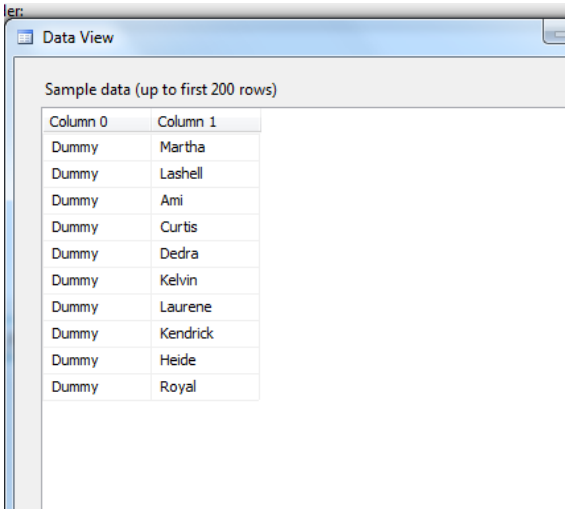
Create an Alias Name, in our example "`AS FirstName`", for the column so that metadata created by the source component will have a name for this column.

SSIS is strongly typed. Not assigning the type to the column in SQL might create issues with converting to the expected data type.

```
SELECT TOP 10 CAST('Dummy' as NVARCHAR(25))  AS FirstName FROM NamesDummy
```



After execution, one will have a list of randomly generated names.

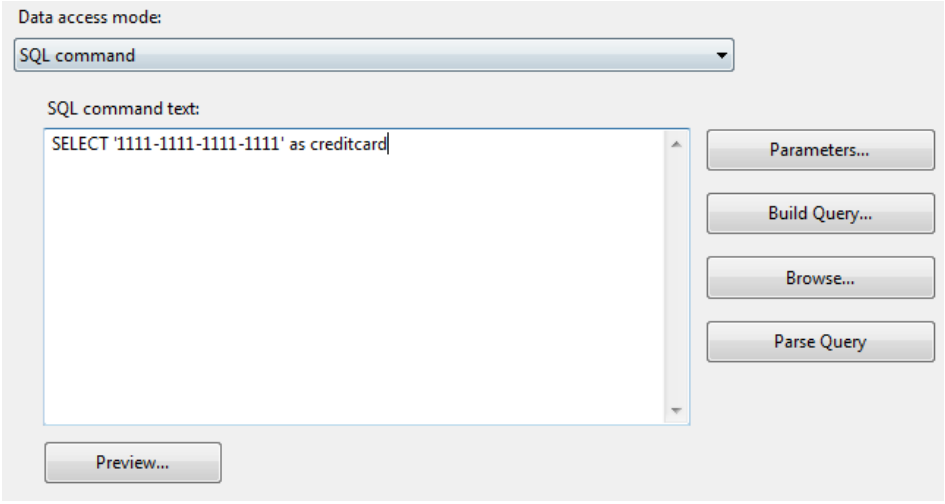


In the example above, we selected a specific number of rows from the existing table.
It is possible to just create a record without selecting from the table. In this case, one would use the `SELECT CAST('xxxxxxxxx' as NCHAR(25)) AS FirstName` construct. Such a construct creates exactly one record. If one wants to make more records at a time, one can use either the JOIN or UNION ALL Operator. Here is an example:

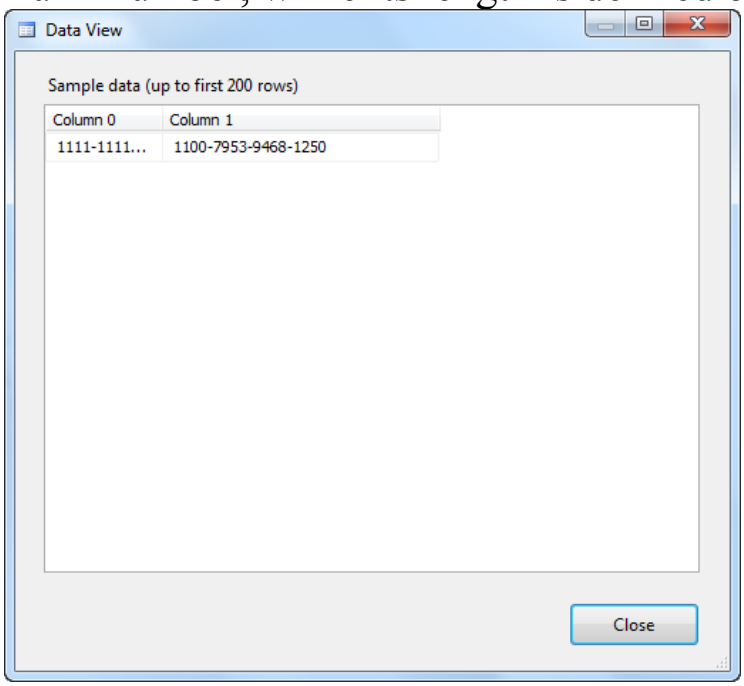
```
SELECT CAST('xxxxxxxxx' as NCHAR(25)) AS FirstName
UNION ALL
SELECT CAST('xxxxxxxxx' as NCHAR(25)) AS FirstName
```

Besides data type and length, numeric components require formatting. For example, Social Security numbers in the USA have exactly nine digits, but you can either separate them with a "-"," " or "." Alternatively, they might not be separated at all. Credit Card lengths vary for different issuers – and initial card

numbers depend on the issuer - so you will have to predefine the length and the issuer.
Before creating data for numeric components that require a predefined format (such as credit card numbers) you must create a format that defines the test set:



The records resulting for the example above will preserve the separators and the format. It will only substitute values and validate rules provided by the component. Credit Card, in the above example, maintains the originating agency number and Luhn number, while its length is defined by the practitioner.

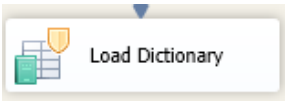


COMPONENTS FUNCTIONALITY DESCRIPTION

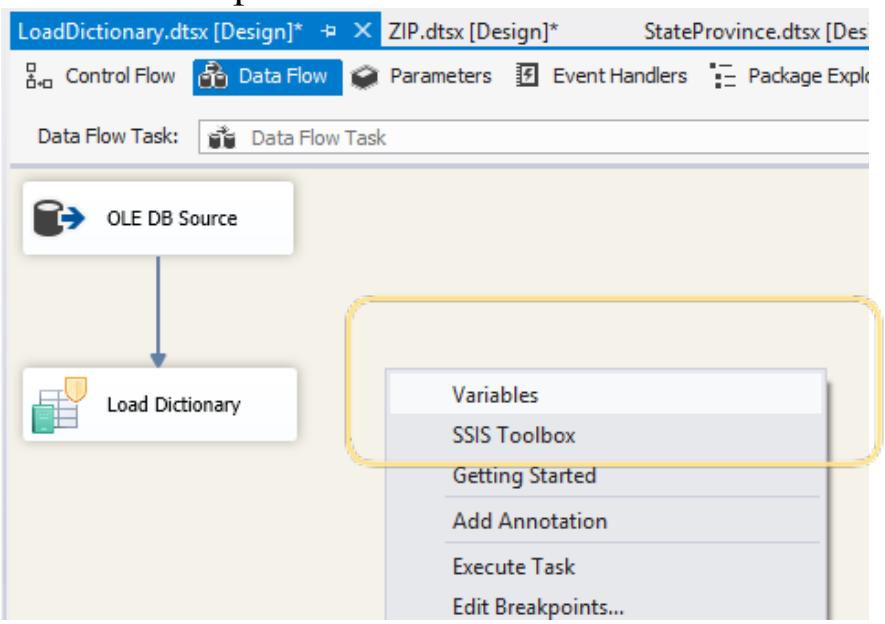
All the components are configured in the same way which we explore in detail in the Masking Country Code Dynamic Component. There are a handful of the components that have custom properties, and we describe properties configuration in

the appropriate section. The Load Dictionary, Mask Substitution and Mask Shuffle components are different in functionality and they have their own description.

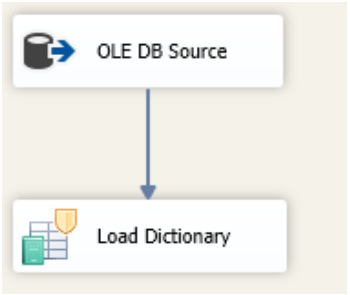
1. Load Dictionary



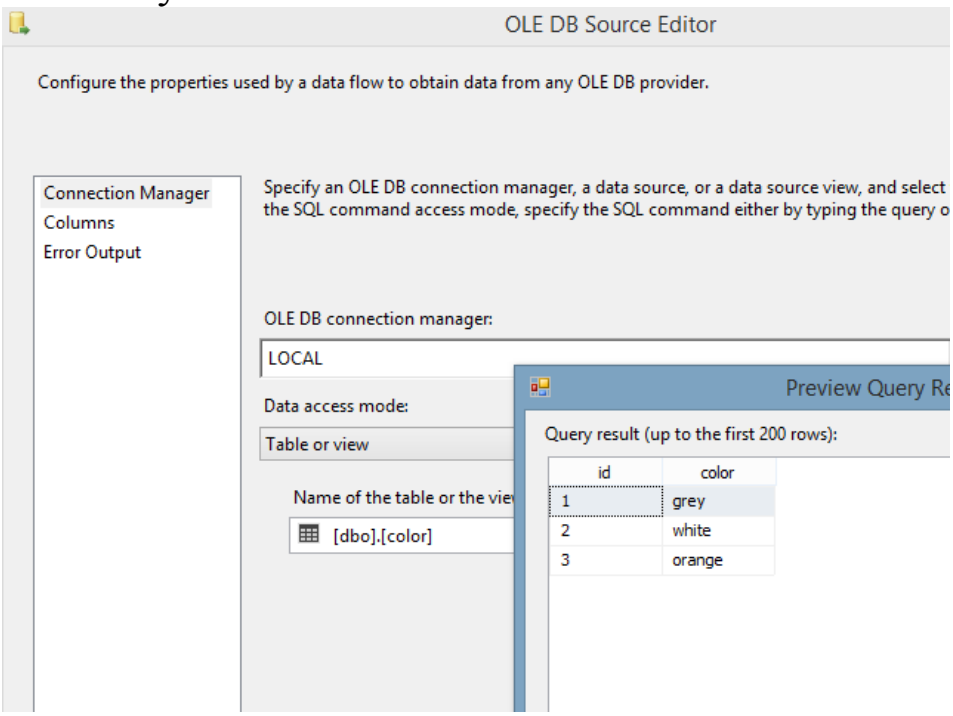
This component creates a custom dictionary of values that are contained in the column of the practitioner's choice. It is the first step in the generic substitution algorithm. Loading dictionary has to be done in its own data flow, followed by another data flow containing Mask Dictionary component. The table from which the data set is selected should contain an unary primary key. Proper operation of the Load Dictionary component requires a pre-set variable of the type object to hold the values. The variable is created by clicking on "New" button in the variable window. The variable window is easily opened in SSIS by right-clicking in the area of the data flow, and selecting "Variables" option.



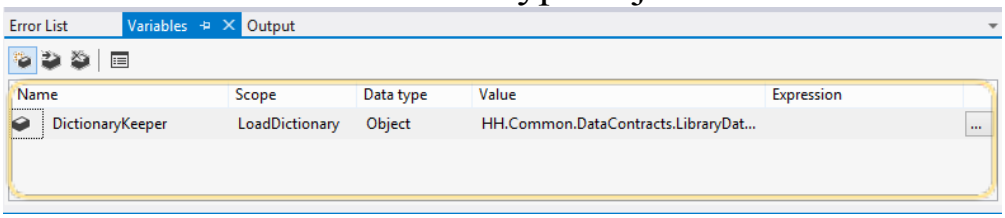
This variable will contain the set of values that becomes a dictionary for the next, Mask Dictionary component. To properly set up the component, do the following steps:
Drag and Drop first the source component and then a Load Dictionary component:



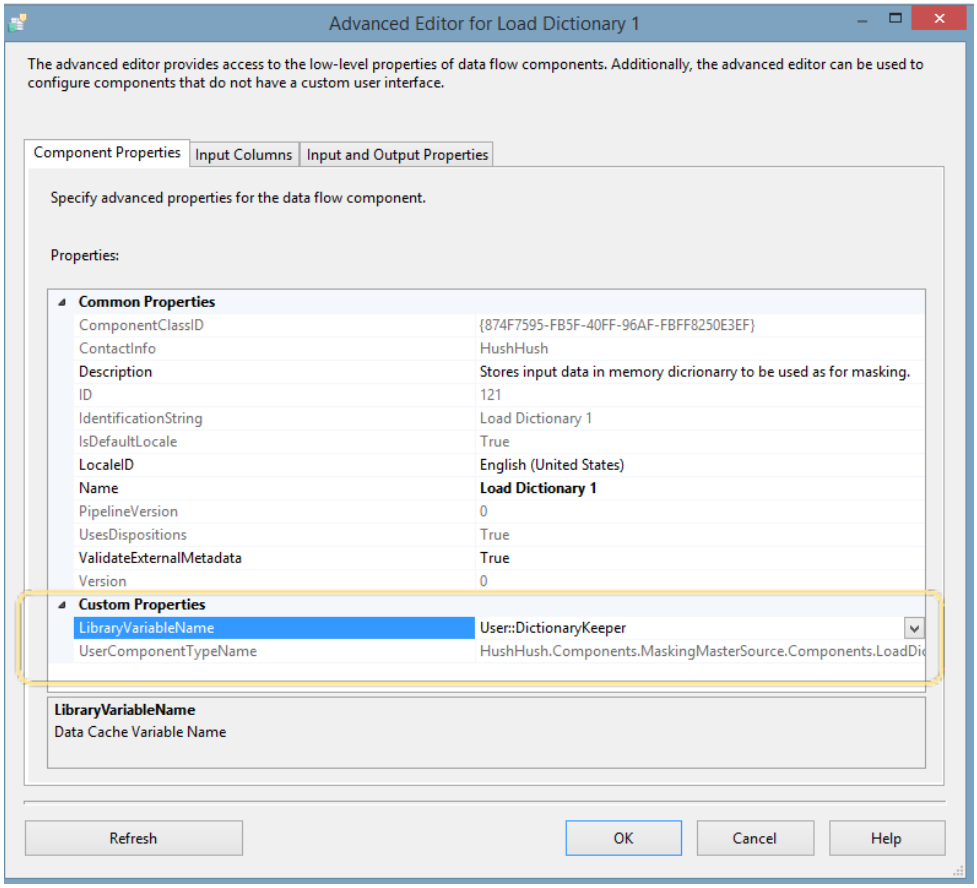
Set the data set in the Source component to the table or file containing the column you want to use to make your own data dictionary to mask with.



