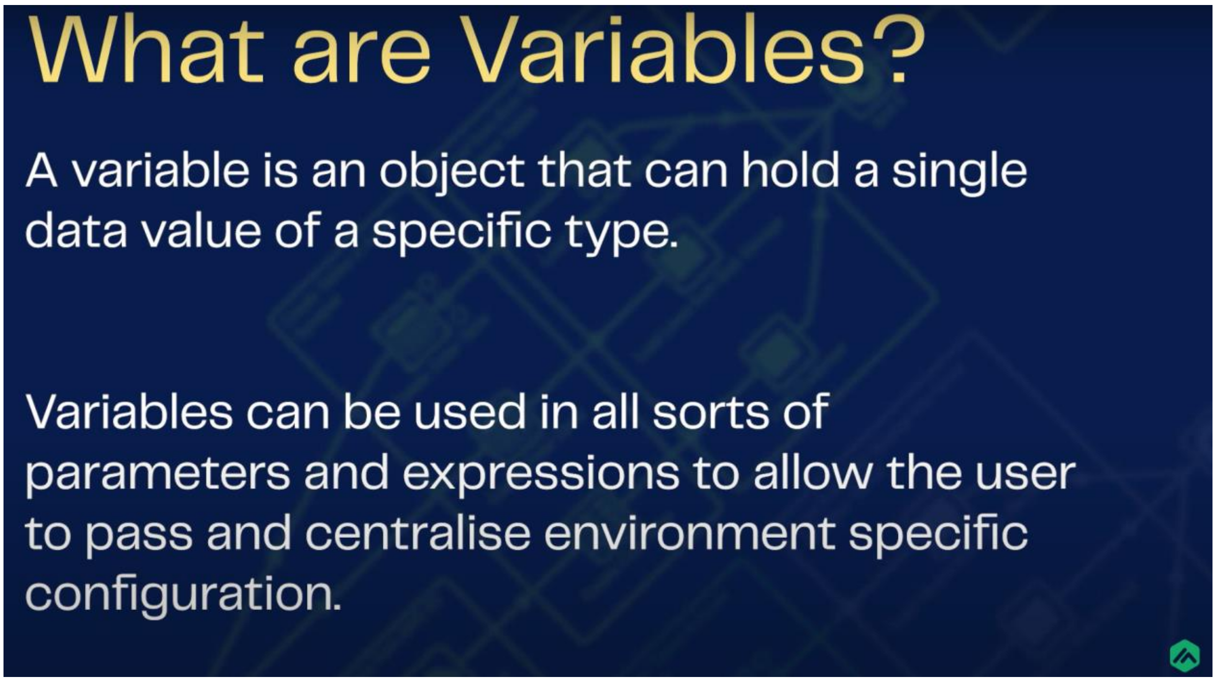
VARIABLES IN MATILLION

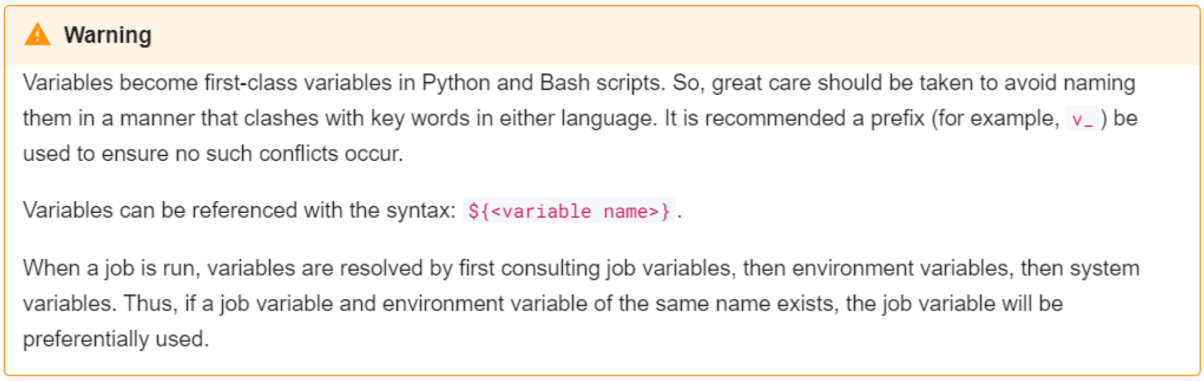


What are Variables?

A variable is an object that can hold a single data value of a specific type.

Variables can be used in all sorts of parameters and expressions to allow the user to pass and centralise environment specific configuration.

Matillion ETL supports Environment Variables, Automatic Variables, Job Variables, and Grid Variables.

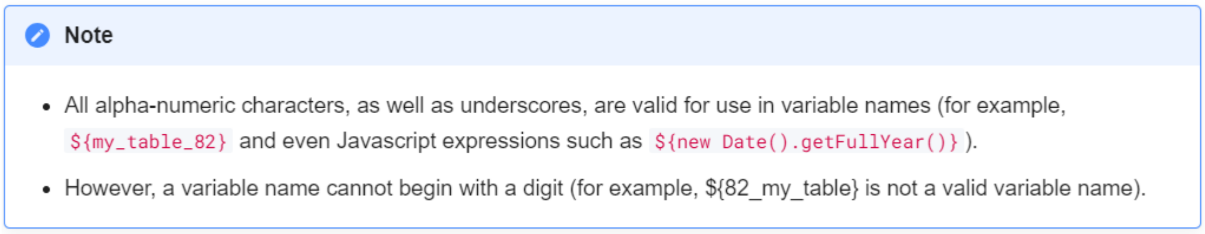


Warning

Variables become first-class variables in Python and Bash scripts. So, great care should be taken to avoid naming them in a manner that clashes with key words in either language. It is recommended a prefix (for example, `v_`) be used to ensure no such conflicts occur.

Variables can be referenced with the syntax: `${<variable name>}`.

When a job is run, variables are resolved by first consulting job variables, then environment variables, then system variables. Thus, if a job variable and environment variable of the same name exists, the job variable will be preferentially used.



Note

- All alpha-numeric characters, as well as underscores, are valid for use in variable names (for example, `${my_table_82}` and even Javascript expressions such as `${new Date().getFullYear()}`).
- However, a variable name cannot begin with a digit (for example, ${82_my_table} is not a valid variable name).

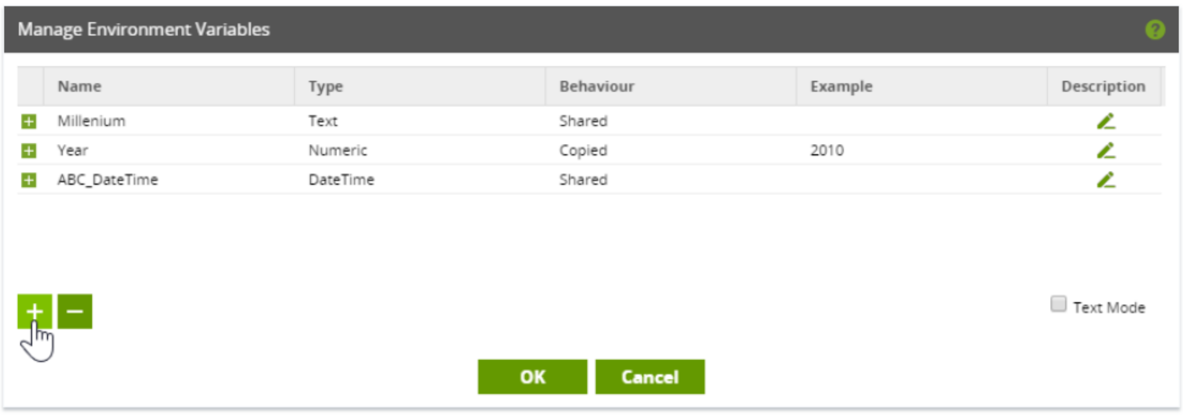
Behaviour (copied and shared)

Throughout this article, variables may be referred to as **Copied** and **Shared**. This refers to their "branch behaviour" inside a job. A "branch" in this context means a divergence of connectors within a job, giving a branched structure. Iterator components are also examples of branching, albeit with a different aesthetic. The branch behaviour describes how a variable behaves when updated during a job run:

- **Copied** variables can be updated within one branch of a job without updating variables of the same name in other branches. A branch using a copied variable will begin by taking its default value, even if this variable has been updated in another, parallel branch.
- Conversely, **Shared** variables are updated in a job-wide fashion. As such, regardless of branches, if a Shared variable is updated in one branch, all other branches will use that updated value.

Note

In previous versions of Matillion ETL, **Copied** and **Shared** variables have been referred to as **Local** and **Global**, respectively, and can be thought of synonymously when reading the documentation.

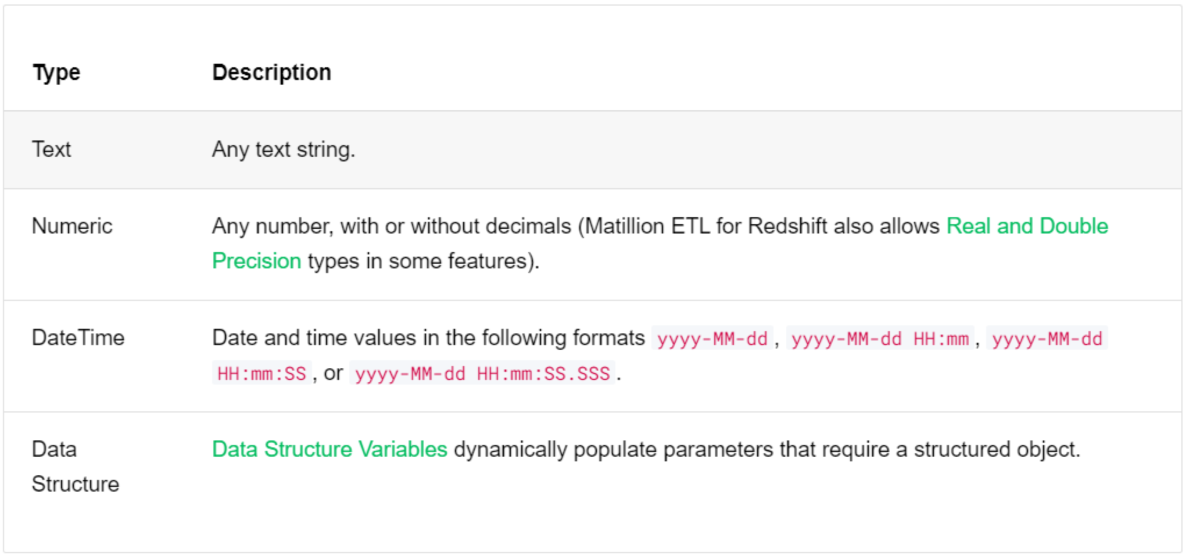

Manage Environment Variables

Name	Type	Behaviour	Example	Description
Millenium	Text	Shared		
Year	Numeric	Copied	2010	
ABC_DateTime	DateTime	Shared		

VARIABLES IN MATILLION

Types :

The value of any given variable must be one of the following types. This type defines how a variable can be used and is important to consider when passing the variable to components and functions. In the Matillion ETL client, types are often referred to by the symbols shown below:



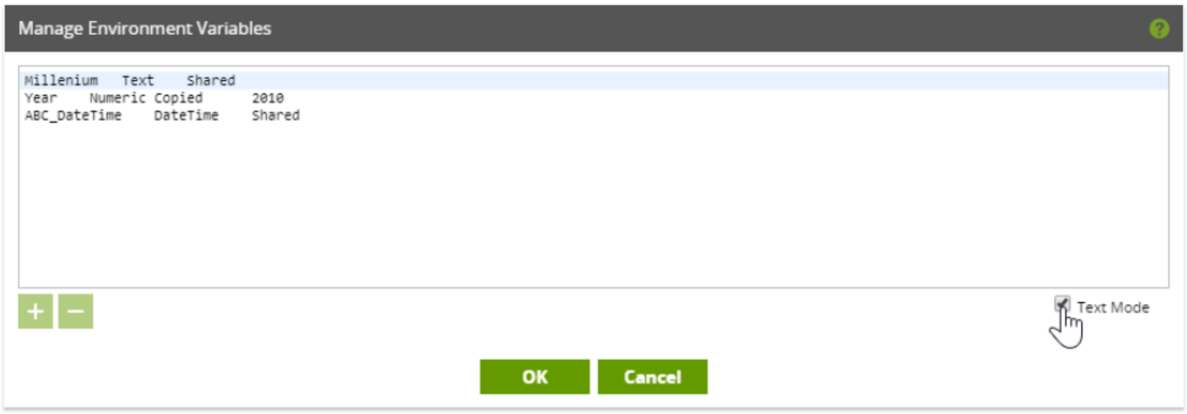
Type	Description
Text	Any text string.
Numeric	Any number, with or without decimals (Matillion ETL for Redshift also allows Real and Double Precision types in some features).
DateTime	Date and time values in the following formats `yyyy-MM-dd`, `yyyy-MM-dd HH:mm`, `yyyy-MM-dd HH:mm:SS`, or `yyyy-MM-dd HH:mm:SS.SSS`.
Data Structure	Data Structure Variables dynamically populate parameters that require a structured object.