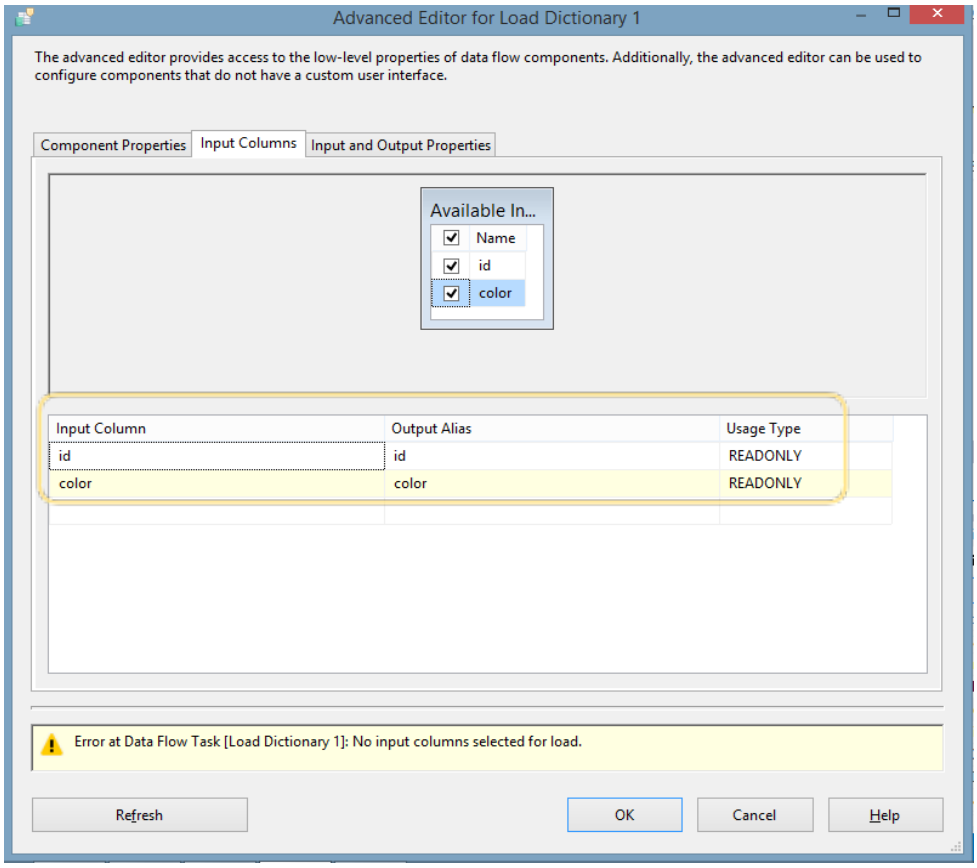
Create a variable of the data type object:



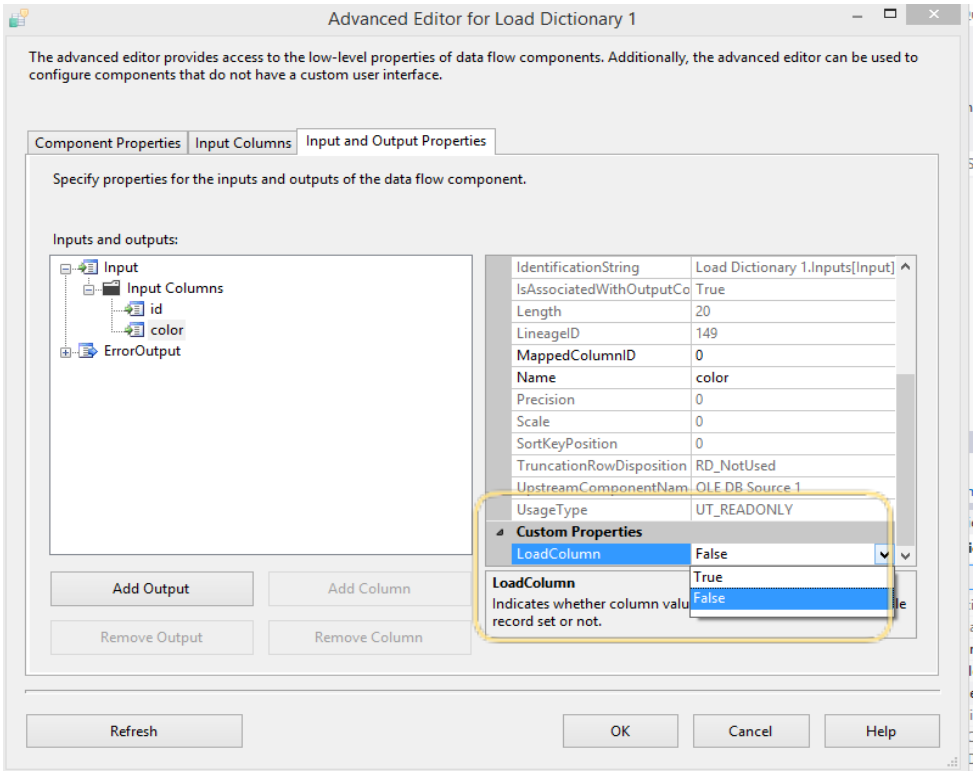
Open the Load Dictionary component's editor by either right-clicking and selecting "Edit" or double-clicking the component. Choose a variable that will hold a dictionary:



In the second tab, choose the primary key column and the attribute column:

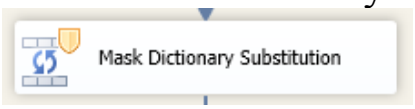


In the third tab, choose the created column that holds the values and set the property of “Load Column” to True.

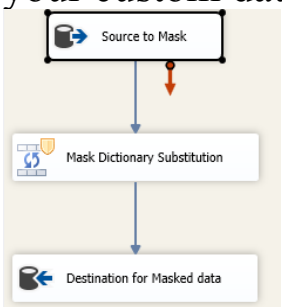


This is the end of the Load dictionary setup – the variable now will load with the index and the values of the element.
You need to configure Mask Substitution component next.

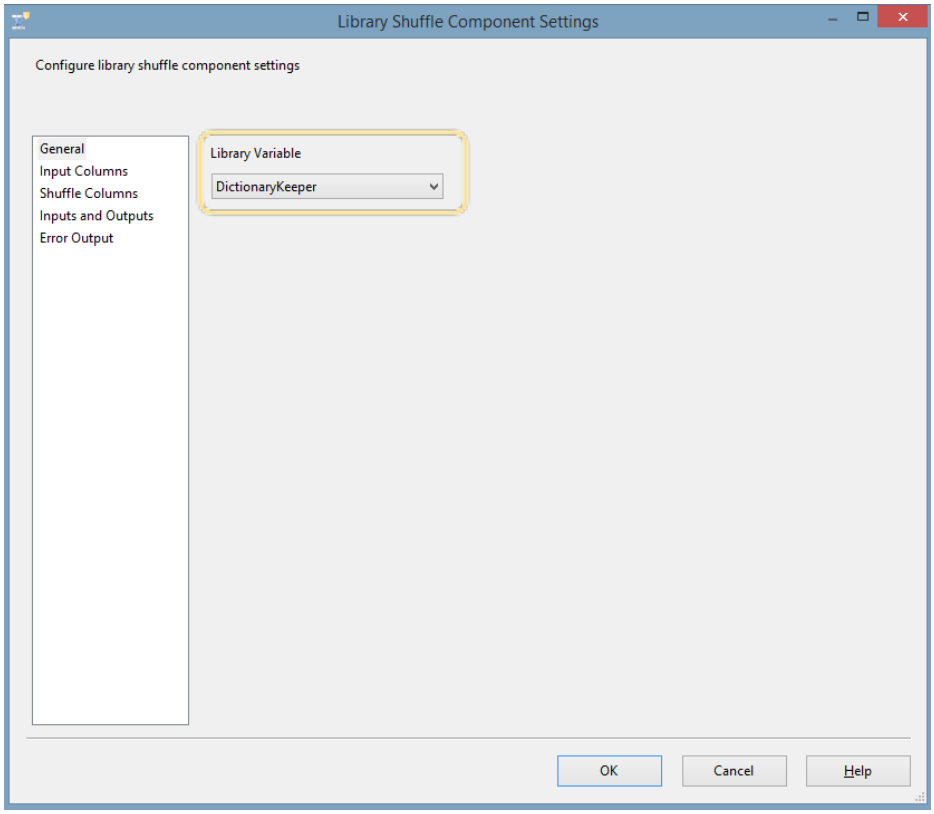
2. Mask Dictionary Substitution



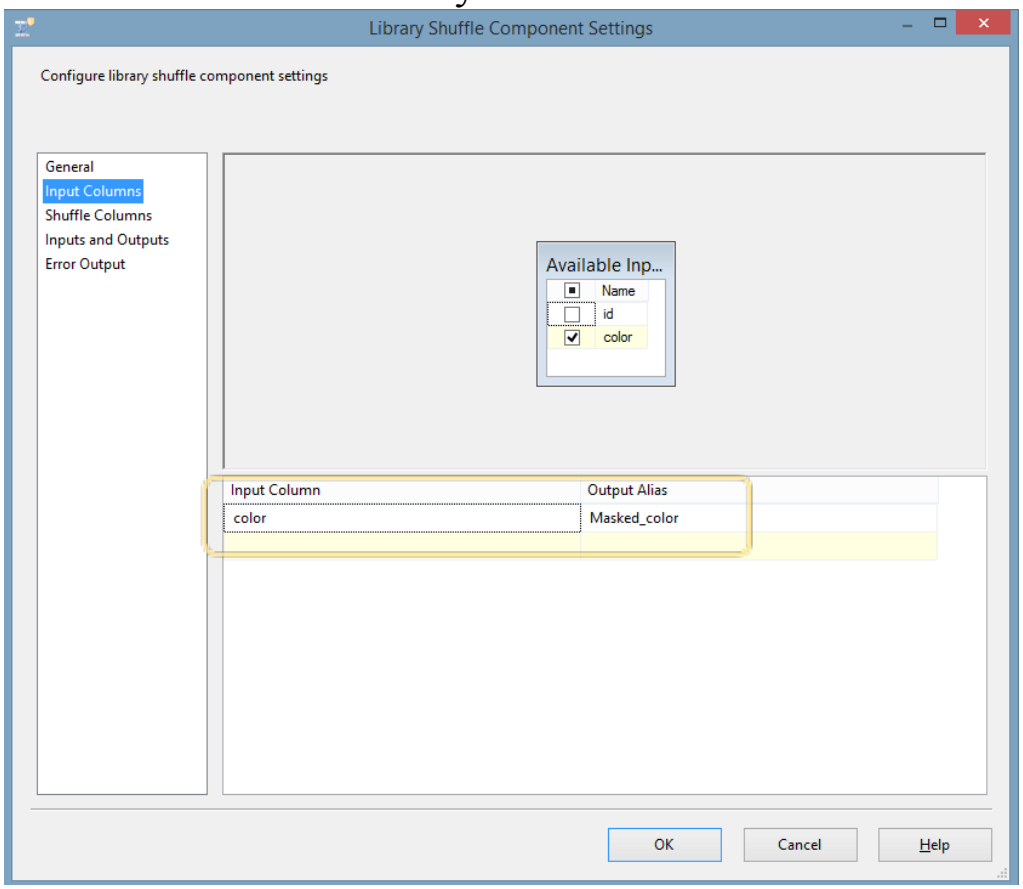
Configure the source component that you want to mask with your custom data in the variable.



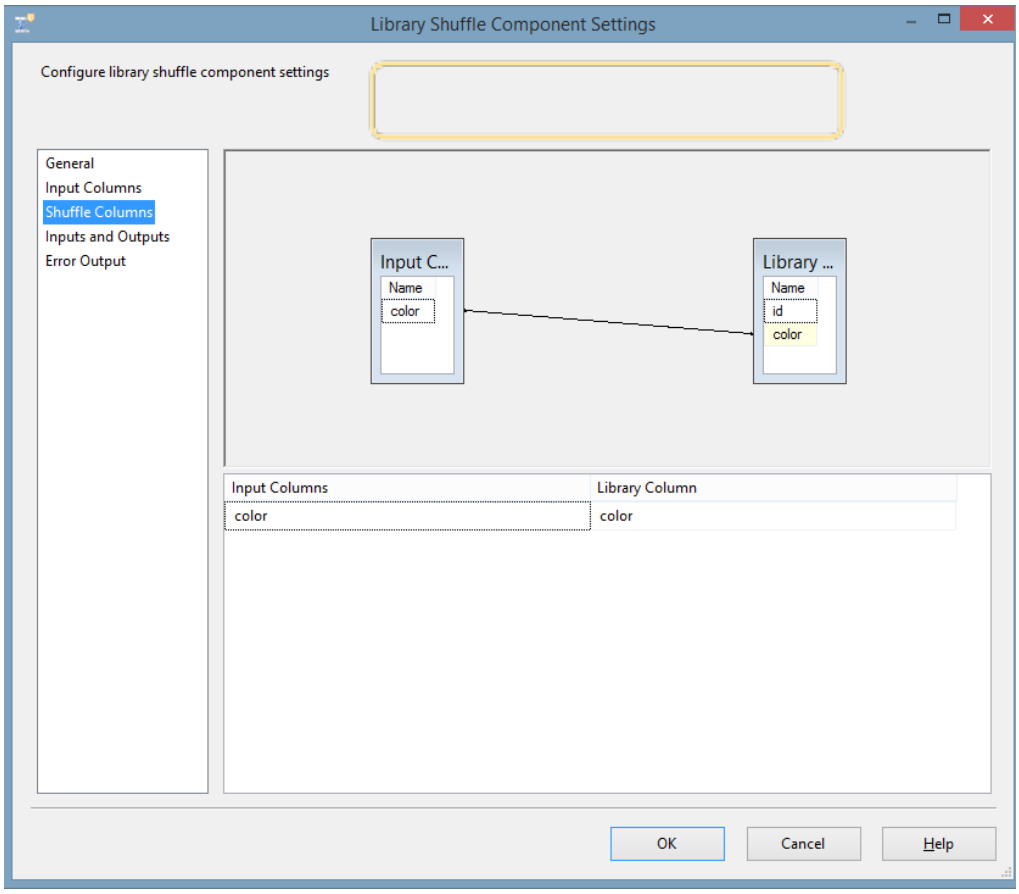
Open Mask Dictionary Substitution component's editor, by either double clicking on the component or right-click and choose edit. Select the variable that contains the dictionary from the drop down:



In Input Columns Tab, select the column you want to mask with the values in the dictionary.

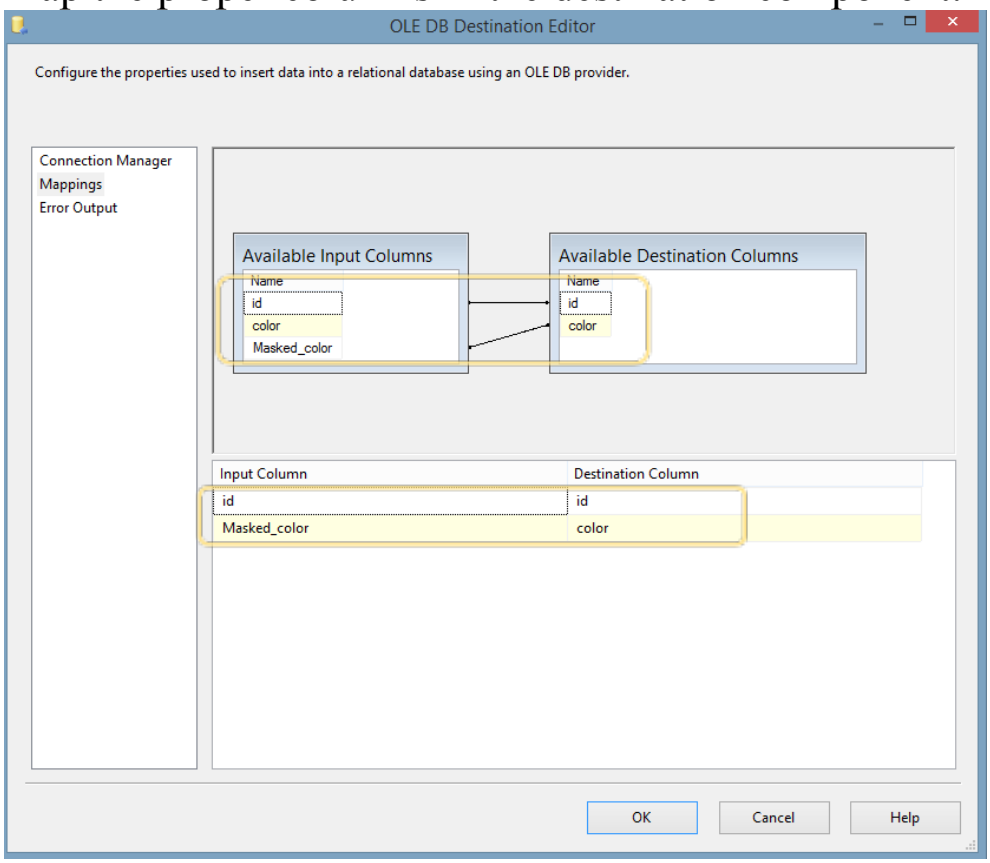


In Shuffle Columns Tab, map input column from the source to the one in the Dictionary Substitution component.