Text Mode

On many menus, such as Manage Job Variables and Manage Environment Variables, users can switch to a **Text Mode** by ticking the checkbox next to **Text Mode**. Switching to **Text Mode** allows users to:

- Edit variables as a set of space-delimited text entries.
- Convert all existing variable data into an appropriate format.
- Copy and paste text, offering a fast way to move large numbers of variables.
- Have certain types of missing data auto-completed by Matillion ETL when parsing.

- Error check data before adding it to the project (Matillion ETL parses the data on exiting **Text Mode**, throwing a message if an error has been made).

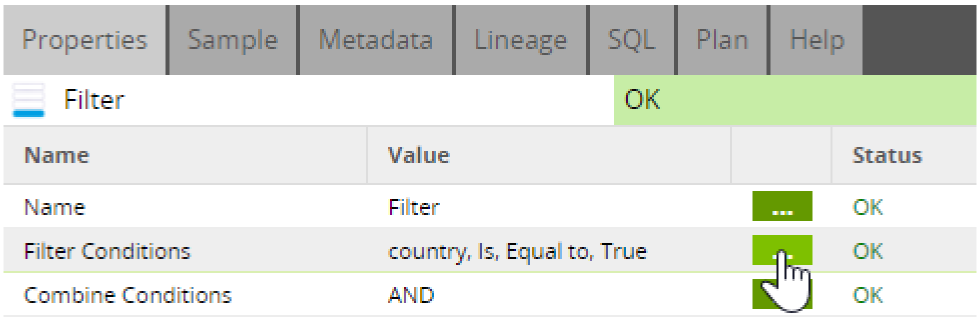


Manage Environment Variables

```
Millenium   Text    Shared
Year    Numeric Copied      2010
ABC_DateTime    DateTime    Shared
```

Filtering Booleans

It's possible to use boolean statements to filter values. In the example below, you can type **True** or **False** for the value combined with **AND** and **OR** conditional statements. You can also use **1** and **0** for the values, which work as **True** or **False** values, respectively.

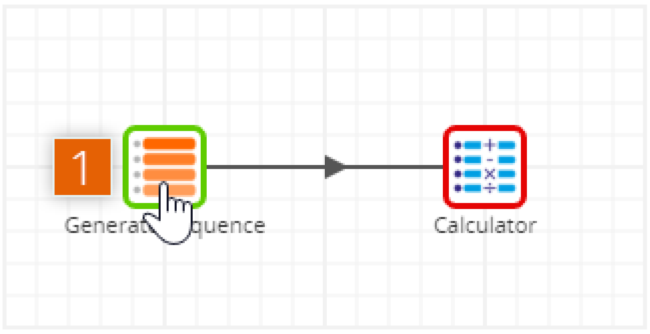


Name	Value		Status
Name	Filter	...	OK
Filter Conditions	country, Is, Equal to, True	...	OK
Combine Conditions	AND		OK

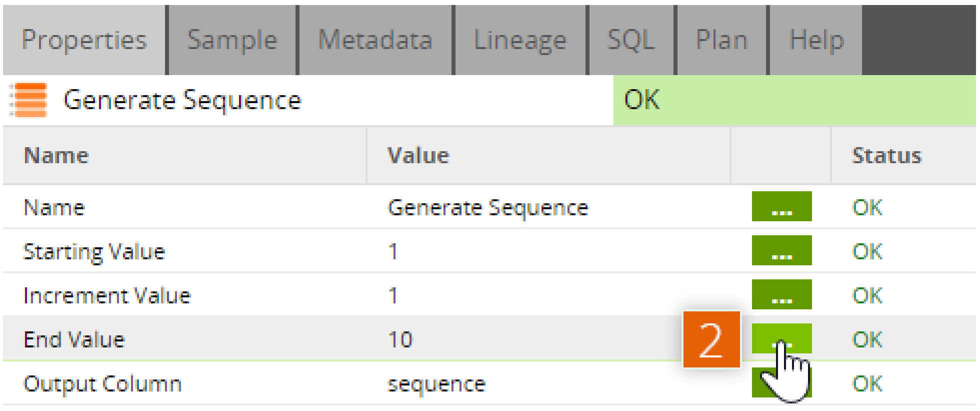
VARIABLES IN MATILLION

Example 1

As a basic example, a defined variable can be used to perform a simple calculation in a Transformation Job. The job consists of a **Generate Sequence** component linked to a **Calculator** component. The **Generate Sequence** component can be used to create a column of numbers that are passed to the **Calculator**.



1. Click the **Generate Sequence component** icon. Then, in the **Properties** panel, set the **Starting Value**, **Increment Value** and **End Value** to "1", "1" and "10", respectively. This will create a column of data that counts from 1 to 10.

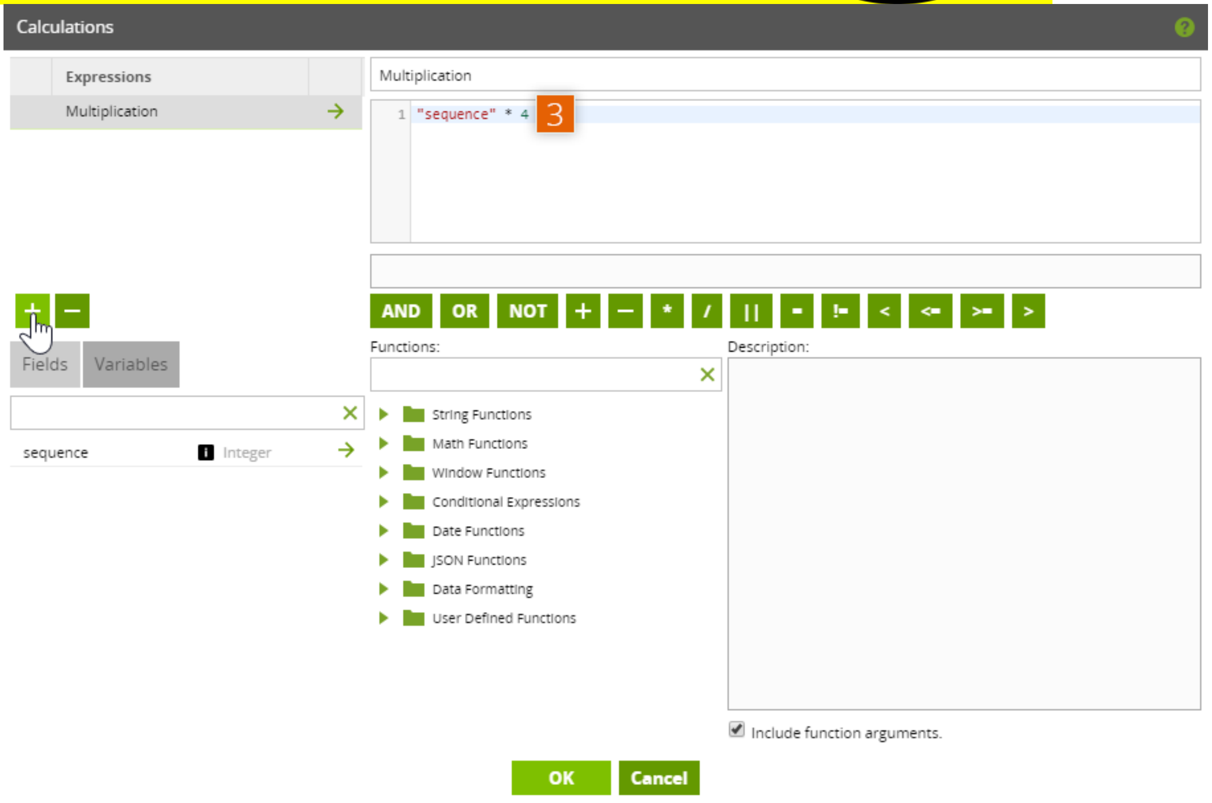


Name	Value		Status
Name	Generate Sequence	...	OK
Starting Value	1	...	OK
Increment Value	1	...	OK
End Value	10	...	OK
Output Column	sequence	...	OK

2. Next, click on the **Calculator** component icon. Enter the **Calculation Editor** and add a new calculation. This will allow values in the **sequence** row (output by the **Generate Sequence** component) to be multiplied, and then multiply that value by a number. In this case, each value in the **sequence** column is multiplied by 4 using the following calculation:

3. "sequence"*4

VARIABLES IN MATILLION



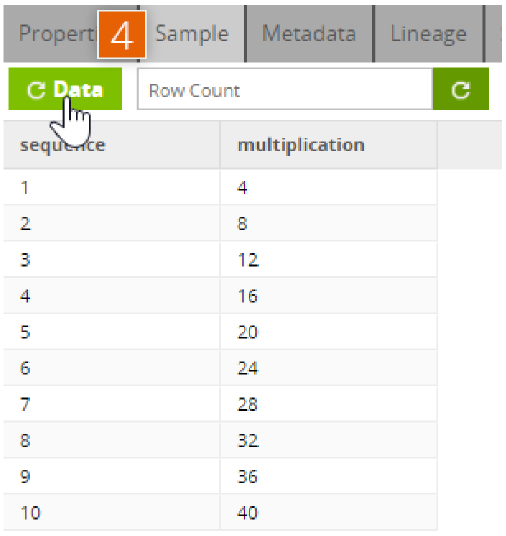
Calculations

Expressions
Multiplication

Multiplication

```
"sequence" * 4
```

4. The output of the **Calculator** can be viewed by clicking the **Sample** tab from within the component's **Properties** panel, then clicking ⟳ **Data**. It will show the sequence column is being multiplied by the number specified in the calculation.



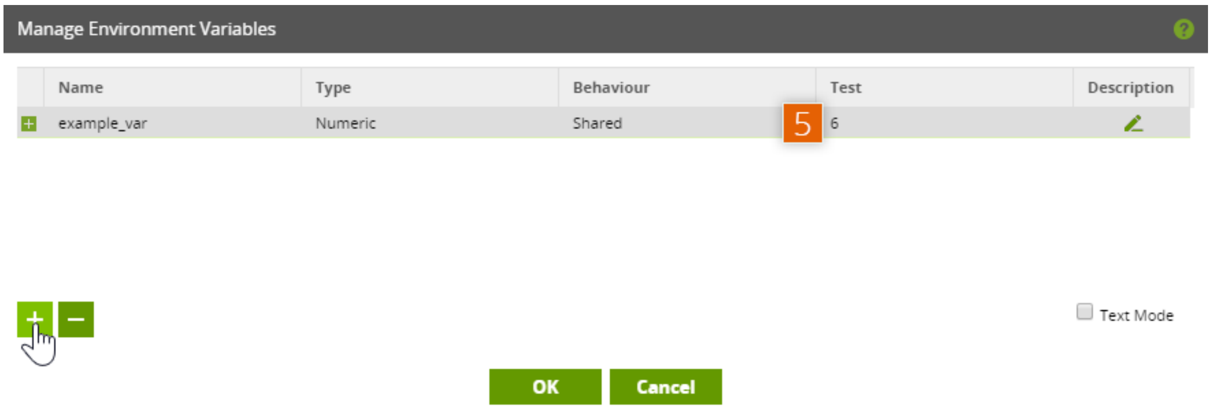
sequence	multiplication
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32
9	36
10	40

5. A variable in this calculation could also have been used instead. First, a variable must be declared via **Project**, then **Manage Environment Variables**. A new variable can be declared here and given a type, value and scope. Since the variable for this example will be used in a simple calculation, the **Numeric** type is appropriate.

   Note

   The scope is of little consequence for this job and can be set to **Shared**.

6. The default value can then be set to any number—6 is chosen in this example.

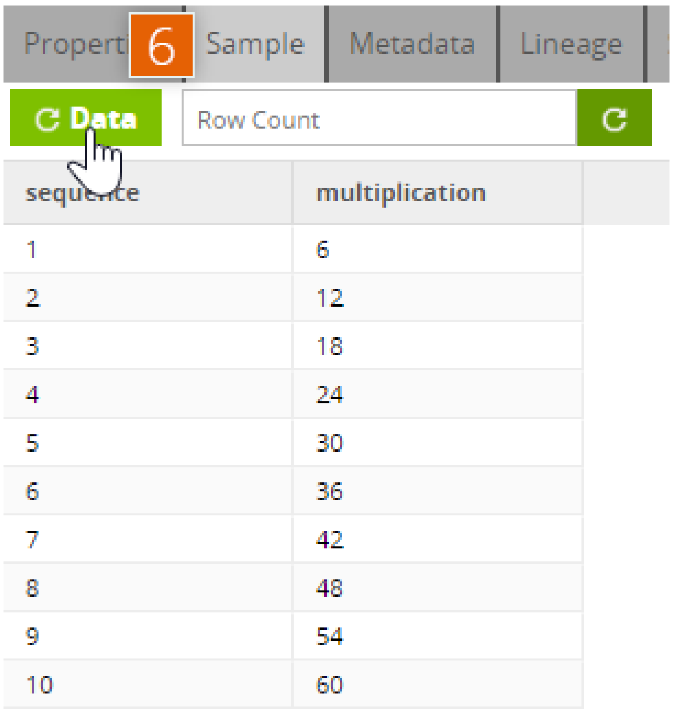


Name	Type	Behaviour	Test	Description
example_var	Numeric	Shared	6	

7. Finally, the **Calculator** component must be instructed to use this variable in its calculation. Reentering the **Calculation Editor**, the constant multiplier can be replaced with the newly declared variable using the following calculation: "sequence" * ${example_var}.

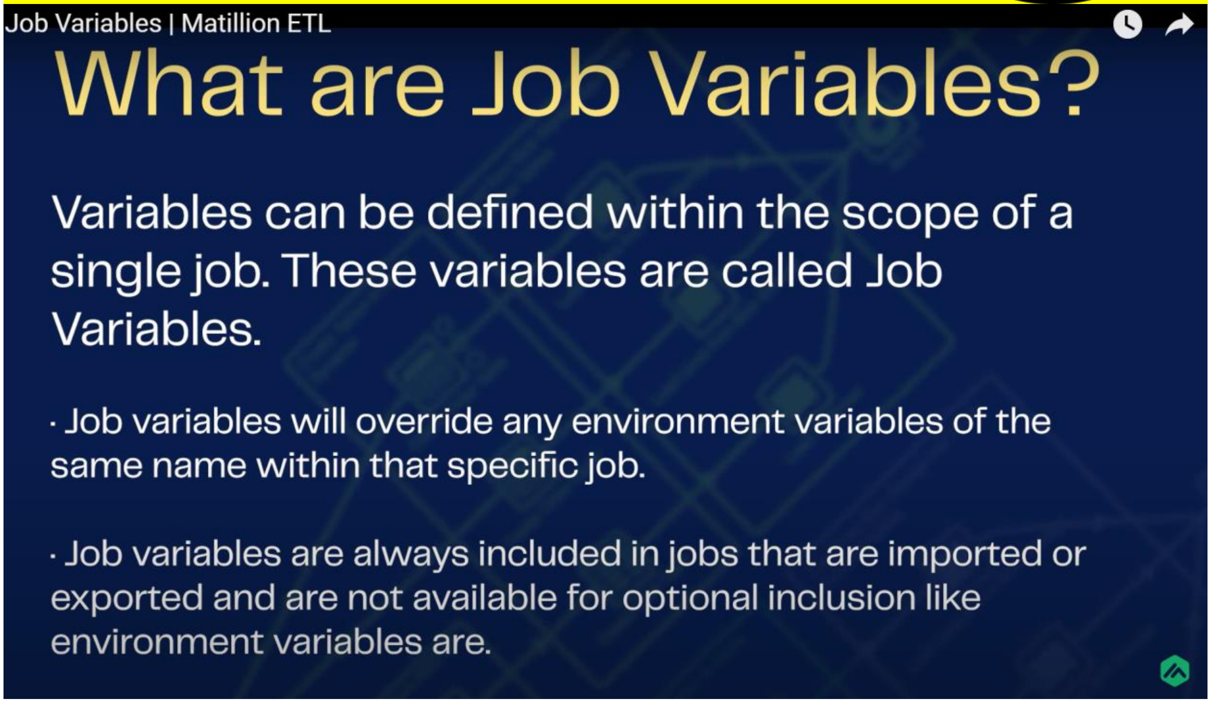
8. "sequence" * ${example_var}

9. Checking the sample output confirms that the **Calculator** component has correctly used the variable's default value in the calculation.



sequence	multiplication
1	6
2	12
3	18
4	24
5	30
6	36
7	42
8	48
9	54
10	60

VARIABLES IN MATILLION



Job Variables | Matillion ETL

What are Job Variables?

Variables can be defined within the scope of a single job. These variables are called Job Variables.

- Job variables will override any environment variables of the same name within that specific job.
- Job variables are always included in jobs that are imported or exported and are not available for optional inclusion like environment variables are.

Job Variables

Complete Video URL : https://youtu.be/sDzhsi-tvvI

Overview

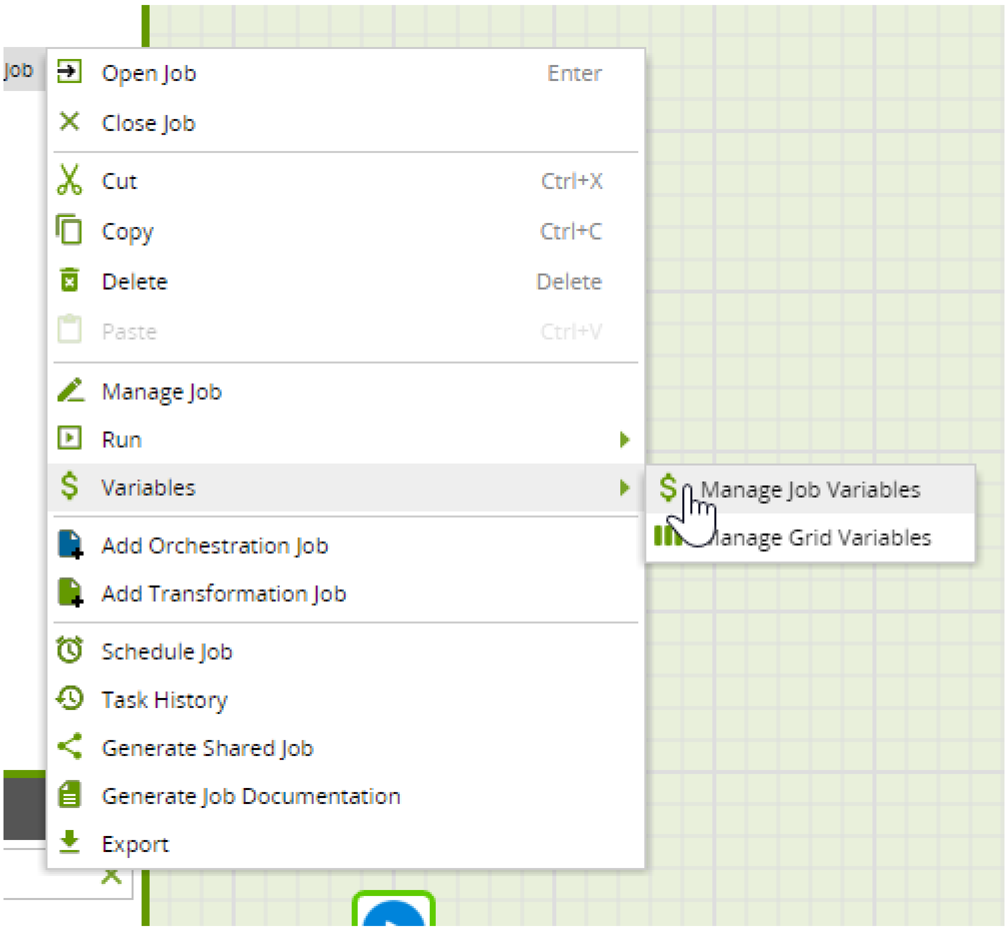
Variables can be defined within the scope of a single job. These variables can still be "copied" and "shared" with regard to their behaviour in a Fixed Flow component and Matillion ETL's various iterator components.

Job variables will override any environment variables of the same name within that specific job. Job variables are always included in jobs that are imported or exported and are not available for optional inclusion like environment variables are.

VARIABLES IN MATILLION

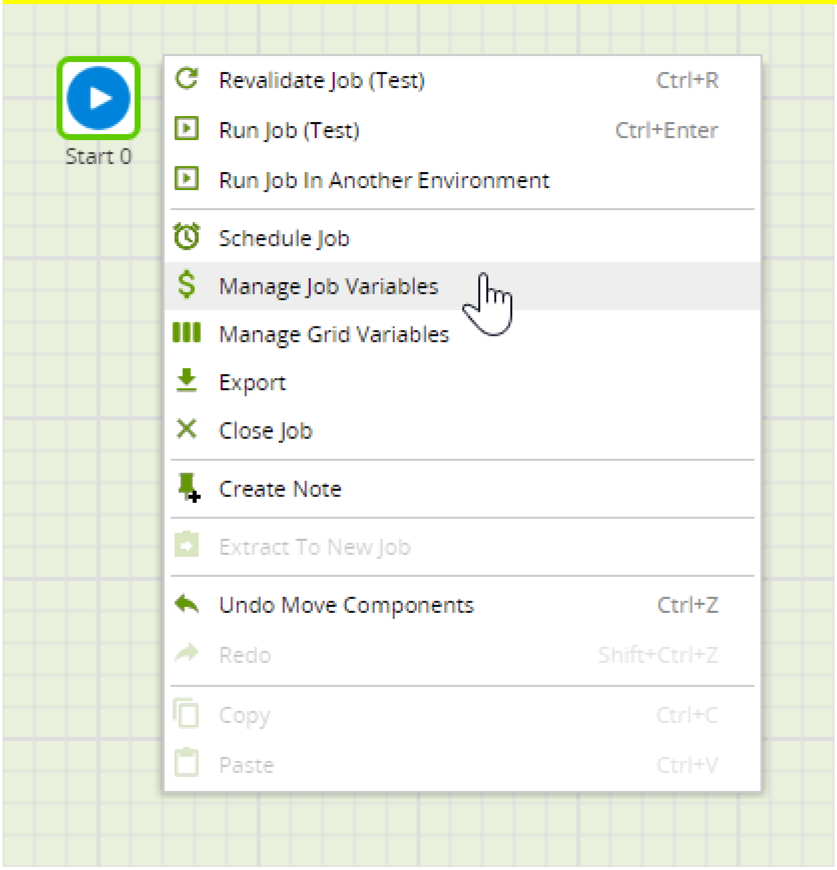
Creating job variables

To create a job-scoped variable, right-click on the desired orchestration or transformation job within the explorer panel and select **Manage Variables**. This will bring up the **Manage Job Variables** dialog.



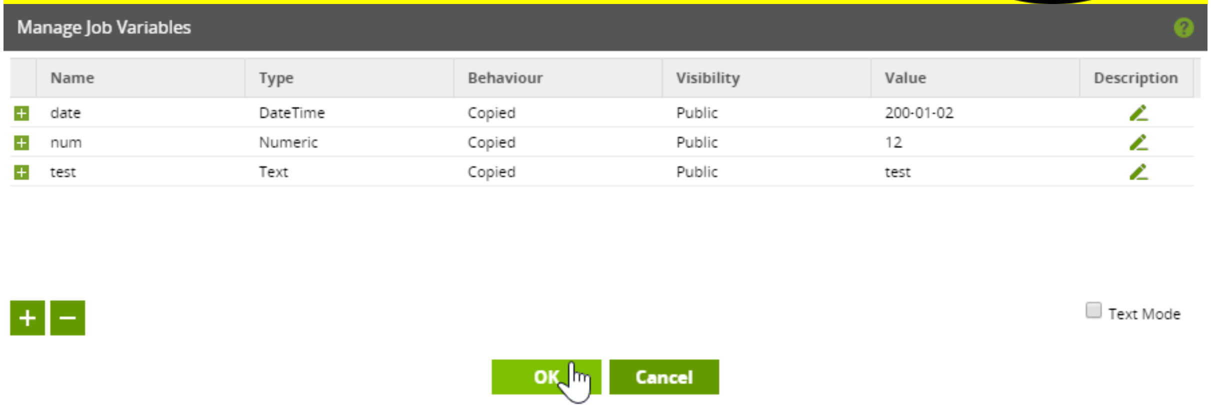
Users can also reach this dialog by right-clicking a component, such as the Start component, and clicking **Manage Job Variables**.

VARIABLES IN MATILLION



The **Manage Job Variables** dialog gives a complete list of all job variables with their properties:

- **Name:** Specify the name of the variable.
- **Type:** Specify the Matillion ETL data type of the variable. More information on variables.
- **Behaviour:** Set the branch behaviour inside a job. This determines how the variable is updated when more than a single job branch is making use of it. More information on the behaviour of variables.
- **Visibility:** Select either Public or Private. When a variable is private, it cannot be discovered or overwritten when this job is called from a Run Orchestration or Run Transformation component.
- **Value:** Set the default value for this variable in this job.