The component's output column now contain Masked_<column_name> containing all the masked values.

Map the proper columns in the destination component:



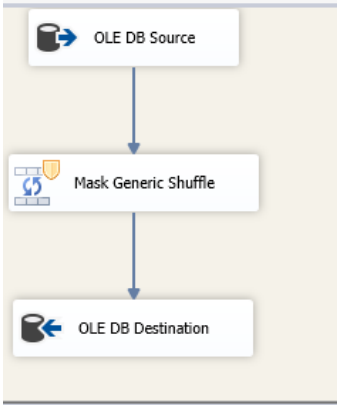
Now, you can run both data flows and mask the values in the column.

3. Mask Generic Shuffle

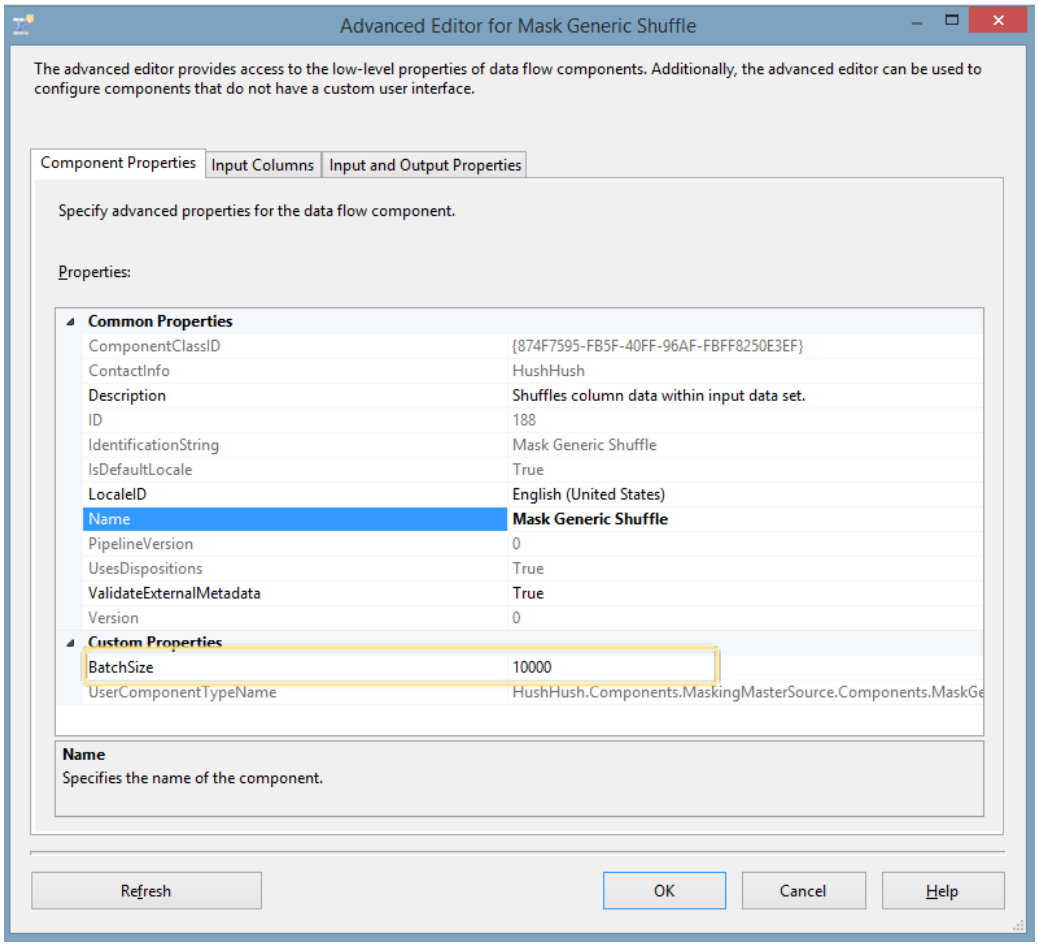
Mask Generic Shuffle component “shuffles” selected column values among the other columns by replacing the values of the column with the values of this very column. One of the columns in the table should be a primary key. It maps a data set of the element on itself. The component starts saving data onto the disk if the memory capacities are exceeded. The batch size is regulated in the custom property.

Here are the steps to use the component:

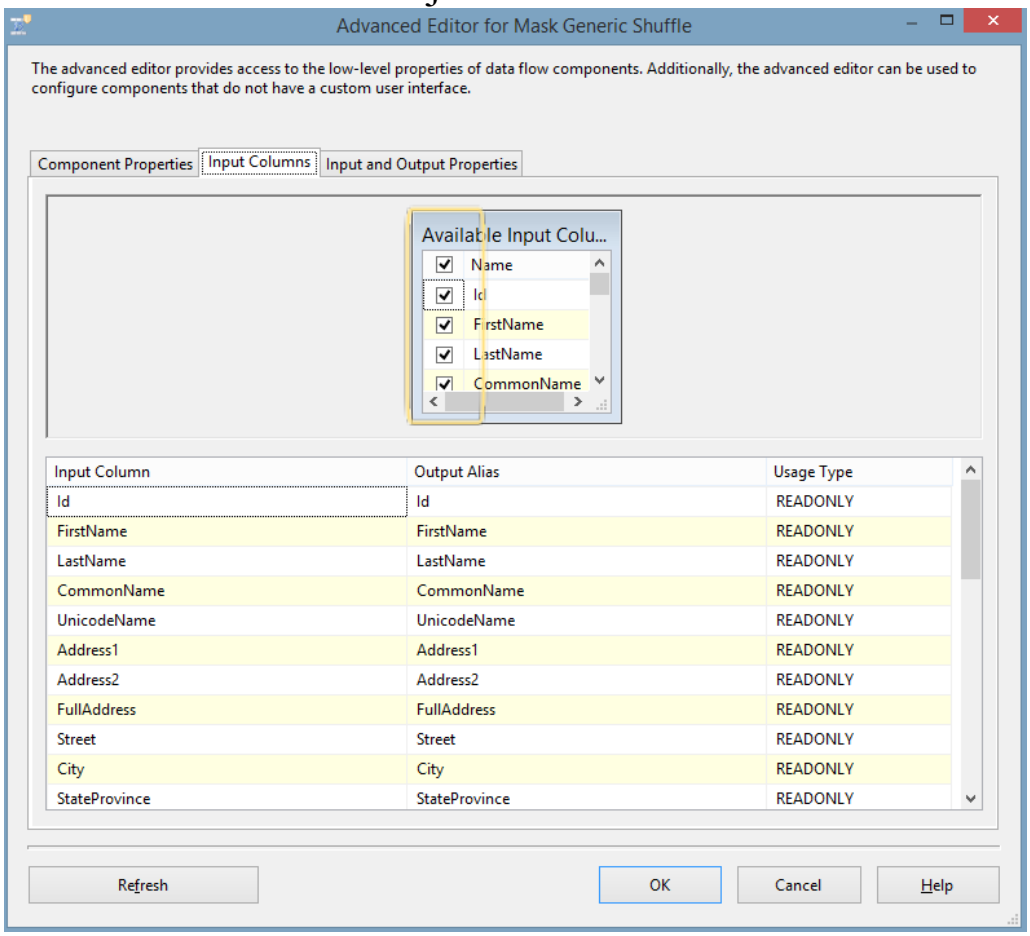
Create a source, a destination and drag and drop the component into the data flow area. Connect component with the source and the destination with the precedence constraints:



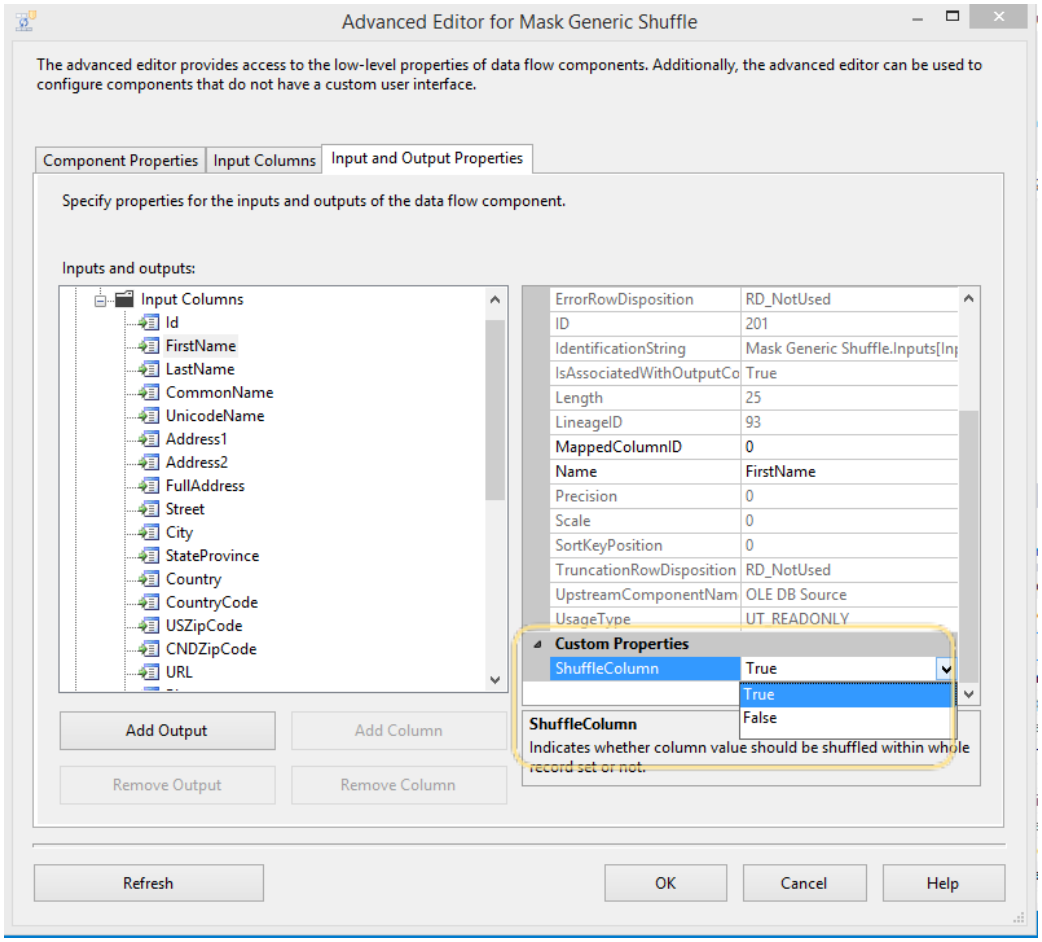
Open the Mask Generic Shuffle’s component’s editor. In the editor, in the first tab, Component Properties, verify the size of the batch that you will be processing one at a time:



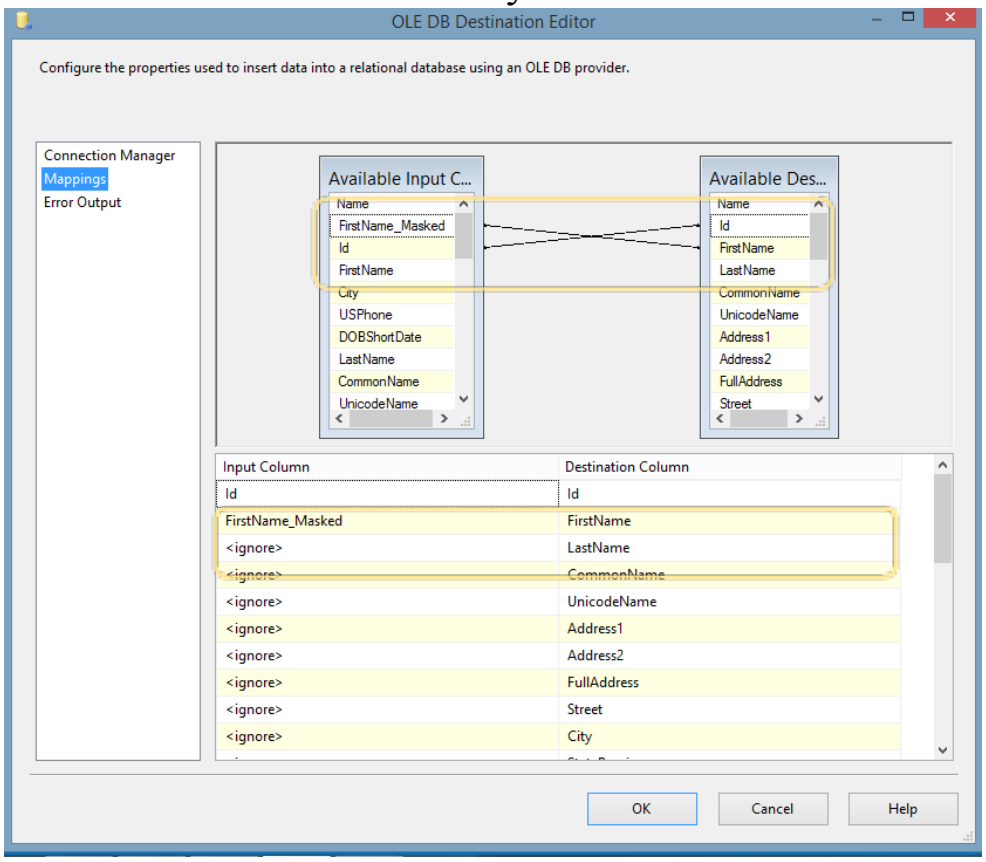
In the second tab in the editor, Input columns, please select ALL of the columns of the object:



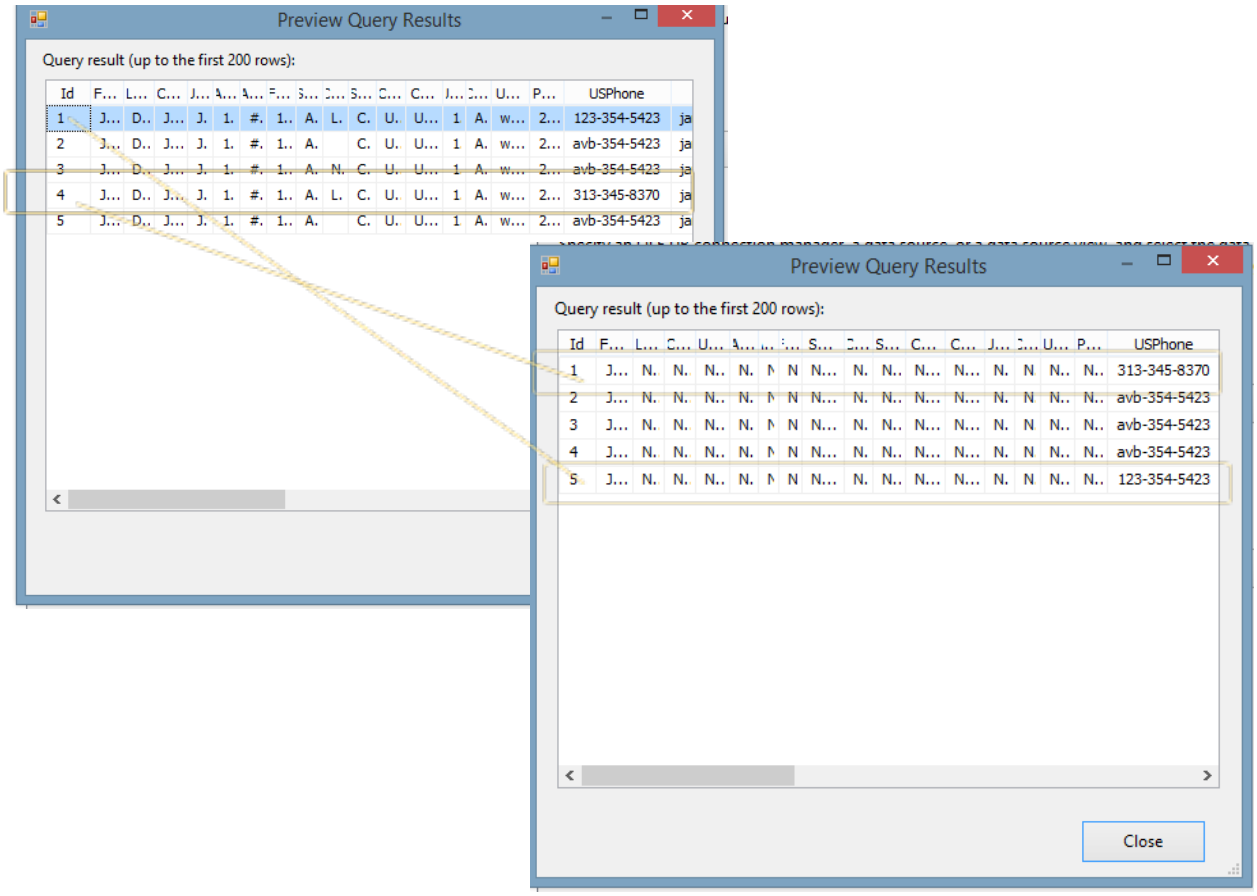
In Input and Output Properties tab, indicate which column you are willing to shuffle by setting ShuffleColumn Property to “True” value:



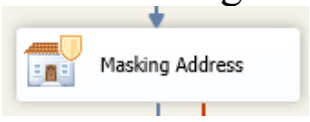
In the Destination Editor’s Mapping tab please map resulting masked set to the column of your choice:



Run the resulting package.
You will see that the initial element's column set is shuffled along the primary key, as in the below example, where the US Phone number values got shuffled, the first one became the fifth and the forth one became the first:

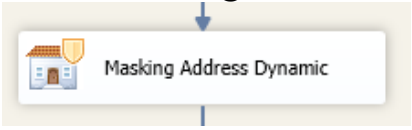


4. Masking Address



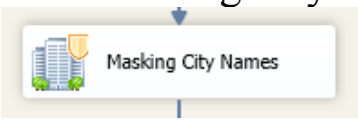
The Masking Address component takes street address as an input and produces a street address as an output in form of <Street Number> <Street Name> <Street Predicate>. It does not guarantee uniqueness of the addresses that are produced as an output but uses more than a million combinations in creating addresses so that the statistical dispersion is high.

5. Masking Address Dynamic



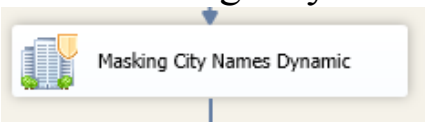
The Masking Address Dynamic component takes street address as an input and produces a street address as an output in form of <Street Number> <Street Name> <Street Predicate>. There is only a Dynamic component in this release. The component guarantees consistent repeated transformation of values. It does not guarantee uniqueness of the addresses that are produced as an output but uses more than a million combinations in creating addresses so the statistical dispersion is high.

6. Masking City Names



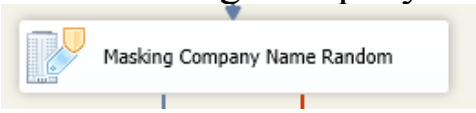
The Masking City Names component changes the city name element's value into another random city name chosen from a list that includes the names of America's 1000 largest cities. The component choses random value but does not guarantee consistent transformation of values but does not guarantee uniqueness.

7. Masking City Names Dynamic

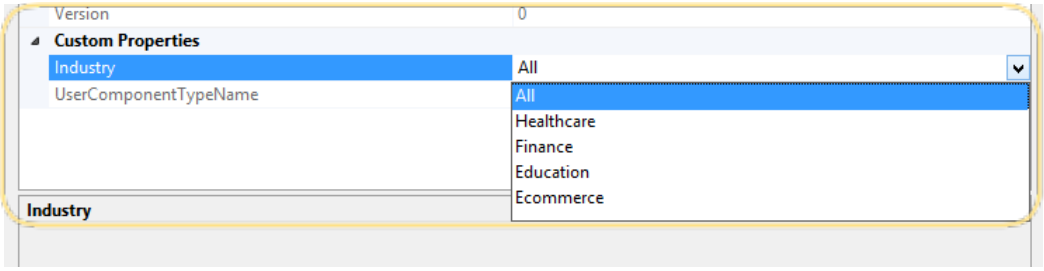


The Masking City Names Dynamic component changes The City name into another city name chosen from a list that includes the names of America's 1000 largest cities. The component guarantees consistent transformation of values but does not guarantee uniqueness.

8. Masking Company Name



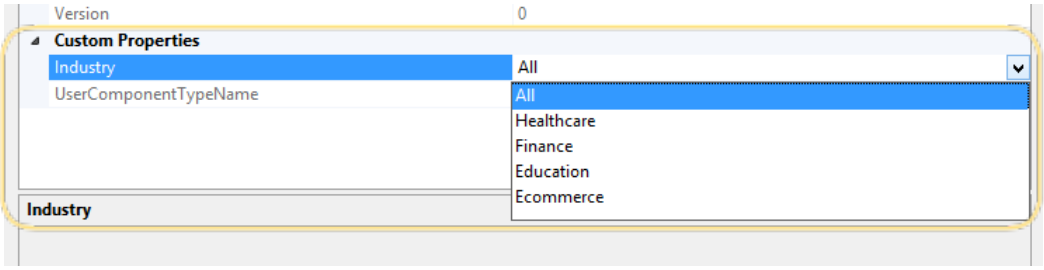
The Masking Company Names component changes the company name element value into another random company name chosen from a list of more than 4000 American companies. There are companies from four major industries, healthcare companies, financial companies, educational institutions and e-commerce properties. The component's custom property allows choosing one of the industries or all:



The matching value will be chosen from the indicated industry. The component provides random value and does not guarantee consistent transformation.

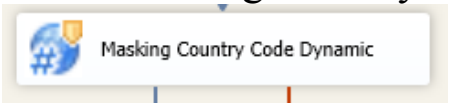
9. Masking Company Name Dynamic

The Masking Company Name Dynamic component changes the company name element value into another random company name chosen from a list of more than 4000 American companies. There are companies from four major industries, namely 732 healthcare companies, 847 financial companies, 1221 educational institutions, and 1290 e-commerce properties. The component's custom property allows choosing one of the industries or all:



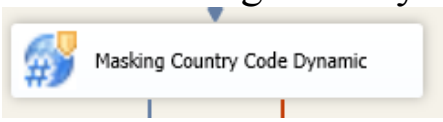
The matching value will be chosen from the indicated industry. The component guarantees consistent transformation of values but does not guarantee uniqueness.

10. Masking Country Code



Masking Country Code Component allows masking of all three types of country codes as per ISO 3166, whether they are two letters, three letters or digital code. It expects valid country codes on entrance and produces mapping on exit. It decides which ISO to use based on the original value – so you can have mixed country code types in the source.

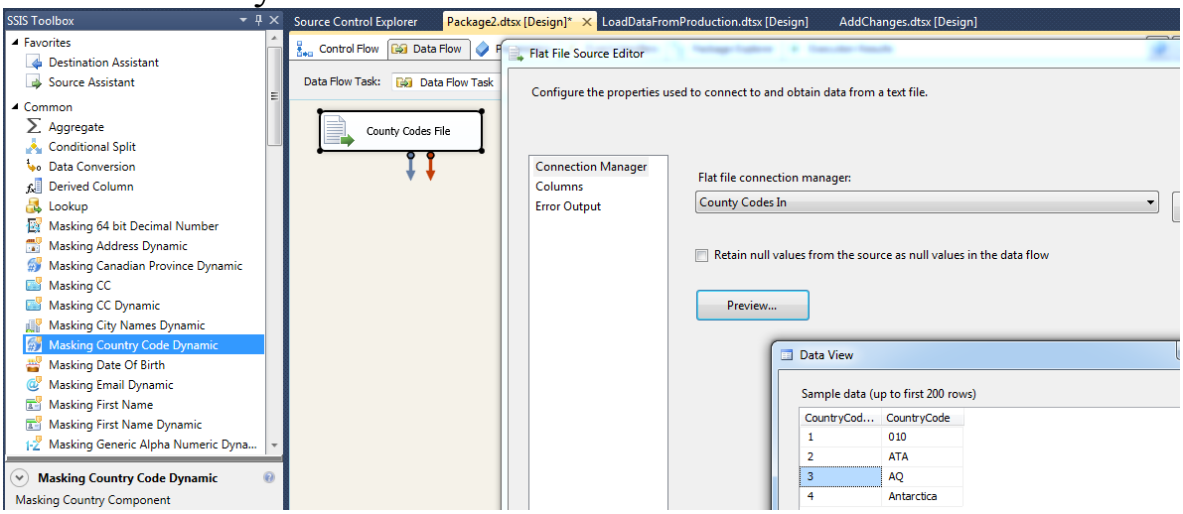
11. Masking Country Code Dynamic



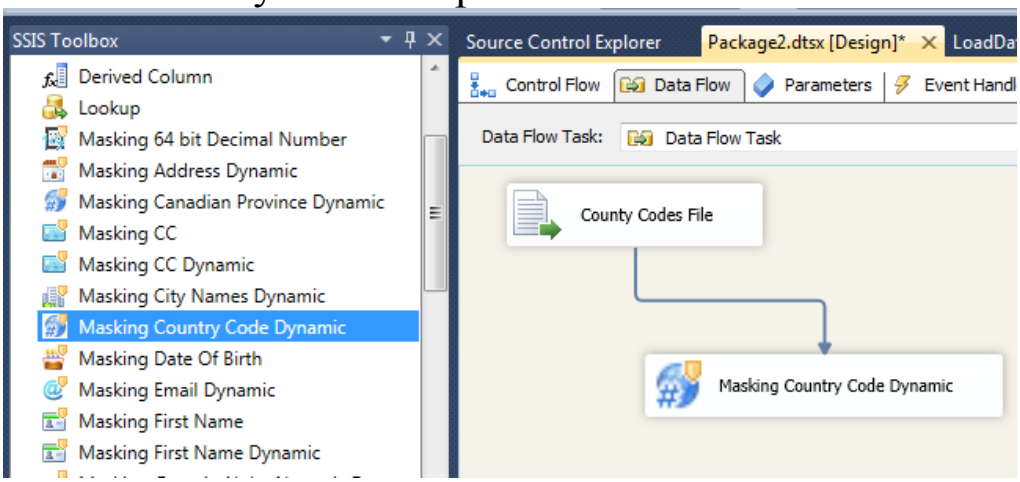
Masking Country Code Dynamic Component allows masking of all three types of country codes as per ISO 3166, whether they are two letters, three letters or digital code. It expects valid country codes on entrance and produces consistent mapping on exit. It decides which ISO to use based on the original value – so you can have mixed country code types in the source.

Usage Instructions:

Configure country code source to contain a country code column. The data should include two-letter, three-letter, or three-digit country code. Country name is also a possible entry value, if confirmed by one of the ISO 3166-1.

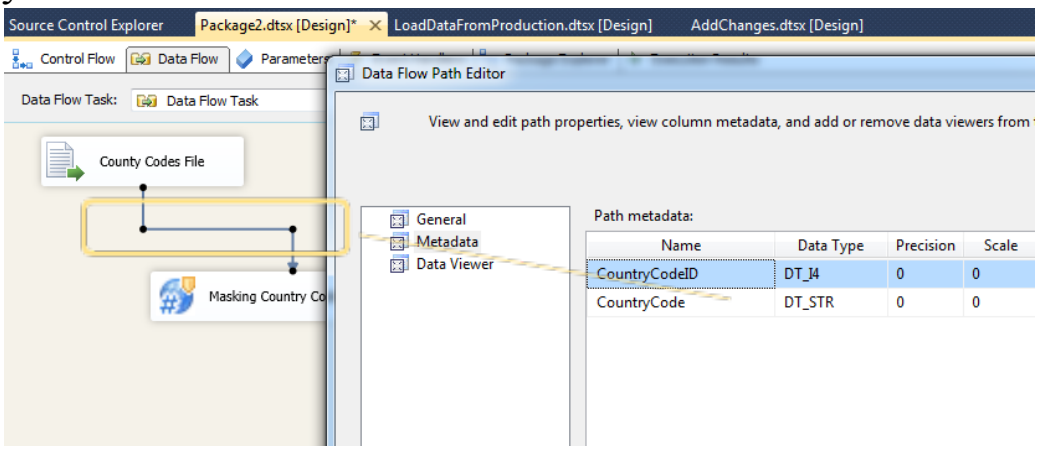


Drag and Drop Country Code Component, connect the source and the country code component:

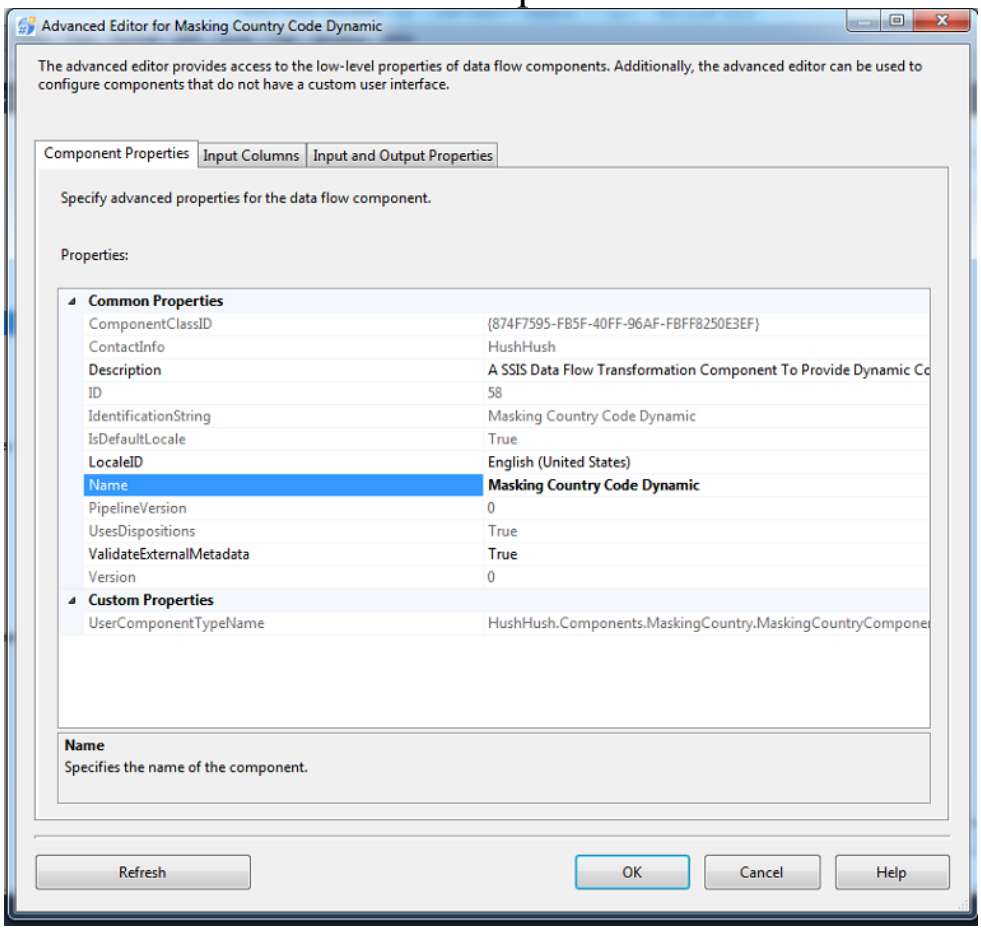


Now, the precedence constraint (the blue arrow) passes proper meta-data for the country code component. If you click on it,

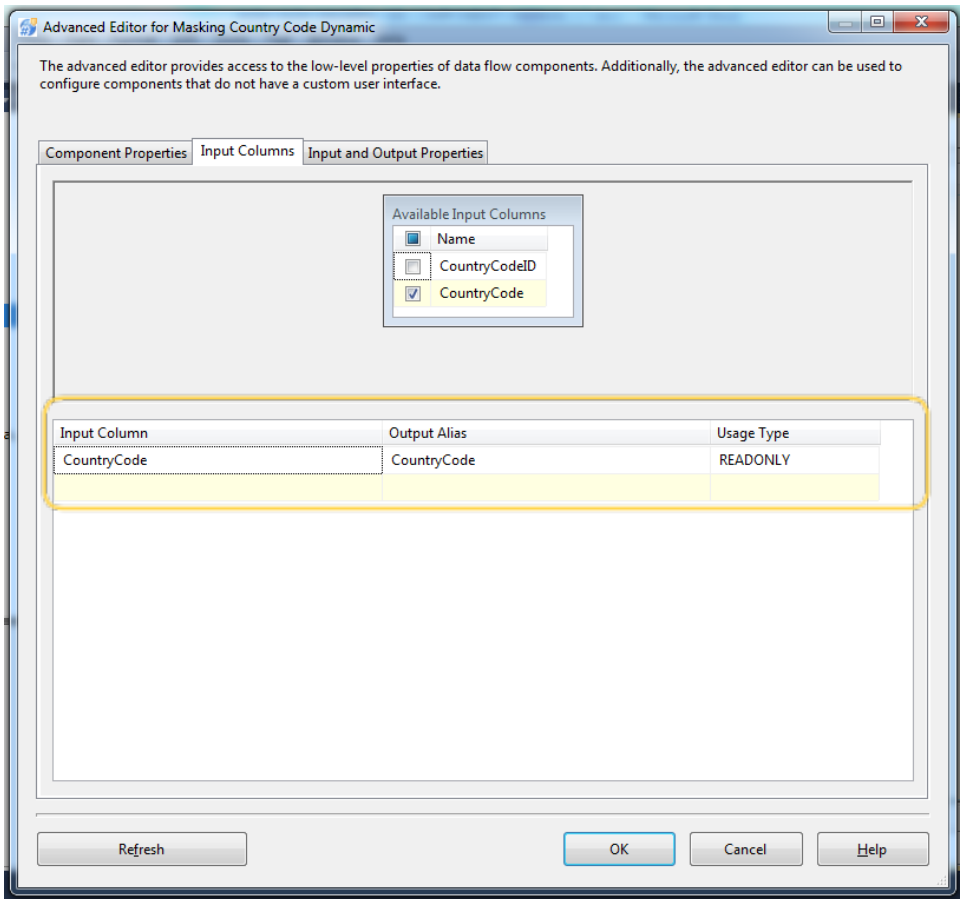
you will see:



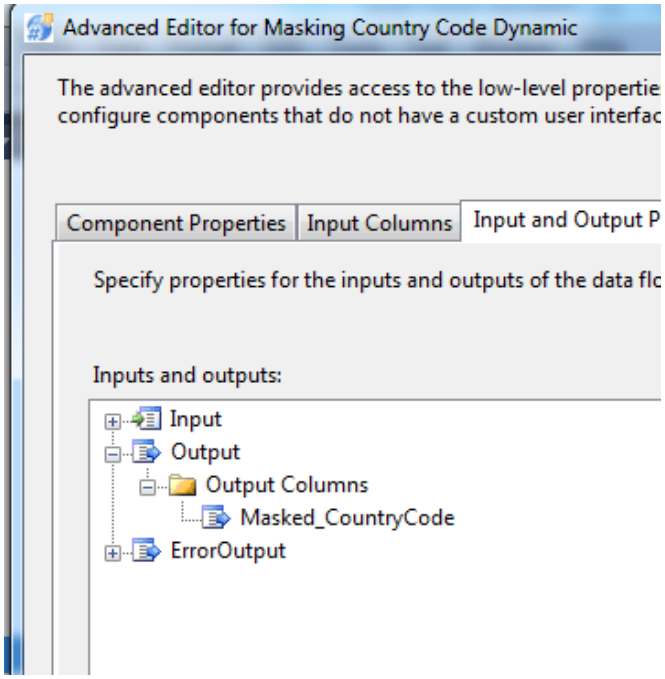
Now that the metadata for the country code exists, and values are passed into the masking component, please open the component editor, either by double clicking component or by right –clicking in the general component area and by choosing "Edit" or "Advanced Edit" Option:



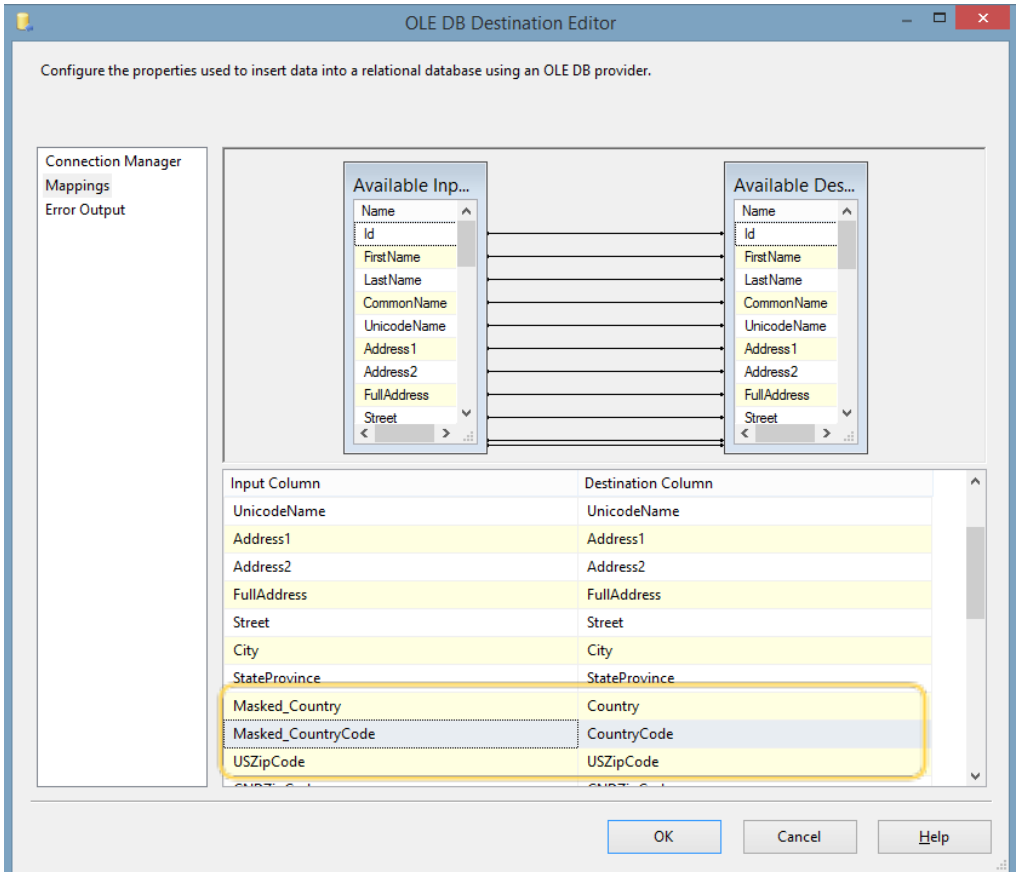
In the second tab, there are input columns. Please check-mark only one column, the Country Code:



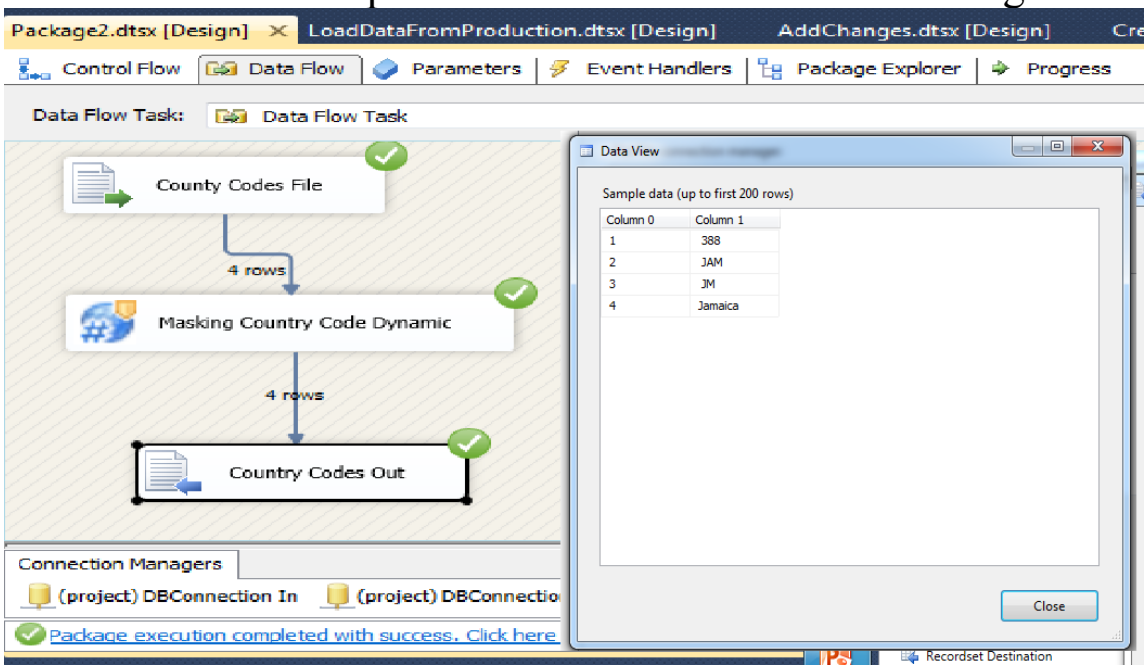
This will create an extra column with the prefix “Masked_”.



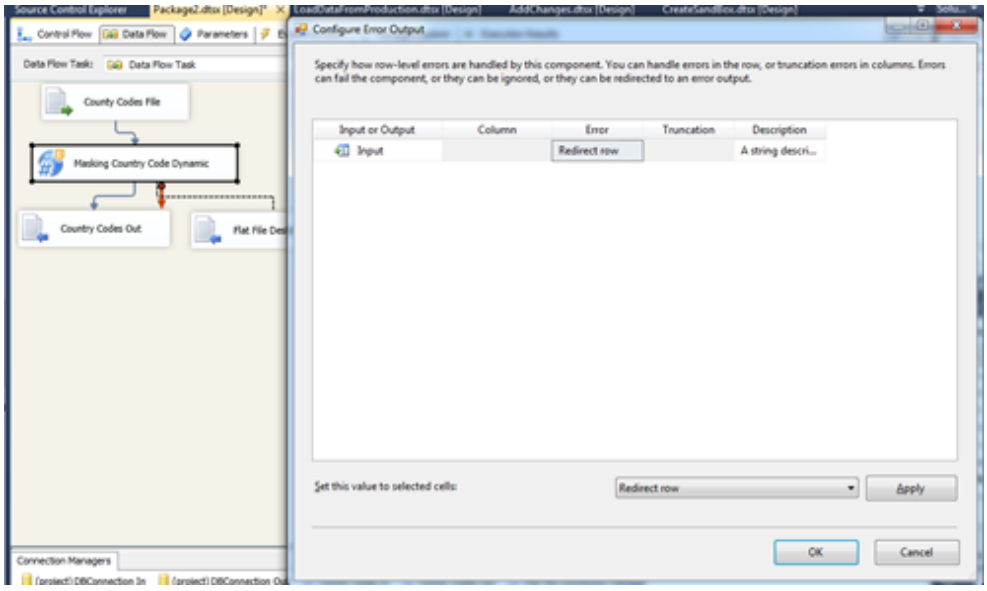
Create a connection manager for the destination and configure source component for the destination. In it in the tab “Mappings” specify that you want newly created CountryCode_Masked to be a field replacing the original value. For that, just click on the available input columns, chose the masked value and map to the “Available Destination Columns”



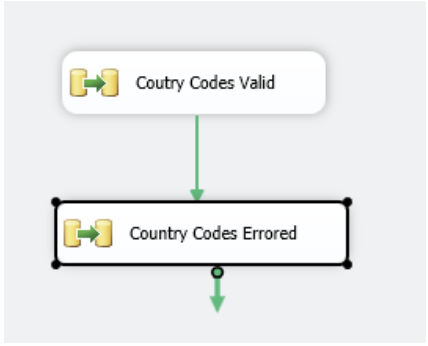
Now, all the configurations are complete for the valid values. You can run the component and see the results of masking:



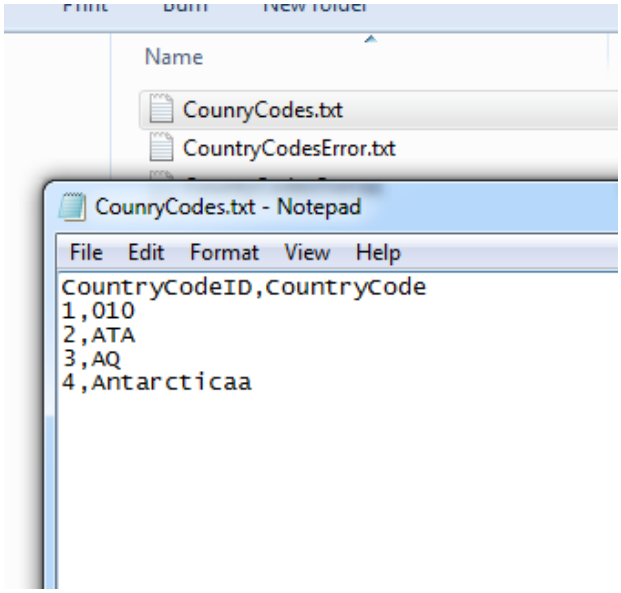
If, however, there are invalid values in the package, one would need to configure error handling. For that, each data masking component has error handling precedence constraint. One needs to create error connection and connect red arrow (error handling constraint) with this destination. As the connection is made, one needs to configure the state: “Fail”, “Ignore”, or “Redirect”.