Manage Job Variables

	Name	Type	Behaviour	Visibility	Value	Description
⊞	date	DateTime	Copied	Public	200-01-02	✎
⊞	num	Numeric	Copied	Public	12	✎
⊞	test	Text	Copied	Public	test	✎

Click the edit button to edit that variable's description. The description has no consequence or functionality beyond being present in the dialog for editing the variable, and when the job is being called from a Run Orchestration or Run Transformation component.

Variables can also be edited via Text Mode by checking the checkbox.

Setting job variable values

When a job begins, all variables are initialized with their default value that was set in the **Manage Job Variables** dialog. The real power of a variable is that its value can be updated. You can update a variable via the following methods:

Iterator components

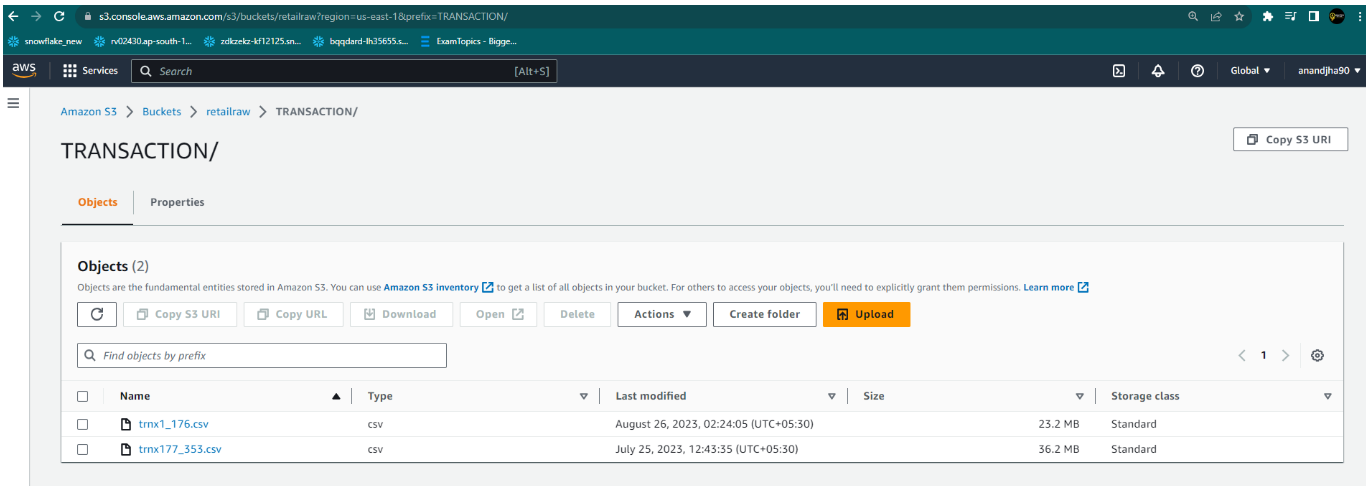
Iterator components work by setting variables to a new value for each iteration. This means you must define variables you wish to iterate in advance, and then use them when configuring iterator components. Iterations can be run in parallel, too, and in which case the variable scope must be set to "Copied" to ensure each parallel component sees its own copied copy of the variable.

VARIABLES IN MATILLION

Python scripts

Python scripts can push new values into variables using their built-in "context" object. See the Python Script component documentation for an example of this.

Say we want to load data from multiple CSV files from s3 bucket to our cloud data warehouse , we can use Job Variables to hold file locations and names so that we can load the data more easily



s3.console.aws.amazon.com/s3/buckets/retailraw?region=us-east-1&prefix=TRANSACTION/

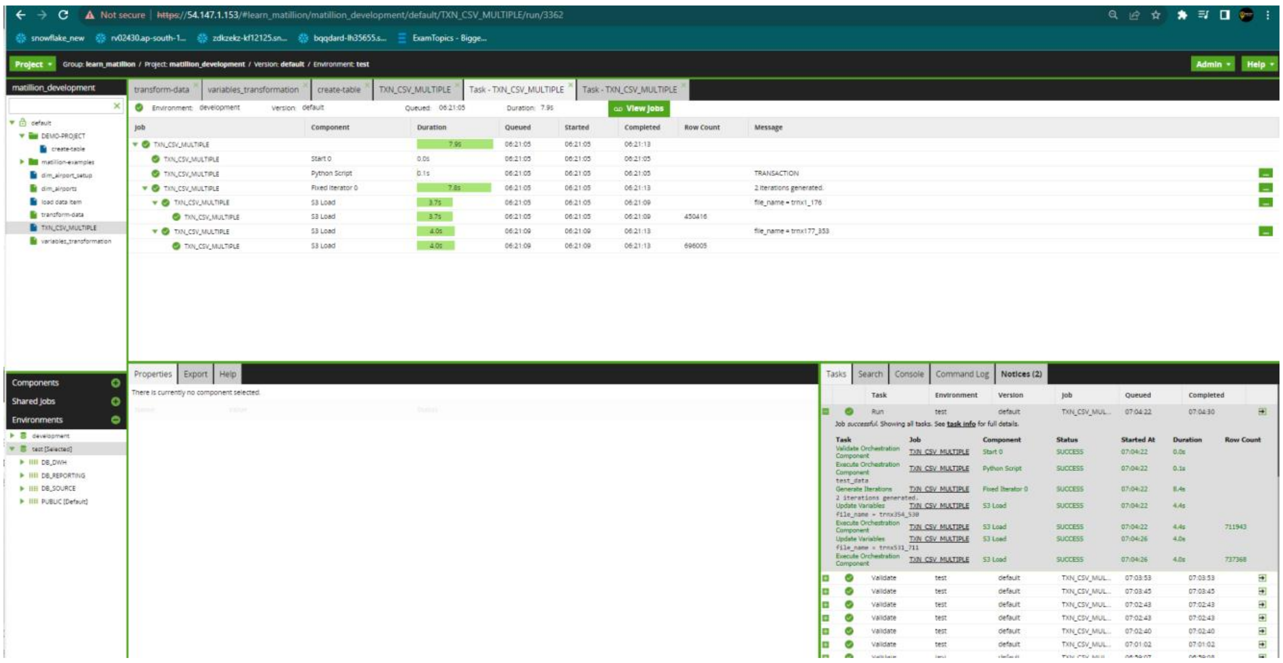
Amazon S3 > Buckets > retailraw > TRANSACTION/

TRANSACTION/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
trnx1_176.csv	csv	August 26, 2023, 02:24:05 (UTC+05:30)	23.2 MB	Standard
trnx177_353.csv	csv	July 25, 2023, 12:43:35 (UTC+05:30)	36.2 MB	Standard



https://54.147.1.153/#learn_matillion/matillion_development/default/TXN_CSV_MULTIPLE/run/3362

Group: learn_matillion / Project: matillion_development / Version: default / Environment: test

Environment: development Version: default Queued: 06:21:05 Duration: 7.9s

Job	Component	Duration	Queued	Started	Completed	Row Count	Message
TXN_CSV_MULTIPLE		7.9s	06:21:05	06:21:05	06:21:13		
TXN_CSV_MULTIPLE	Start 0	0.0s	06:21:05	06:21:05	06:21:05		
TXN_CSV_MULTIPLE	Python Script	0.1s	06:21:05	06:21:05	06:21:05		TRANSACTION
TXN_CSV_MULTIPLE	Fixed Iterator 0	7.8s	06:21:05	06:21:05	06:21:13		2 iterations generated.
TXN_CSV_MULTIPLE	S3 Load	3.7s	06:21:05	06:21:05	06:21:09		file_name = trnx1_176
TXN_CSV_MULTIPLE	S3 Load	3.7s	06:21:05	06:21:05	06:21:09	450416	
TXN_CSV_MULTIPLE	S3 Load	4.0s	06:21:09	06:21:09	06:21:13		file_name = trnx177_353
TXN_CSV_MULTIPLE	S3 Load	4.0s	06:21:09	06:21:09	06:21:13	696005	

There is currently no component selected.

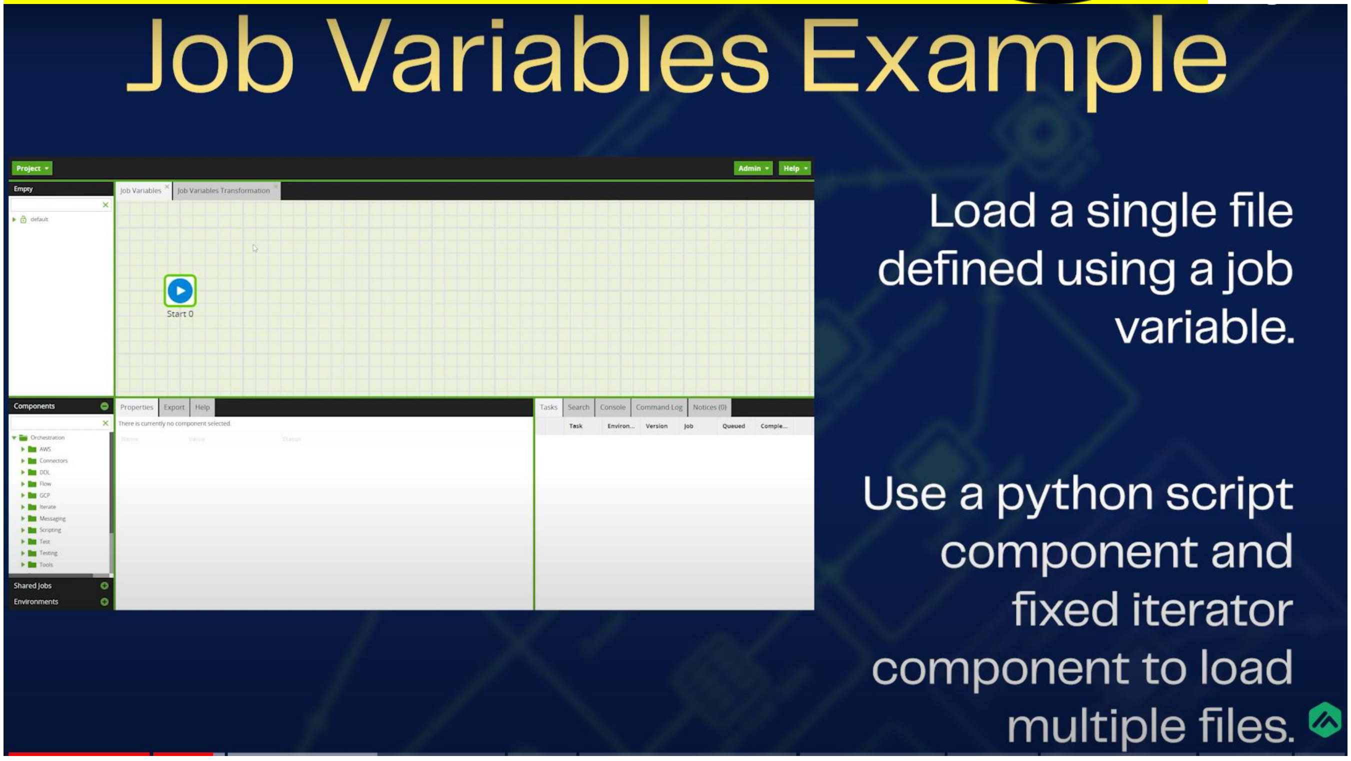
Task	Environment	Version	Job	Queued	Completed
Run	test	default	TXN_CSV_MUL...	07:04:22	07:04:30

Job successful. Showing all tasks. See task info for full details.