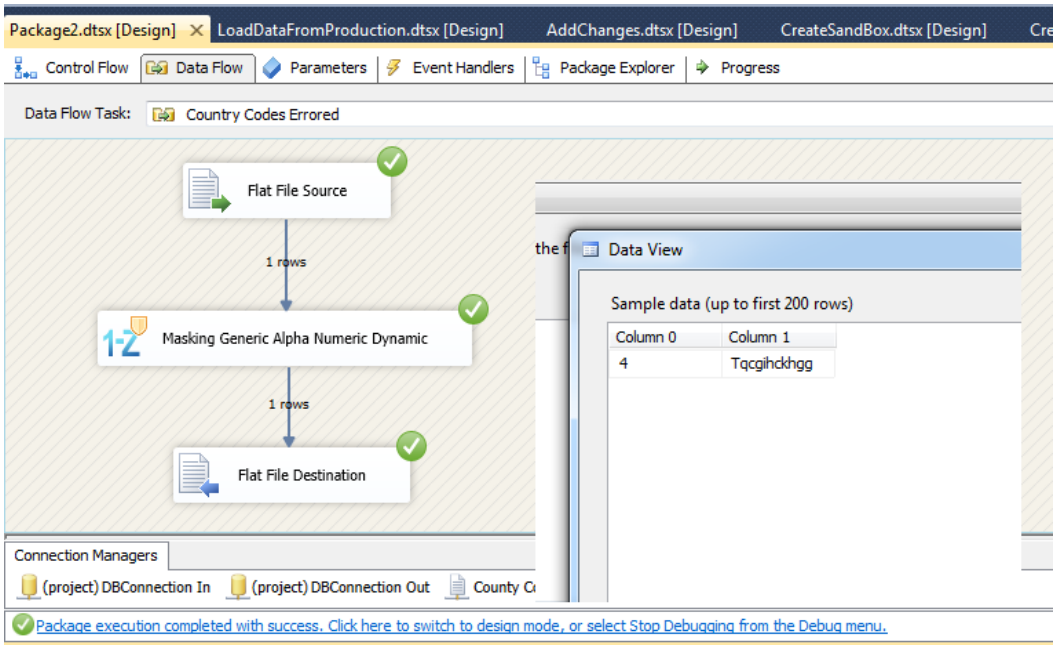
It is recommended that one re-directs the output into the error destination, so that later one be able to analyze and process data for quality purposes. After the re-direct is done, one still can mask the data with the Generic Alpha Numeric component.



The value in the original file is erroneous in spelling" Antarctica**a**" vs. "Antarctica" and it results in error, in the error destination file:

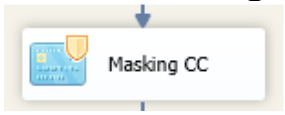


Generic Alpha Numeric component might be used to mask errored data – or if one choses, one simply store errored data for quality inspection purposes and not mask it at all.



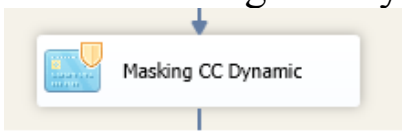
The rest of the components are configured in a similar fashion.

12. Masking CC (credit card)



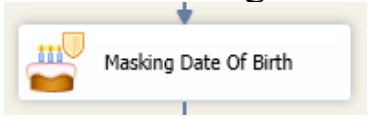
The Masking Credit Card component expects a credit card number in a string that consists of integers, e.g.: XXXXXXXXXXXXXXX. There might be separators in between the digits, such as "-"or "/". It produces a card of the same issuer with the parity digit consistent with Luhn calculation.

13. Masking CC Dynamic (credit card, dynamic)



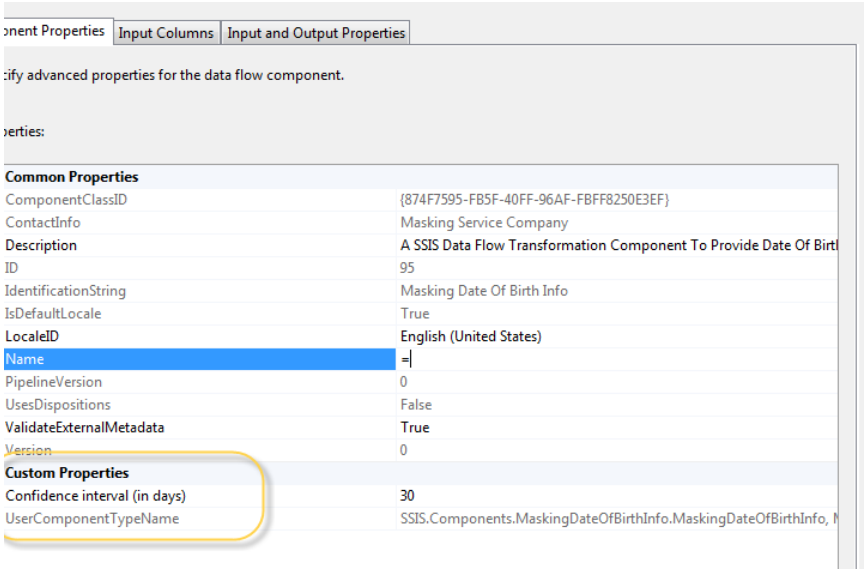
The Masking Dynamic Credit Card component ("Masking CC Dynamic Info") is transforming credit card numbers consistenly, with repeated results, using FPE –like algorithms, while retaining the issuer and Luhn.

14. Masking Date of Birth

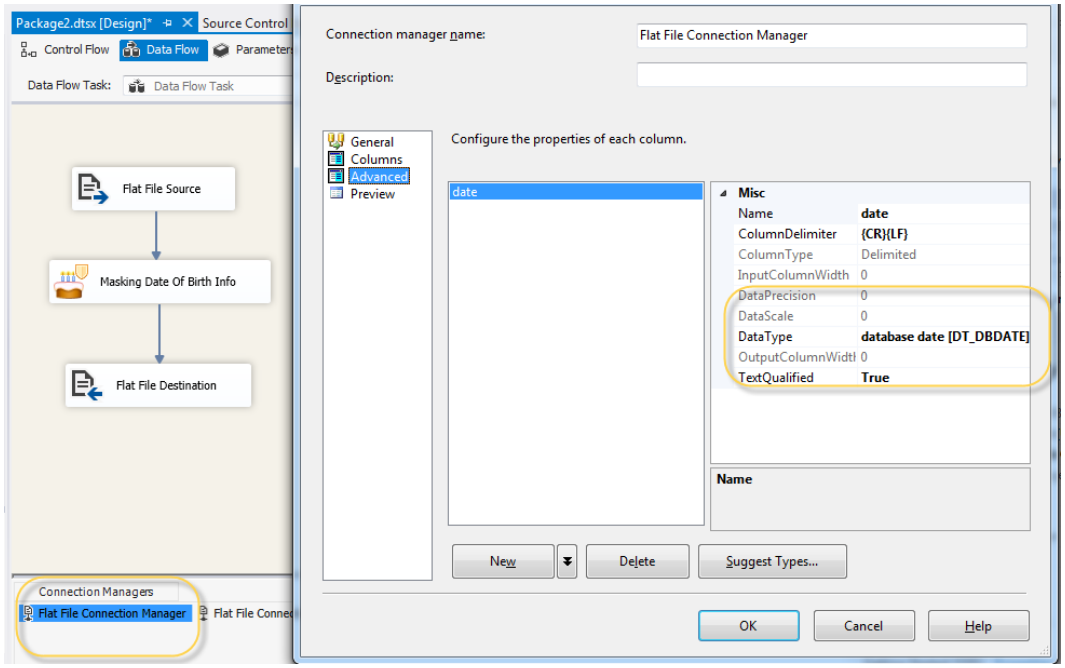


The Masking Date of Birth component provides generic date handling. It expects a string in the date format ('MM/DD/YYYY) and can be used to mask any date. However, one has to be aware that per HIPAA "Safe Harbor" requirements dates of birth that exceed 89 years age limit are going to output a

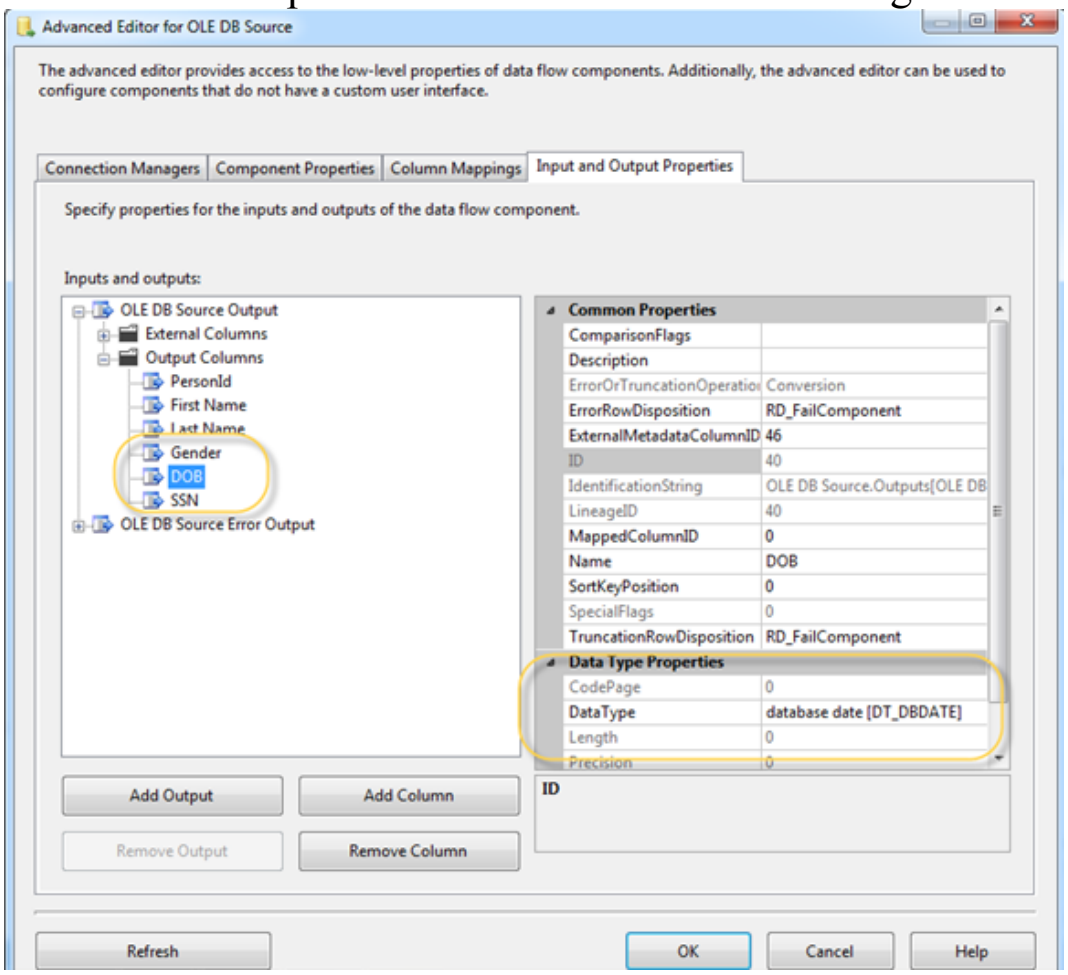
date that does not allow discovery of age. The component also allows selection of the interval of confidence in days that defines interval of variation for the date. The date itself is in the middle, interval values add both positive and negative numbers of days to the date. Example: if May, 1st 2000 is the date, defining an interval of confidence as two days will produce April 29th, April 30th, May 1st, May 2nd, and May 3rd as values.



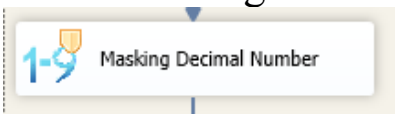
Masking Date of Birth was changed to a DT_DBDate data type. Prior versions had this component as a string and the type conversions have been cumbersome. Currently, to make component work with any other types please configure your connection manager's metadata to configure to DT_DBDate. Here is an example of Flat File Connection Manager:



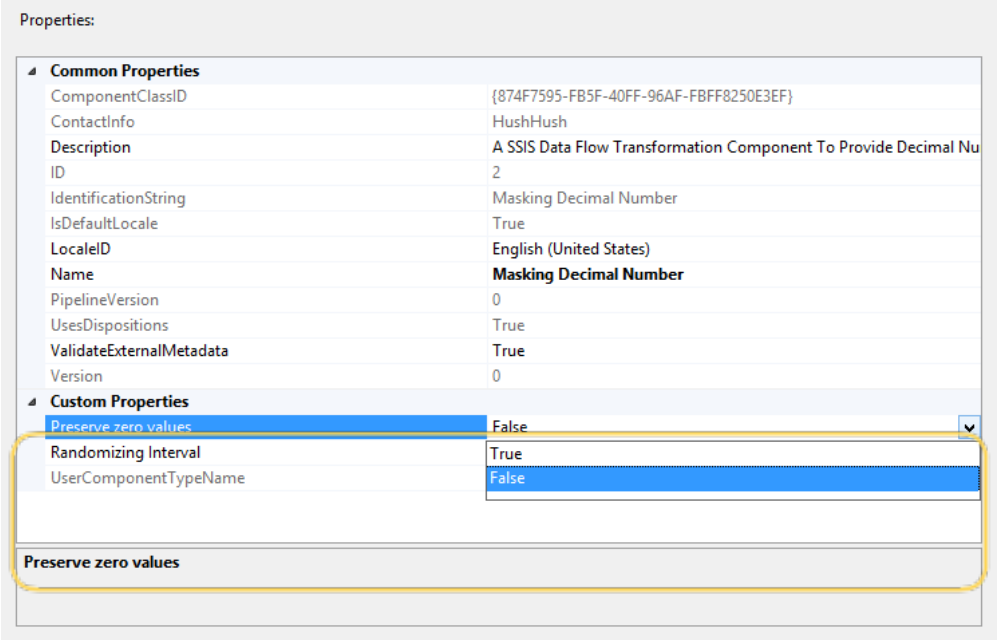
Here is an example of OLE DB connection's configuration:



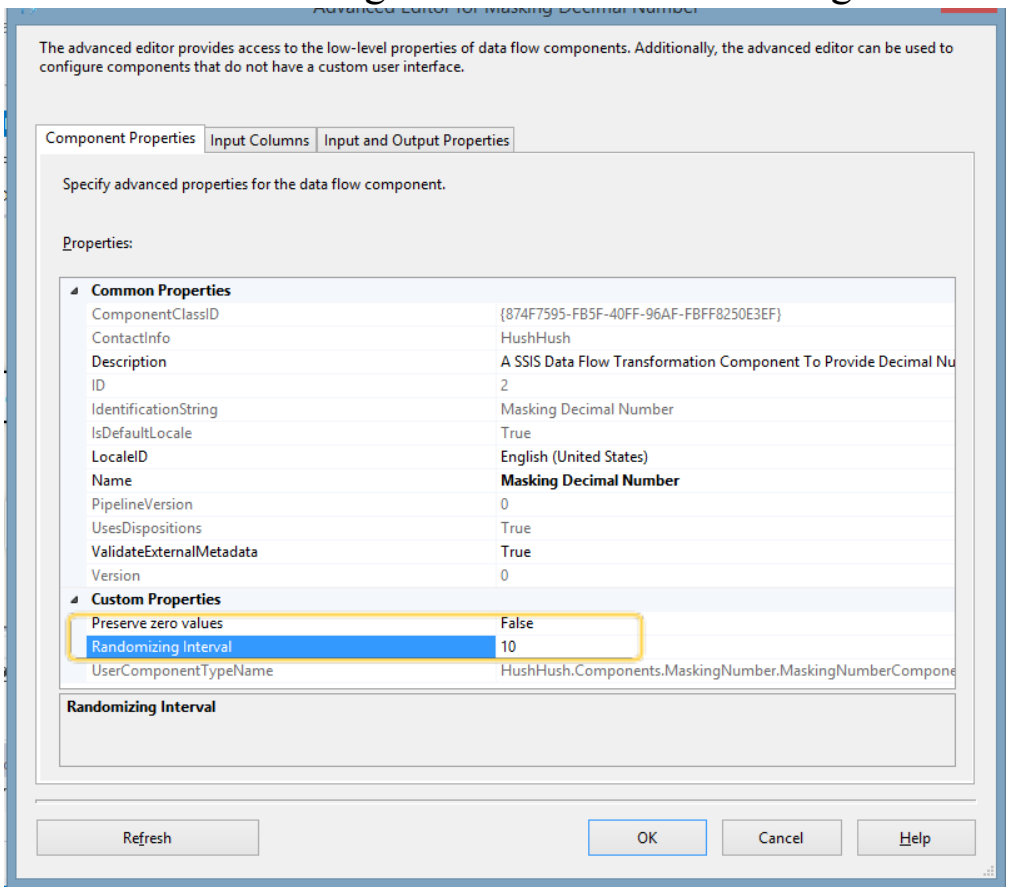
15. Masking Decimal Number



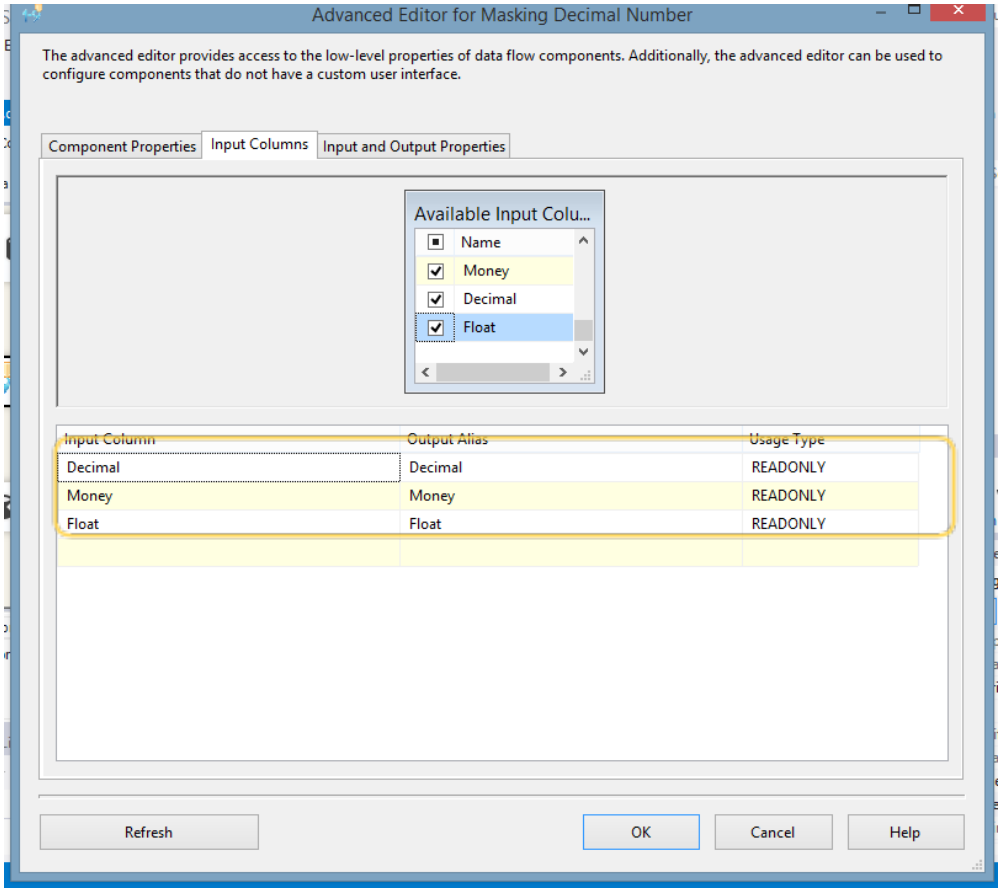
Masking Decimal Number is a component that takes any numeric type value, such as float, integer, decimal number and money as an input and generates a numeric value of the same type within an integer interval of confidence of the one supplied. For example, if you supply 29.00 and you supply the interval of confidence of 10, you will be adding/subtracting 10 out of 29.00 and created values will fall within 19.00 and 39.00
The component has two custom properties: it allows to either preserve zero values or skip them:



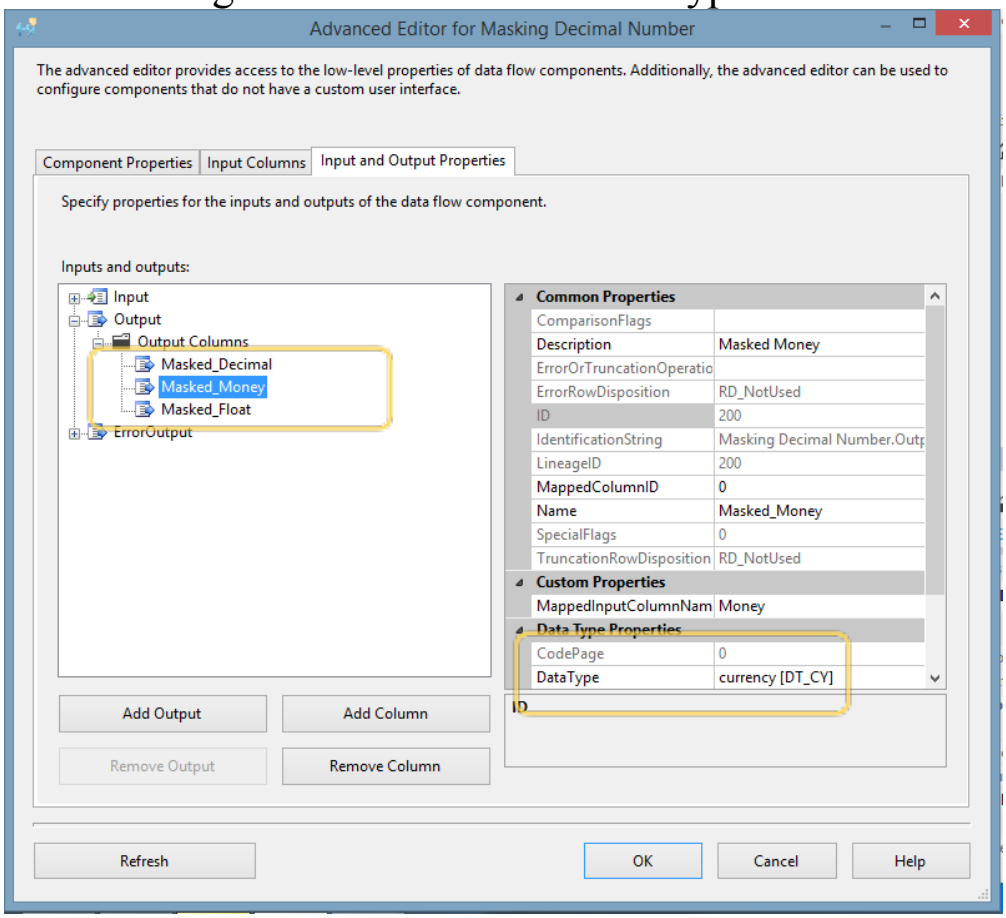
and it allows to change a value of a randomizing interval:



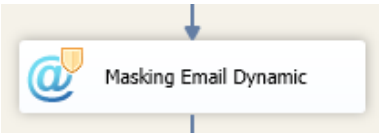
The practitioner can mask more than one value, of multiple numeric data types at a time:



The resulting values contain the same type masked values:

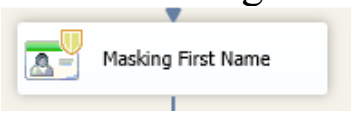


16. Masking Email Dynamic



The Masking Email Dynamic component expects data in xxxx@xxxx.xxx format. It provides output in the same format, with .com domain. It provides consistency in the output, meaning given the same input, it will provide the same output. However, collisions are possible although highly unlikely. While it is not difficult to create unique emails using names and sequence numbers with the derived component's string functions, the Email component provides a quick and dirty way to create consistent masked emails.

17. Masking First Name

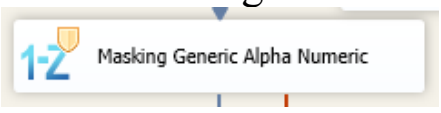


The Masking First Name component expects First Name as an input and produces First Name as an output.

18. Masking First Name Dynamic

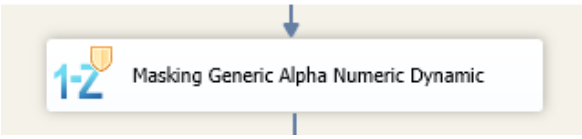
The First Name Dynamic input will produce consistent First Names, if the name repeats, the output will repeat the same name in accordance with original mapping. It does not provide uniqueness.

19. Masking Generic Alpha Numeric



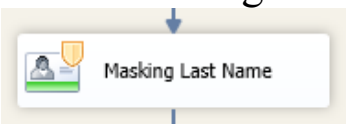
Generic Alpha Numeric component replaces letters with letters, maintains capitalization, and replaces numbers with numbers. It retains the non-alphanumeric characters, such as dashes, slashes, dots and comas, and keeps track of their position in the string. It currently only replaces characters from alphabets that uses English characters. It will replace strings based on letters only with letter strings, and strings based on numeric characters only with numeric characters.

20. Masking Generic Alpha Numeric Dynamic



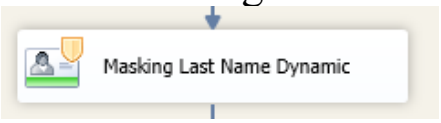
Generic Alpha Numeric Dynamic component replaces letters with letters, maintains capitalization, and replaces numbers with numbers. It retains the non-alphanumeric characters, such as dashes, slashes, dots and comas, and keeps track of their position in the string. It currently only replaces characters from alphabets that uses English characters. It will replace strings based on letters only with letter strings, and strings based on numeric characters only with numeric characters. The component masks values consistently and preserves uniqueness of the data set.

21. Masking Last Name



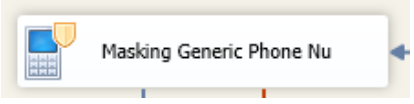
The Masking Last Name component expects Last Name as an input and produces Last Name as an output.

22. Masking Last Name Dynamic



The Last Name Dynamic component's output will produce consistent Last Names. If the name repeats, the output will repeat the same name in accordance with original mapping. It does not provide uniqueness in mapping. Many names can map into the same name.

23. Masking Generic Phone Number



The Masking Phone Number component is a random component that creates valid international phone number of the same format as given. The component allows a variety of separators as formatting, including "-","." and spaces. The resulting value will preserve the separating characters in their current position.

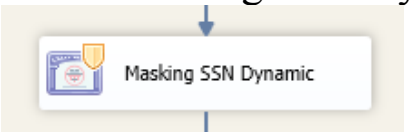
24. Masking Generic Phone Number Dynamic

The Masking Phone Number Dynamic component is a Dynamic component that expects valid international phone number starting with the international country code. It will process numbers both starting with plus (‘+’) and not. The component allows a variety of separators as formatting including “-“,”.” and spaces. The resulting value will preserve the separating characters in their current position. The country code prefix will remain unchanged.

25. Masking SSN (social security number)

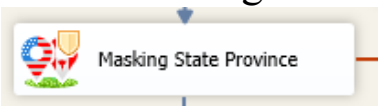
The SSN component creates a social security number. It generates only real social security numbers. There are test numbers that start with 000,666, 900-999 – these will be respectfully converted into test data. It expects input in 123-45-6789 format. The separators will be retained in their respective places after masking. Allows different separators, including ‘- ‘,’ . ‘,’ ‘,’/’ etc.

26. Masking SSN Dynamic (social security number, dynamic)



The SSN Dynamic component maintains all the features of the SSN component and can be used without creating a masking table. The component replaces values on the fly. The SSN Dynamic component recognizes that your current system indeed might have testing records starting with 000,666, 900-999 prefixes. If it encounters these prefixes, it keeps the prefix intact, masking the rest of the number. The separators will be retained in their respective places after masking. Allows different separators, including ‘- ‘,’.’,’ ‘,’/’ etc.

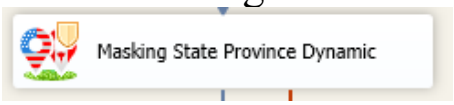
27. Masking State Province



Masking State Province is a component that takes US states or Canadian provinces two- letter abbreviations and maps them to other values. The mapping is random. Canadian provinces map

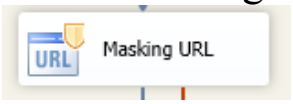
to Canadian provinces, and US states map to US states only. Territories map to territories.

28. Masking State Province Dynamic

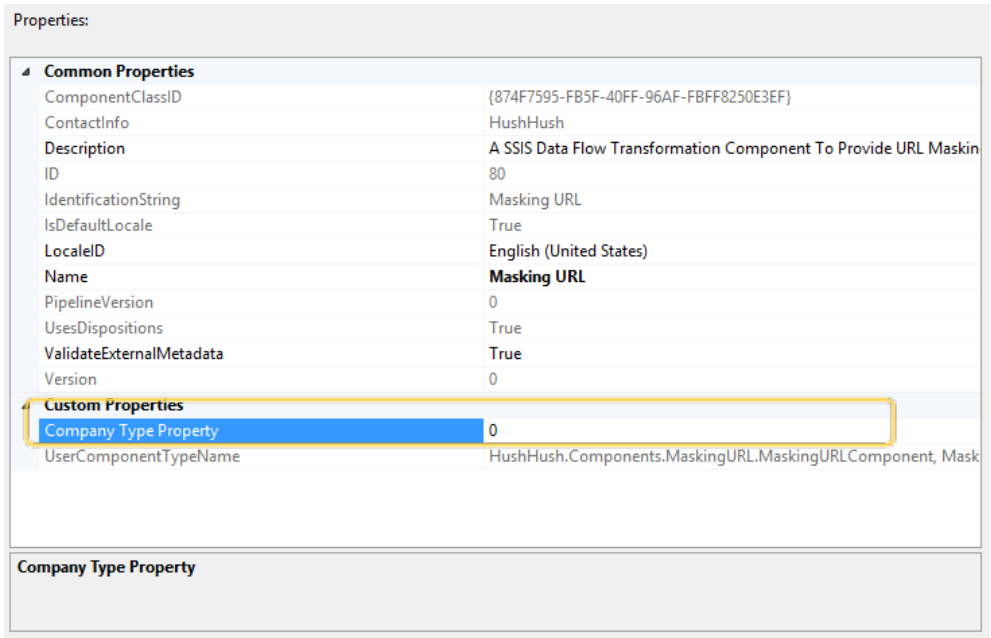


Masking State Province Dynamic is a component that takes US states or Canadian provinces two- letter abbreviations and maps them to other values. This is NOT a one-to-one mapping, it is many-to-one mapping, meaning that both CA and NV can map to OH. Canadian provinces map to Canadian provinces, and US states map to US states only. Territories map to territories.

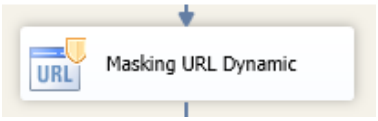
29. Masking URL



The Masking URL component expects a URL-like string upon input and provides a value for the URL upon output. There are three types of the URL, those for financial companies, health companies, and the mix of both. 0 stands for the mix, 1 indicates financial institutions, and 1 provides healthcare companies' URL.

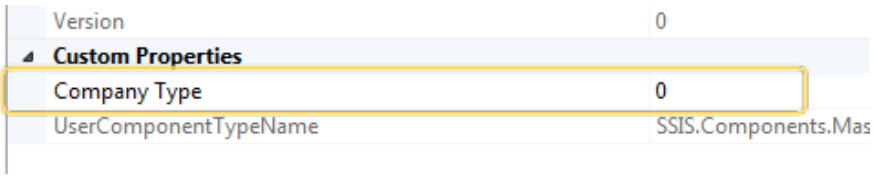


30. Masking URL Dynamic



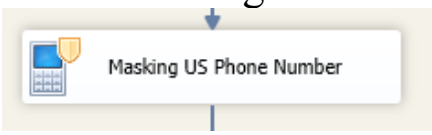
The Masking URL Dynamic component provides a choice for

financial institutions URLs, health institutions URLs, or mix. To choose the type of URLs, one has to put 1, 2, or 0 correspondingly in the custom property of company type field in the editor:



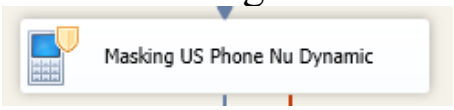
By default “Company Type” = 0 and presents a mix of URLs.

31. Masking US Phone Number



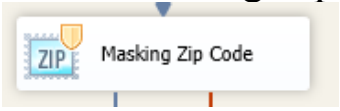
The Masking US Phone Number component expects the data to be in the format of xxx-xxx-xxxx where x is an integer. The output value retains the area code and masks the phone number. The random component does not preserve values’ repeatedness and consistency, providing random values upon input. It does not guarantee uniqueness.

32. Masking US Phone Number Dynamic



The Masking US Phone Number Dynamic component expects the data to be in the format of xxx-xxx-xxxx where x is an integer. The output value retains the area code and masks the phone number. The algorithm allows the practitioner to retain consistency and uniqueness of the element’s values upon output.

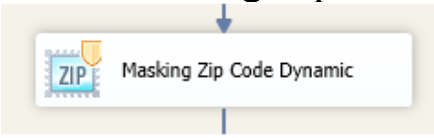
33. Masking Zip Code



The Masking Zip Code creates a random zip code that uses the three digit prefixes of the currently active US zip codes and changes Canadian postal codes provided on input to other Canadian postal codes values at the output. The component expects a string consisting of digits only for US postal codes and

"A1A A1A' format for the Canadian Postal codes. Any other format will result in failure.

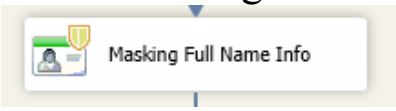
34. Masking Zip Code Dynamic



The Masking Zip Code Dynamic component consistently changes United States zip codes to the zip codes with three digits prefixes that are currently in use and changes Canadian postal codes provided on input to other Canadian postal codes values at the output. The component expects a string consisting of digits only for US postal codes and "A1A A1A' format for the Canadian Postal codes. Any other format will result in failure. If a practitioner wants to mask a zip or address code containing alphanumeric combination other than the ones in Zip code dictionary, s/he could use Generic Alphanumeric component. Zip Code Dynamic component masks data for US zip code with statistically limited population in consideration of 18 rules of "Safe Harbor".

INTRODUCED IN THIS RELEASE:

35. Masking Full Name Components (random/dynamic)



The component first parses the data. In parsing, it assumes the following:

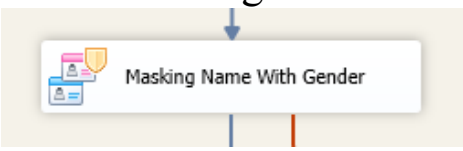
- It assumes that if there is only one word in the field, it is a First Name. It does not check against the dictionaries of names as such operation would slow execution down. Besides, there might be name values that are not currently in the dictionary. American parents invent new names every year.
- If there are two words in the field, the component's logic treats them as the First Name followed by the Last Name.
- If there are more than two words in the field, and no commas, the first word is treated as the First Name, the last is

treated as the Last name, and the second word is treated as the Middle name. If the second word is a one-letter initial, it is treated as such initial.

- If there are more than two words and there is a comma after the first word, the first word is treated as the last name, the word after the comma is treated as the first name, and the next one is treated as the middle name and/or initial.
- If there are quotes around the word in the field, they are disregarded.
- If there is a dash without spaces, like Ivan-Lee, the word is treated as one word.
- The assumed values, based on the positions, are run against the dictionaries as they would in the First Name(random/dynamic) and in the Last Name(random/dynamic) and are concatenated upon completion of the substitution algorithm. Thus, your four word entry may become 3 word entry, etc.
- Spaces and nulls retain values.
- In general, only the following formats of the full name are currently considered (Fn – first name, Ln – last name, MI – middle initial):
  - Fn
  - Fn Ln
  - Fn Fn Ln
  - Fn MI Ln
  - Ln, Fn
  - Ln, Fn MI
  - Ln, Fn Fn

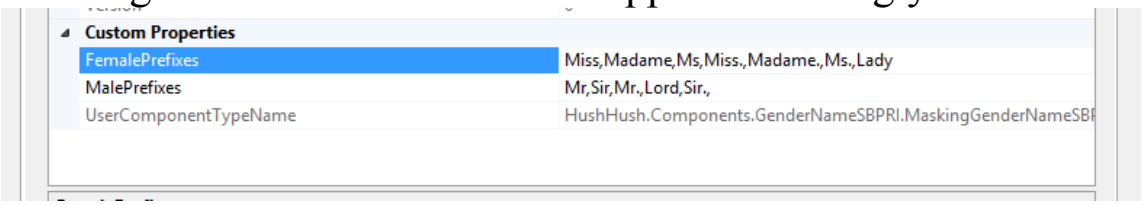
- The rest of the possible combinations of values within the full name element value are driven to these formats
- There might be discrepancies after masking, if along with the Full Name, data is also normalized as First Name and Last Name, with the Masked First Names and Masked Last Names, but if data in the Full Name has been normalized to the above formats, it will provides majority of values proper and will retain referential integrity of the combined values.
- The error output works as with the rest of the components, and should be used with GAN (generic alpha numeric) to mask erroneous entries or the values should be stored for the later examination.

36. Masking Name with Gender

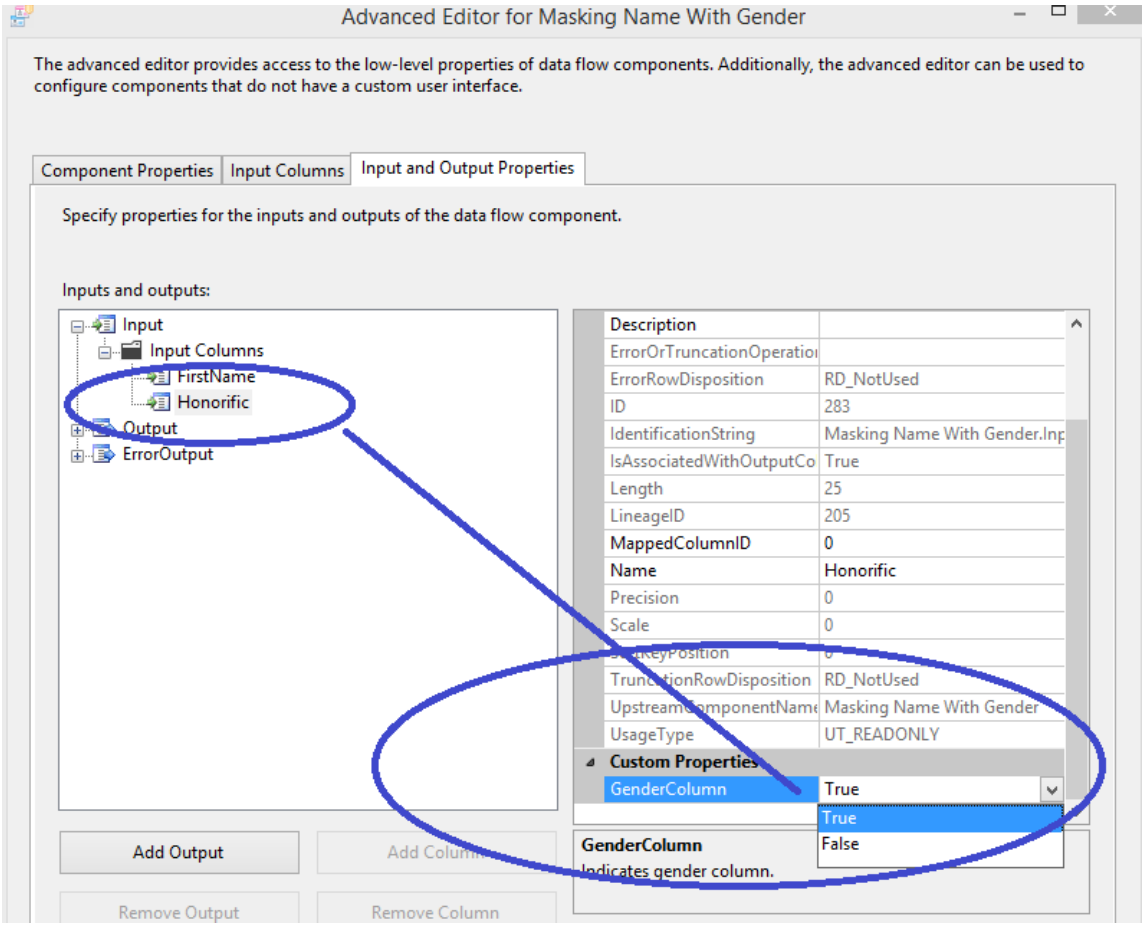


The Masking First Name component expects First Name as an input and produces First Name as an output. It also identifies whether the name is Female, Male, or Unisex and maps names according to existing popular name mappings. If however the name is not among those that have component's mapping, it will map into the unisex name.

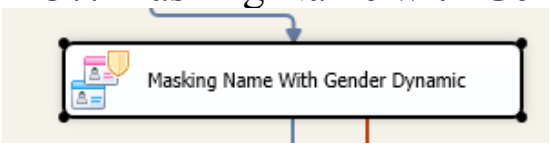
There is an option that will override default behavior and allow one to be more precise in gender definition. One has an option of providing a list of the possible gender identifiers, be it values of the gender (male/female, m/f) or honorifics with the gender meaning – and the name will be mapped accordingly:



One has to make sure to introduce the values in the Female Prefixes and Male Prefixes Custom property, and indicate which column contains the gender definition in the source:



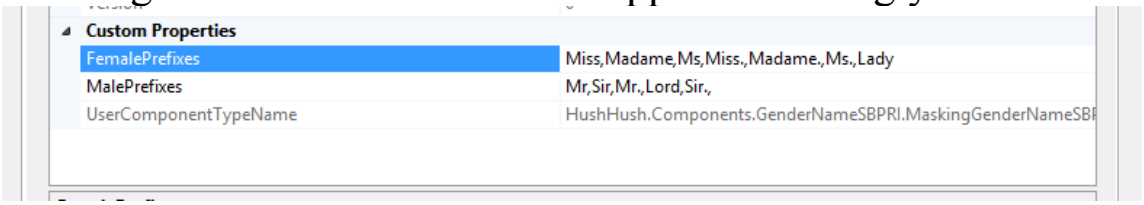
37. Masking Name with Gender Dynamic



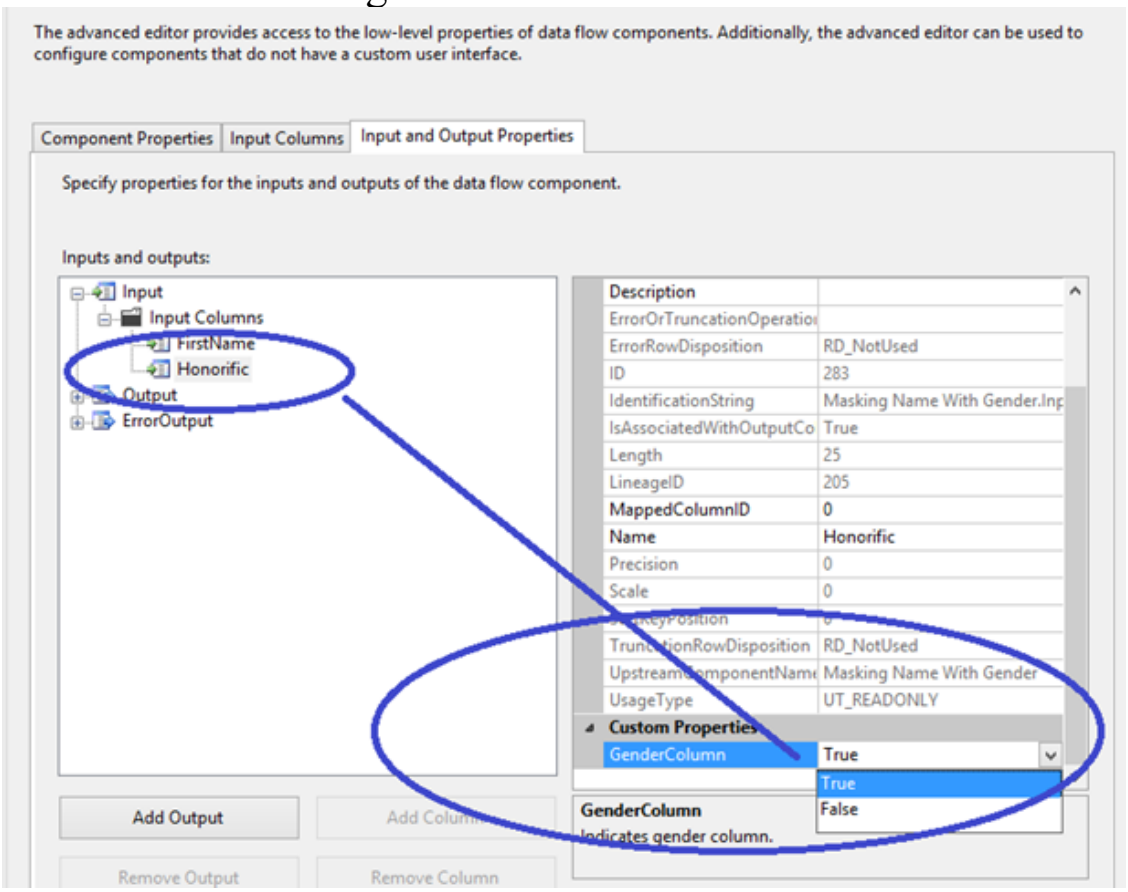
The Masking First Name with Gender Dynamic component expects First Name as an input and produces First Name as an output. The output is “deterministic” – meaning it always provides the same output on the same input and as such helps maintain referential integrity enterprise wide.

It also identifies whether the name is Female, Male, or Unisex and maps names according to existing popular name mappings. If however the name is not among those that have component’s mapping, it will map into the unisex name.

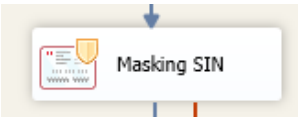
There is an option that will override default behavior and allow one to be more precise in gender definition. One has an option of providing a list of the possible gender identifiers, be it values of the gender (male/female, m/f) or honorifics with the gender meaning – and the name will be mapped accordingly:



One has to make sure to introduce the values in the Female Prefixes and Male Prefixes Custom property, and indicate which column contains the gender definition in the source:



38. Masking SIN (Canadian Social Insurance Number)



The Masking SIN data masking component creates a Canadian Social Insurance Number. It generates only real Canadian Social Insurance Numbers. It maintains the triad format and calculates Luhn for verification of the last digit**.**

39. Masking SIN Dynamic (Canadian Social Insurance Number)

The Masking SIN data masking component masks a Canadian Social Insurance Number in deterministic fashion The same number on input always produces the same number on the output and allows to maintain referential integrity enterprise wide. It generates only real Canadian Social Insurance Numbers. It maintains the triad format and calculates Luhn for verification

of the last digit**.**

COMING SOON

More components for masking passports, IPs,VIM, and bank account numbers.
CUSTOM COMPONENTS
Please contact us if there is a custom component that you need **in your particular situation!**

TABLE OF COMPONENTS AND ALGORITHMS

component	*random*	*dynamic*	*date and number variance*
SSN	X	X	
Credit Card	X	X	
First Name	X	X	
Last Name	X	X	
Address	X	X	
City Names	X	X	
US Phone Number	X	X	
Email	X		
ZIP	X	X	
Date of Birth			X
URL	X	X	
Generic Alpha Numeric	X	X	
Phone Number	X	X	
Country Codes	X	X	
Decimal Number			X
State Province	X	X	
Dictionary Load - generic	X		
Name with Gender	x	x	
SIN	x	x	
Company Name	X	X	
Full Name	X	X	
Shuffle – Generic			
Substitution - Generic			