Task	Job	Component	Status	Started At	Duration	Row Count
Validate Orchestration Component	TXN_CSV_MULTIPLE	Start 0	SUCCESS	07:04:22	0.0s	
Execute Orchestration Component test_data	TXN_CSV_MULTIPLE	Python Script	SUCCESS	07:04:22	0.1s	
Generate Iterations 2 iterations generated.	TXN_CSV_MULTIPLE	Fixed Iterator 0	SUCCESS	07:04:22	8.4s	
Update Variables file_name = trnx354_530	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:22	4.4s	
Execute Orchestration Component	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:22	4.4s	711943
Update Variables file_name = trnx531_711	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:26	4.0s	
Execute Orchestration Component	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:26	4.0s	737368

Task	Environment	Version	Job	Queued	Completed
Validate	test	default	TXN_CSV_MUL...	07:03:53	07:03:53
Validate	test	default	TXN_CSV_MUL...	07:03:45	07:03:45
Validate	test	default	TXN_CSV_MUL...	07:02:43	07:02:43
Validate	test	default	TXN_CSV_MUL...	07:02:43	07:02:43
Validate	test	default	TXN_CSV_MUL...	07:02:40	07:02:40
Validate	test	default	TXN_CSV_MUL...	07:01:02	07:01:02

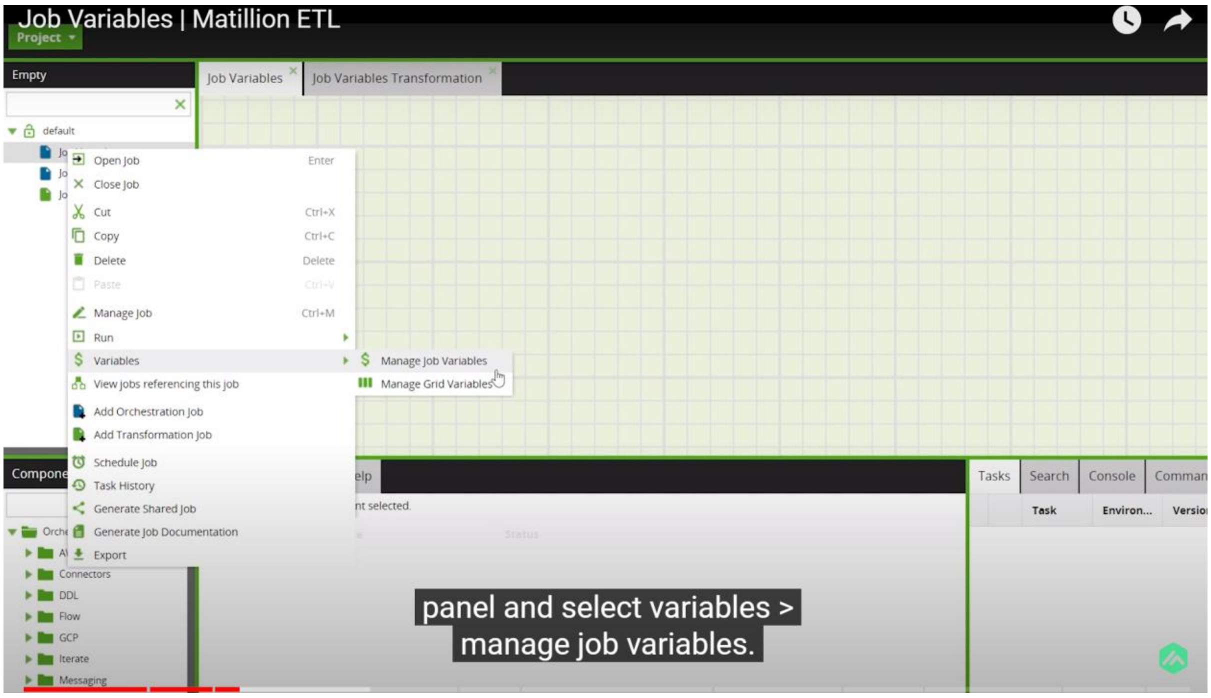
VARIABLES IN MATILLION



Job Variables Example

Load a single file defined using a job variable.

Use a python script component and fixed iterator component to load multiple files.



Job Variables | Matillion ETL

panel and select variables > manage job variables.

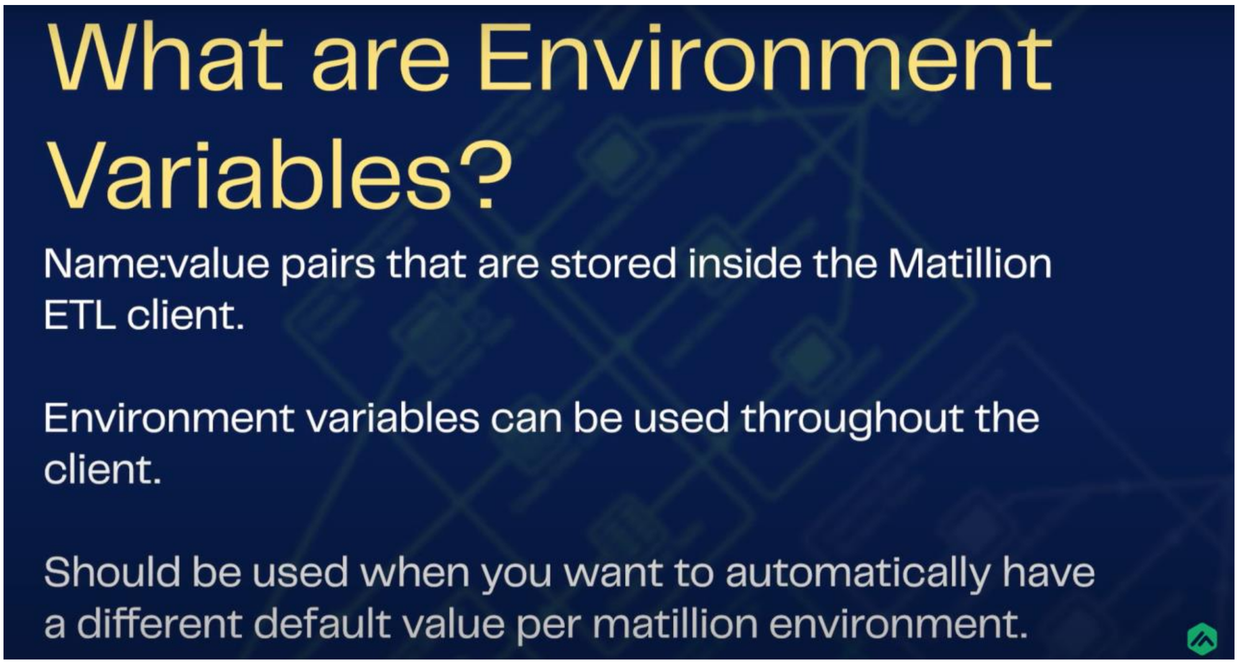
VARIABLES IN MATILLION

Environment Variables

Video Link URL : https://youtu.be/SuMukyDgiqU

Overview

Environment variables are name:value pairs that are stored inside the Matillion ETL client and are fully configurable by its users. Unlike Job Variables, environment variables can be used across Matillion ETL, in configurations and in all jobs through many components.



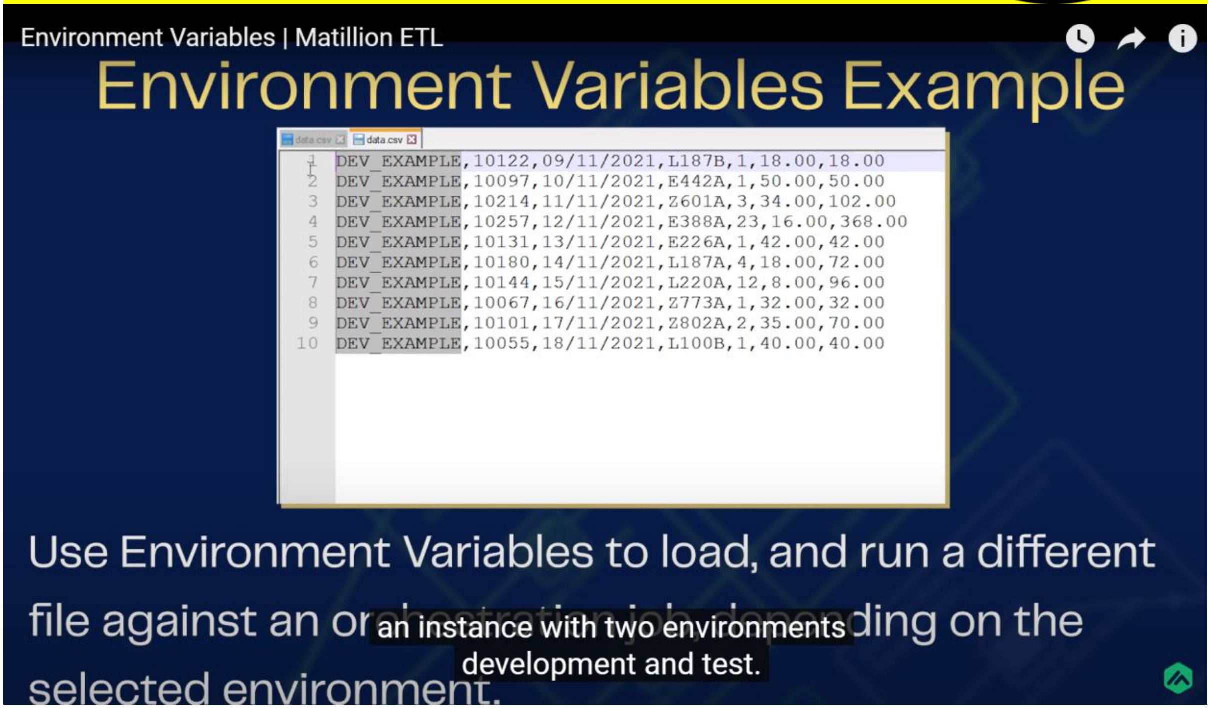
What are Environment Variables?

Name:value pairs that are stored inside the Matillion ETL client.

Environment variables can be used throughout the client.

Should be used when you want to automatically have a different default value per matillion environment.

VARIABLES IN MATILLION



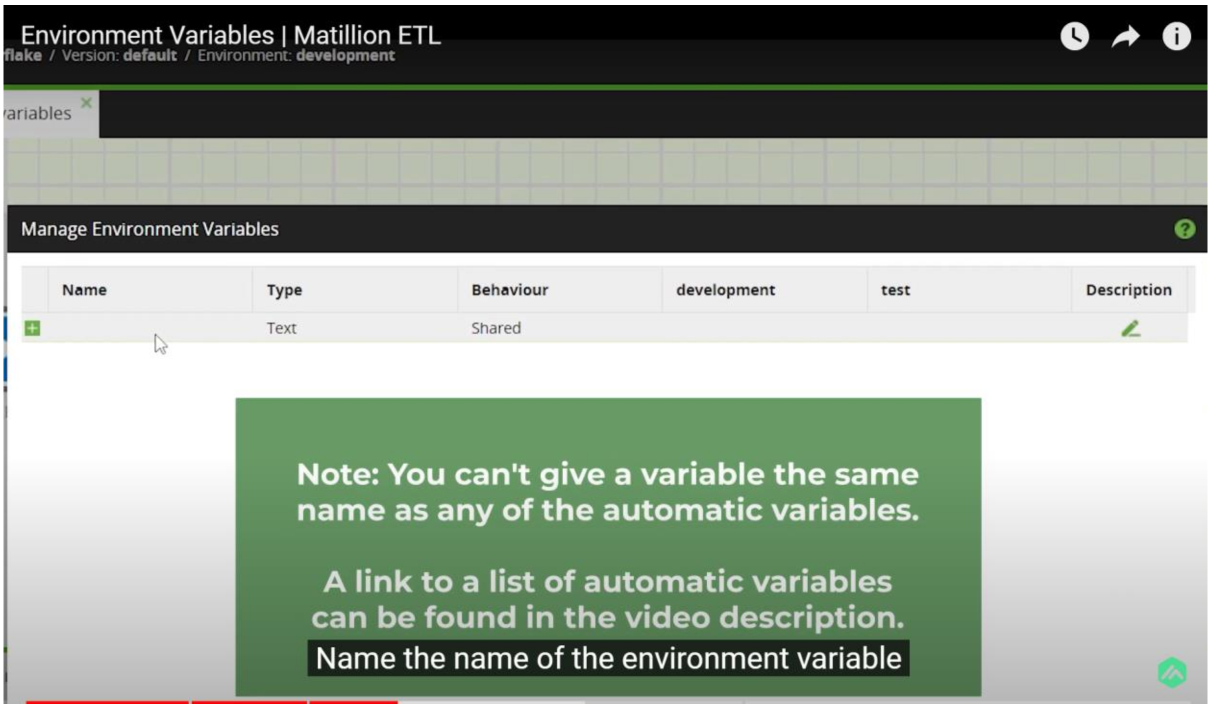
Environment Variables | Matillion ETL

Environment Variables Example

```
DEV_EXAMPLE,10122,09/11/2021,L187B,1,18.00,18.00
DEV_EXAMPLE,10097,10/11/2021,E442A,1,50.00,50.00
DEV_EXAMPLE,10214,11/11/2021,Z601A,3,34.00,102.00
DEV_EXAMPLE,10257,12/11/2021,E388A,23,16.00,368.00
DEV_EXAMPLE,10131,13/11/2021,E226A,1,42.00,42.00
DEV_EXAMPLE,10180,14/11/2021,L187A,4,18.00,72.00
DEV_EXAMPLE,10144,15/11/2021,L220A,12,8.00,96.00
DEV_EXAMPLE,10067,16/11/2021,Z773A,1,32.00,32.00
DEV_EXAMPLE,10101,17/11/2021,Z802A,2,35.00,70.00
DEV_EXAMPLE,10055,18/11/2021,L100B,1,40.00,40.00
```

Use Environment Variables to load, and run a different file against an orchestration job, depending on the selected environment.

an instance with two environments development and test.



Environment Variables | Matillion ETL

flake / Version: **default** / Environment: **development**

Manage Environment Variables

Name	Type	Behaviour	development	test	Description
	Text	Shared			

Note: You can't give a variable the same name as any of the automatic variables.

A link to a list of automatic variables can be found in the video description.

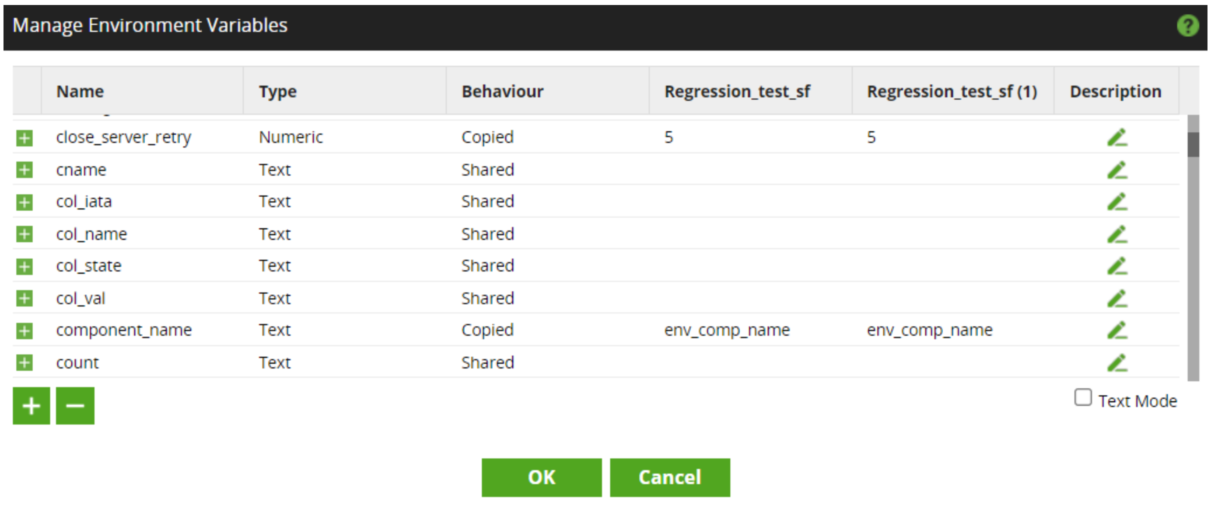
Name the name of the environment variable

Creating environment variables

VARIABLES IN MATILLION

Environment variables must be created, or "declared", before being used. To create an environment variable:

1. Click **Project** → **Manage Environment Variables**.
2. **Manage Environment Variables** will open.



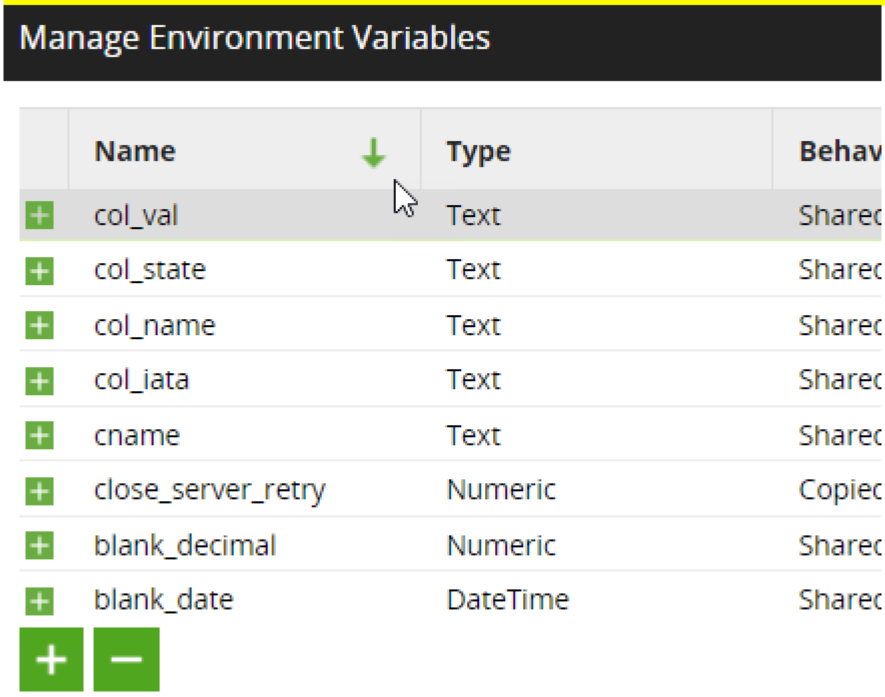
Name	Type	Behaviour	Regression_test_sf	Regression_test_sf (1)	Description
close_server_retry	Numeric	Copied	5	5	
cname	Text	Shared			
col_iata	Text	Shared			
col_name	Text	Shared			
col_state	Text	Shared			
col_val	Text	Shared			
component_name	Text	Copied	env_comp_name	env_comp_name	
count	Text	Shared			

The dialog lists every environment variable in a list. By default, they're sorted by **Name**, but you can sort the list on any column. To change the sort order, click a column title to sort in ascending order of that column, and click a second time to sort in descending order.

VARIABLES IN MATILLION

matillion

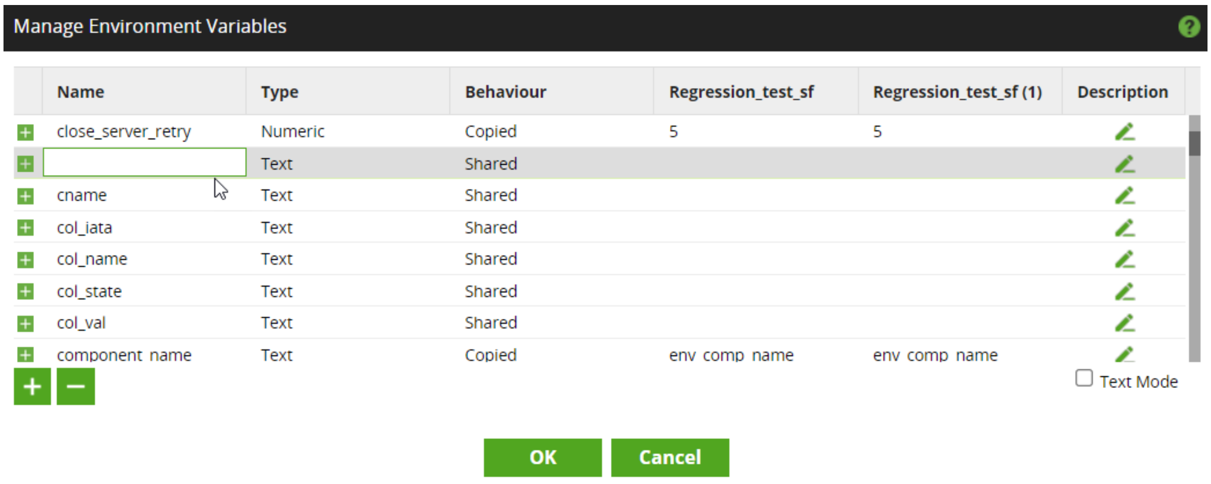
ANALYTICS WITH ANAND
LET THE DATA DO THE TALKING



Manage Environment Variables

	Name	Type	Behav
+	col_val	Text	Sharec
+	col_state	Text	Sharec
+	col_name	Text	Sharec
+	col_iata	Text	Sharec
+	cname	Text	Sharec
+	close_server_retry	Numeric	Copiec
+	blank_decimal	Numeric	Sharec
+	blank_date	DateTime	Sharec

1. To create a new environment variable, click **+** at the bottom of the overlay. This will create a blank row beneath the currently selected variable (or at the bottom of the list if none is selected), into which you can then type the details of the new variable.



Manage Environment Variables

	Name	Type	Behaviour	Regression_test_sf	Regression_test_sf (1)	Description
+	close_server_retry	Numeric	Copied	5	5	
+		Text	Shared			
+	cname	Text	Shared			
+	col_iata	Text	Shared			
+	col_name	Text	Shared			
+	col_state	Text	Shared			
+	col_val	Text	Shared			
+	component_name	Text	Copied	env_comp_name	env_comp_name	

☐ Text Mode

OK Cancel

1. Provide the following details for the new variable:

2. **Name:** The name of the environment variable. Read Variables for variable naming guidelines. Note that you can't give a variable the same name as any of the automatic variables listed below.
3. **Type:** The Matillion ETL data type of the environment variable: Text, Numeric, or DateTime. **Text** is selected by default. Read Variables for more information on data types.
4. **Behaviour:** The behaviour of the environment variable will be **Shared** or **Copied**, depending on how the variable will be used by multiple job branches. **Shared** is selected by default.
5. **Default Environment Value:** The default value of the variable in each environment. The dialog includes a separate column for each environment (two environments, called **Regression_test_sf** and **Regression_test_sf (1)**, are shown in the preceding image) with the default values listed in the column. A variable's default value can be left blank, but failing to give a reasonable default value may cause problems in validating components that use that variable—for example, if a component uses a DateTime variable to which you have given a default value that is not in a valid DateTime format.
6. **Description:** An environment variable can optionally be given a description, which is a free-form text field that can contain any information about the variable that you wish to record. Descriptions are not displayed by default; to view descriptions, click **+** to the right of variable. To add or edit a description, click the edit button in the **Description** column, edit your text in the **Description** dialog, and then click **OK**.

To edit an existing environment variable, click in the column whose value you want to change, and then:

- For **Name** or **Default Environment Values**, type the new text, then press the **Enter** key on your keyboard or click elsewhere in the dialog.
- For **Type** or **Behaviour**, select the required value from the menu.

To delete an environment variable, click on the variable you want to delete (this will put you in editing mode, but ignore this) and then click the remove button at the bottom of the dialog.

Variables can also be added or edited in text mode by selecting the **Text Mode** checkbox.

Setting environment variable values

When a job begins, all variables are initialized with their default environment value, which should be set in the **Manage Environment Variables** overlay. The real power of a variable is that its value can be updated as a job runs, through the following methods:

Iteration components

Iteration components work by setting variables to a new value for each iteration. You must first define the variables you wish to iterate, using the **Manage Environment Variables** dialog, and then use those variables when configuring iteration components.

Iterations can be run in parallel, in which case the variable behaviour must be set to **Copied** to ensure each parallel component sees and iterates its own copy of the variable.

Python scripts

Python scripts can push new values into variables using their built-in context object, following this format:

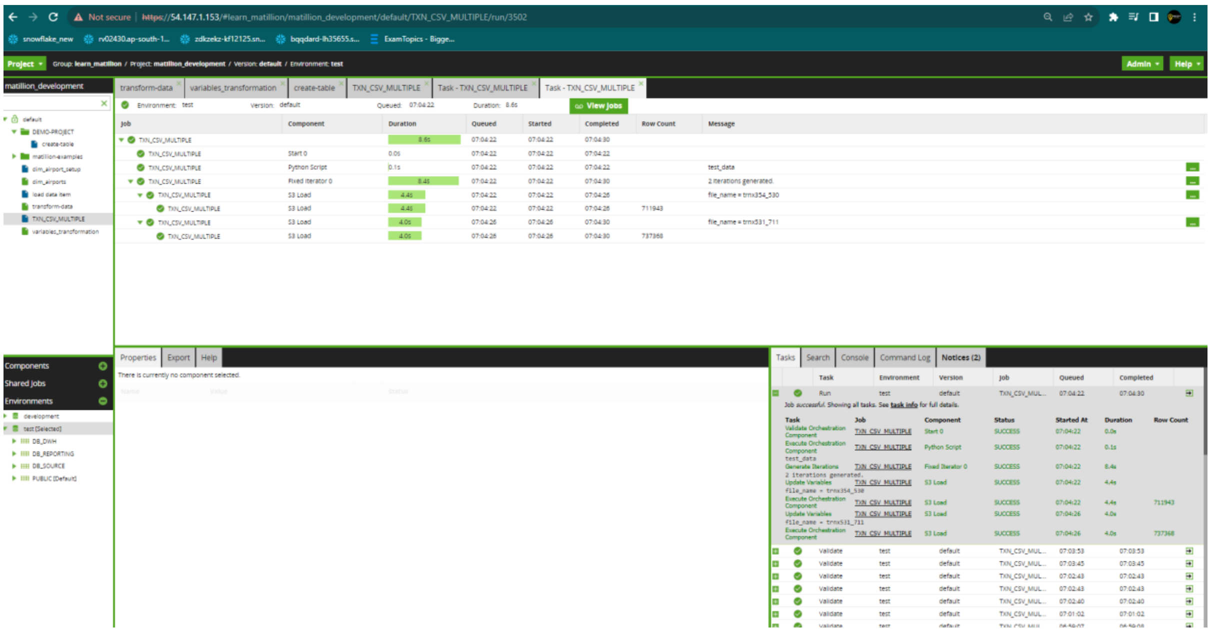
context.updateVariable("variable", "new value")

Read the Python Script component documentation for more information.

SQS runs

Job executions triggered by SQS messages can also set variable values using the optional 'variables' item when sending the message. Read Manage SQS Configuration for more information, including the syntax for setting variables on the incoming message.

VARIABLES IN MATILLION



Job	Component	Duration	Queued	Started	Completed	Row Count	Message
TXN_CSV_MULTIPLE		8.6s	07:04:22	07:04:22	07:04:30		
TXN_CSV_MULTIPLE	Start 0	0.0s	07:04:22	07:04:22	07:04:22		
TXN_CSV_MULTIPLE	Python Script	0.1s	07:04:22	07:04:22	07:04:22		test_data
TXN_CSV_MULTIPLE	Fixed Iterator 0	8.4s	07:04:22	07:04:22	07:04:30		2 iterations generated.
TXN_CSV_MULTIPLE	S3 Load	4.4s	07:04:22	07:04:22	07:04:26		file_name = trnx354_530
TXN_CSV_MULTIPLE	S3 Load	4.4s	07:04:22	07:04:22	07:04:26	711943	
TXN_CSV_MULTIPLE	S3 Load	4.0s	07:04:26	07:04:26	07:04:30		file_name = trnx531_711
TXN_CSV_MULTIPLE	S3 Load	4.0s	07:04:26	07:04:26	07:04:30	737368	

Task	Environment	Version	Job	Queued	Completed
Run	test	default	TXN_CSV_MUL...	07:04:22	07:04:30

Job successful. Showing all tasks. See task info for full details.

Task	Job	Component	Status	Started At	Duration	Row Count
Validate Orchestration Component	TXN_CSV_MULTIPLE	Start 0	SUCCESS	07:04:22	0.0s	
Execute Orchestration Component test_data	TXN_CSV_MULTIPLE	Python Script	SUCCESS	07:04:22	0.1s	
Generate Iterations 2 iterations generated.	TXN_CSV_MULTIPLE	Fixed Iterator 0	SUCCESS	07:04:22	8.4s	
Update Variables file_name = trnx354_530	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:22	4.4s	
Execute Orchestration Component	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:22	4.4s	711943
Update Variables file_name = trnx531_711	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:26	4.0s	
Execute Orchestration Component	TXN_CSV_MULTIPLE	S3 Load	SUCCESS	07:04:26	4.0s	737368

Task	Environment	Version	Job	Queued	Completed
Validate	test	default	TXN_CSV_MUL...	07:03:53	07:03:53
Validate	test	default	TXN_CSV_MUL...	07:03:45	07:03:45
Validate	test	default	TXN_CSV_MUL...	07:02:43	07:02:43
Validate	test	default	TXN_CSV_MUL...	07:02:43	07:02:43
Validate	test	default	TXN_CSV_MUL...	07:02:40	07:02:40
Validate	test	default	TXN_CSV_MUL...	07:01:02	07:01:02

There is currently no component selected.

Variable exports

The **Export** tab of the **Properties** panel allows you to edit mappings between the runtime information that the component makes available and the variables you have already defined. All orchestration components and some transformation components support exporting runtime information into variables during job execution. A list of common exports is given in Component Exports.

Component export values are also used to populate the tasks view and the run history. To export a value, it must be mapped to an existing variable through the component's **Export** tab. It is important to map the value to a variable of the correct type according to the table given at Component Exports.

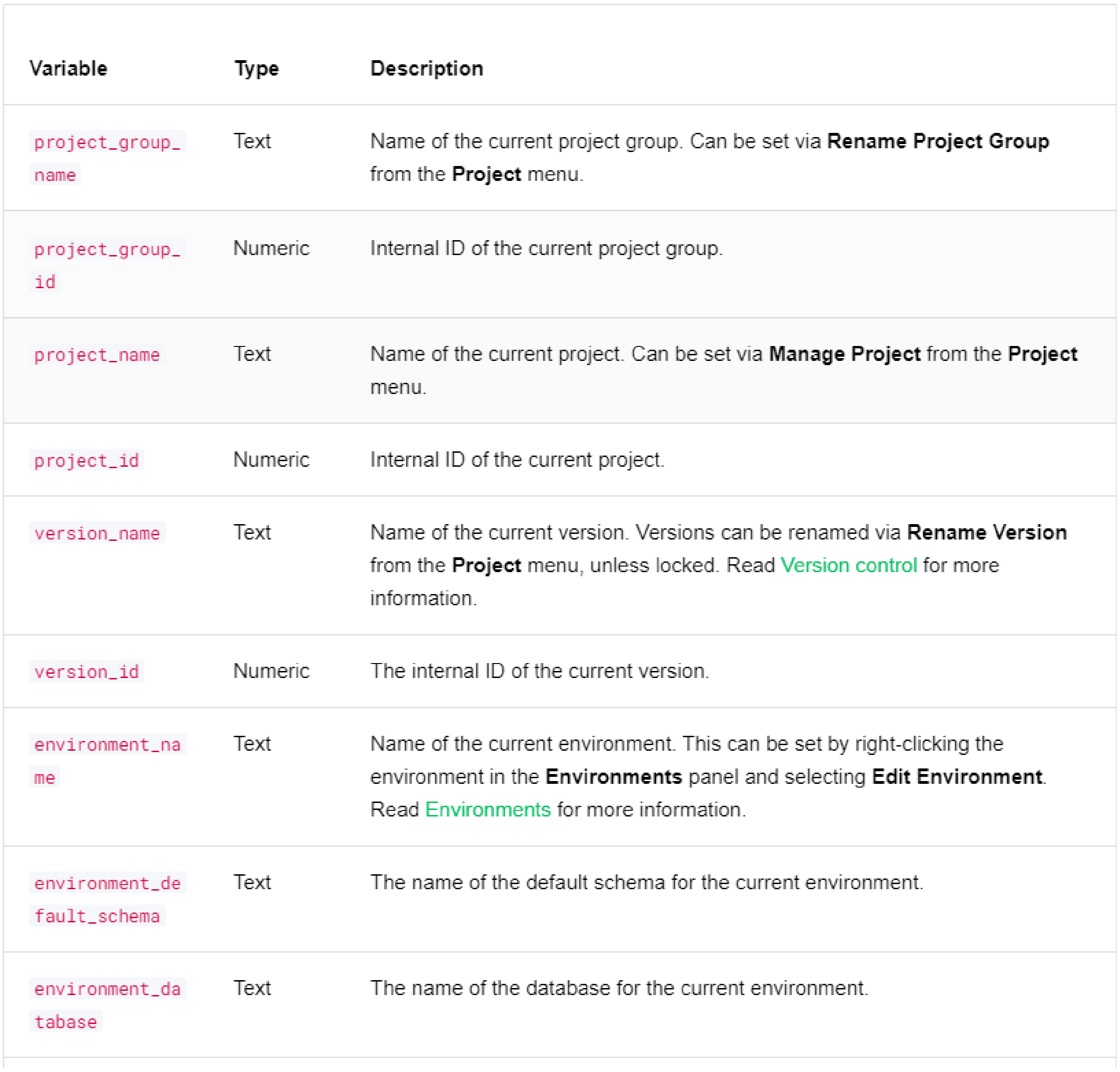
Note

When an environment variable is exported or imported, it is only the name of the variable that is exported/imported, not the value of the variable.

VARIABLES IN MATILLION

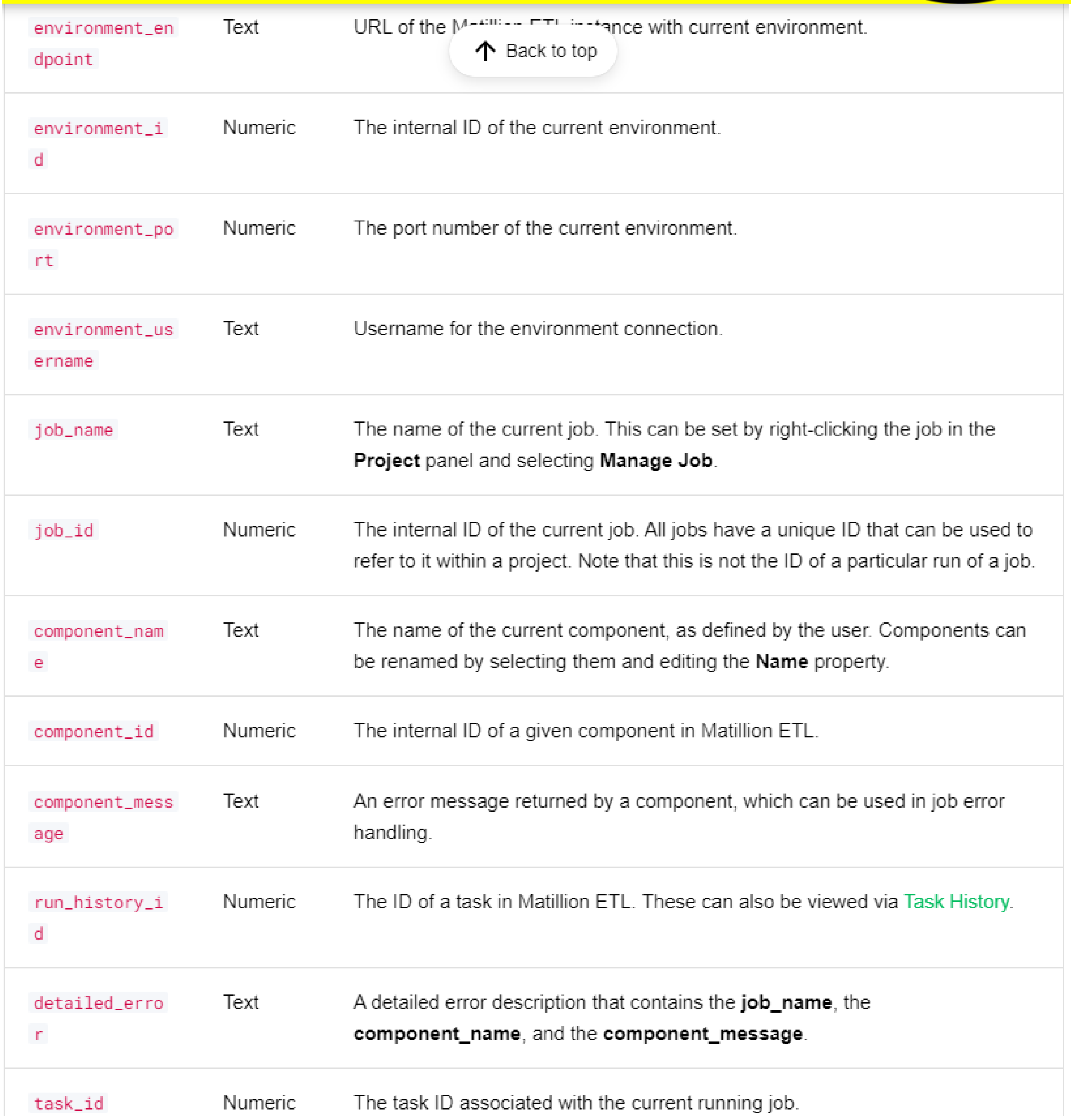
Automatic variables

The following environment variables are automatically available without first needing to be defined. The values of these variables cannot be set in **Manage Environment Variables** but must be set as described in this table



Variable	Type	Description
`project_group_name`	Text	Name of the current project group. Can be set via **Rename Project Group** from the **Project** menu.
`project_group_id`	Numeric	Internal ID of the current project group.
`project_name`	Text	Name of the current project. Can be set via **Manage Project** from the **Project** menu.
`project_id`	Numeric	Internal ID of the current project.
`version_name`	Text	Name of the current version. Versions can be renamed via **Rename Version** from the **Project** menu, unless locked. Read Version control for more information.
`version_id`	Numeric	The internal ID of the current version.
`environment_name`	Text	Name of the current environment. This can be set by right-clicking the environment in the **Environments** panel and selecting **Edit Environment**. Read Environments for more information.
`environment_default_schema`	Text	The name of the default schema for the current environment.
`environment_database`	Text	The name of the database for the current environment.

VARIABLES IN MATILLION



`environment_endpoint`	Text	URL of the Matillion ETL instance with current environment.
`environment_id`	Numeric	The internal ID of the current environment.
`environment_port`	Numeric	The port number of the current environment.
`environment_username`	Text	Username for the environment connection.
`job_name`	Text	The name of the current job. This can be set by right-clicking the job in the **Project** panel and selecting **Manage Job**.
`job_id`	Numeric	The internal ID of the current job. All jobs have a unique ID that can be used to refer to it within a project. Note that this is not the ID of a particular run of a job.
`component_name`	Text	The name of the current component, as defined by the user. Components can be renamed by selecting them and editing the **Name** property.
`component_id`	Numeric	The internal ID of a given component in Matillion ETL.
`component_message`	Text	An error message returned by a component, which can be used in job error handling.
`run_history_id`	Numeric	The ID of a task in Matillion ETL. These can also be viewed via Task History.
`detailed_error`	Text	A detailed error description that contains the **job_name**, the **component_name**, and the **component_message**.
`task_id`	Numeric	The task ID associated with the current running job.

All of the above also have an id variable that is an internally generated ID and should be avoided in most cases.

Manipulating environment variables via the v1 API

Some examples of common ways to manipulate environment variables via the v1 API are given below.

When using these examples you should replace **api-user** and **api-password** with the username and password you use to access the instance. This user will require API access as well as any permissions required to alter environment variables.

These examples can be used through cURL by replacing **[InstanceAddress]**, **[GroupName]**, **[ProjectName]**, **[Environment Name]**, and **[VariableName]** with details appropriate for your resources.

To list all environment variables in an environment:

```
curl -X GET -o varlist.json -u api-user:api-password
"http://<InstanceAddress>/rest/v1/group/name/<GroupName>/project/name/<ProjectName>/environment/name/<EnvironmentName>/variable"
```

Using that list of names, you can add ***/name/*** to the cURL command to get the default value for variable :

```
curl -X GET -o varlist.json -u api-user:api-password
"http://<InstanceAddress>/rest/v1/group/name/<GroupName>/project/name/<ProjectName>/environment/name/<EnvironmentName>/variable/name/<VariableName>"
```

This is not useful on its own, but can be used to manipulate that environment variable in the ways shown below.

Add ***/value*** to get the default value for variable :

```
curl -X GET -o varlist.json -u api-user:api-password
"http://<InstanceAddress>/rest/v1/group/name/<GroupName>/project/name/<ProjectName>/environment/name/<EnvironmentName>/variable/name/<VariableName>/value"
```

Add ***/delete*** to delete the environment variable:

```
curl -X POST -u api-user:api-password
"http://<InstanceAddress>/rest/v1/group/name/<GroupName>/project/name/<ProjectName>/environment/name/<EnvironmentName>/variable/name/<VariableName>/delete"
```

Add **/set/value/** to update this environment variable to a new default value:

```
curl -X POST -u api-user:api-password "http://<InstanceAddr
```

Grid variables

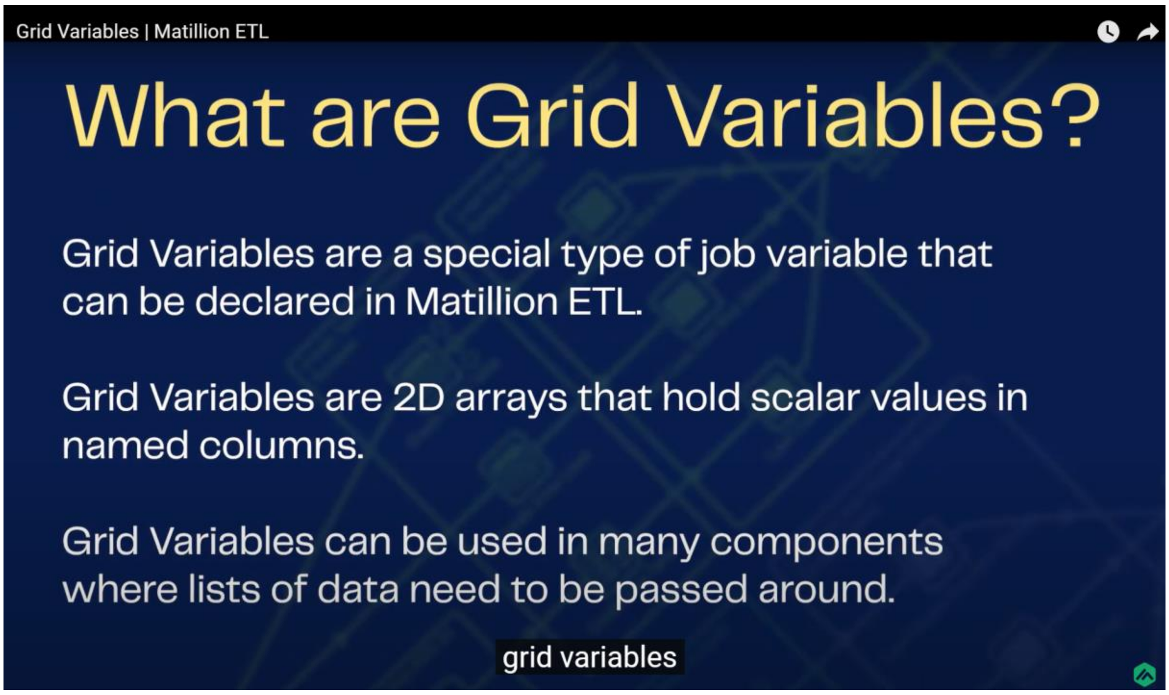
Grid Variables Video Guide

https://youtu.be/nlC6Kw5DsEQ

Overview

Grid variables are a special type of job variable that can be declared in Matillion ETL. Grid variables are 2D arrays that hold scalar values in named columns.

Grid variables can be used in many components (usually via the **Use Grid Variable** checkbox in component property dialogs) where lists of data need to be passed around. For example, a grid variable can be used to easily populate table metadata, or to pass arrays of data for use in Python scripts. In this article we give just a few examples of their utility.



Grid Variables | Matillion ETL

What are Grid Variables?

Grid Variables are a special type of job variable that can be declared in Matillion ETL.

Grid Variables are 2D arrays that hold scalar values in named columns.

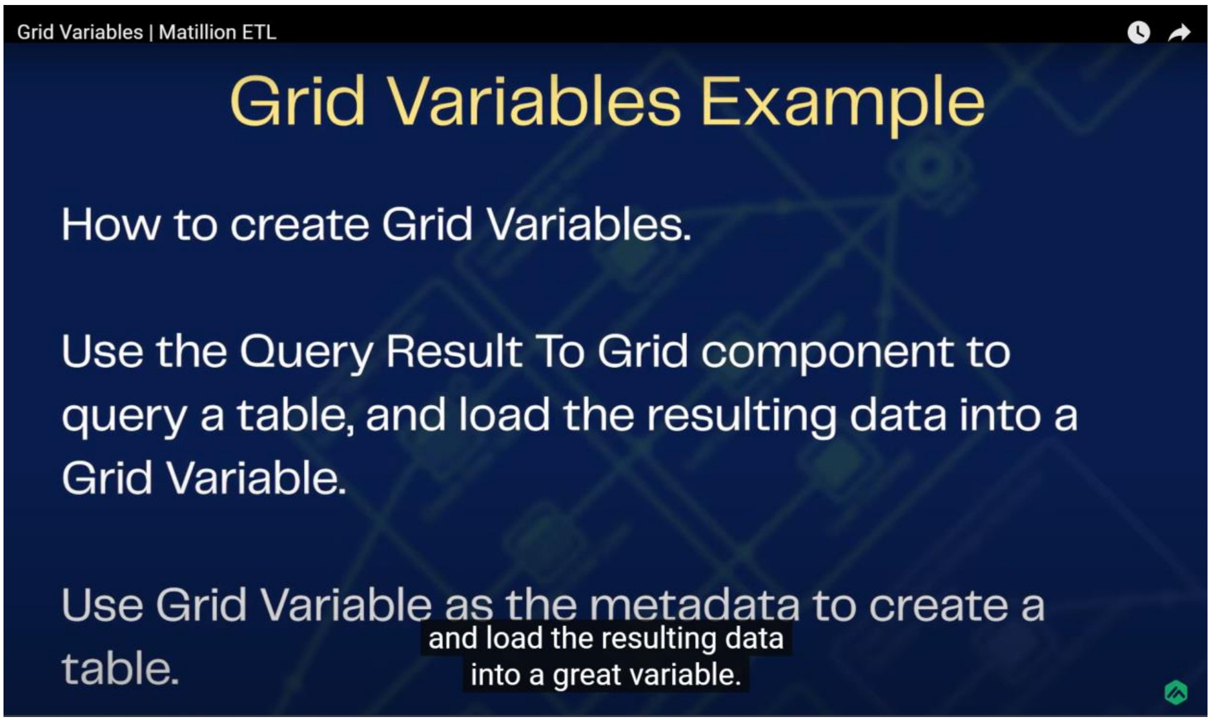
Grid Variables can be used in many components where lists of data need to be passed around.

grid variables

VARIABLES IN MATILLION

The following Matillion ETL components can be used when working with grid variables.

- Table Metadata To Grid: Takes metadata from a table and loads it into a grid variable.
- Remove From Grid: Removes rows from a preexisting grid variable.
- Append To Grid: Adds rows to a preexisting grid variable.
- Query Result To Grid: Queries a table and loads the resulting data into a grid variable.


Grid Variables | Matillion ETL

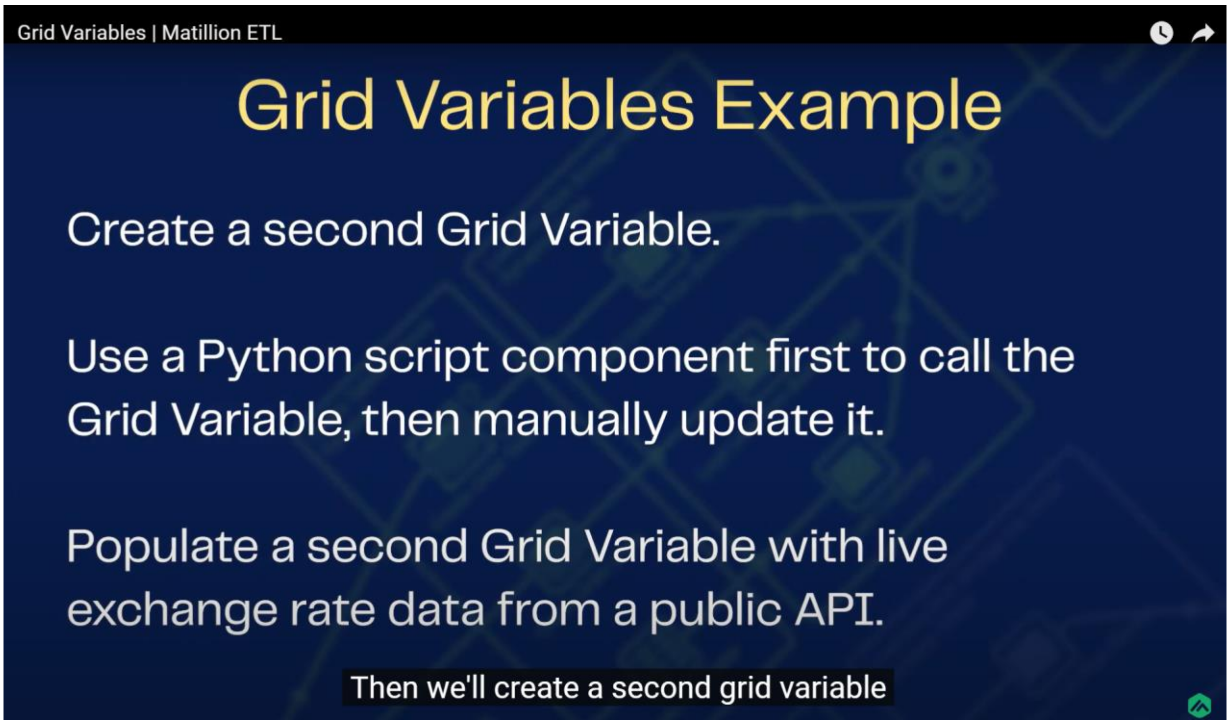
Grid Variables Example

How to create Grid Variables.

Use the Query Result To Grid component to query a table, and load the resulting data into a Grid Variable.

Use Grid Variable as the metadata to create a table.

and load the resulting data into a great variable.



Grid Variables | Matillion ETL

Grid Variables Example

Create a second Grid Variable.

Use a Python script component first to call the Grid Variable, then manually update it.

Populate a second Grid Variable with live exchange rate data from a public API.

Then we'll create a second grid variable

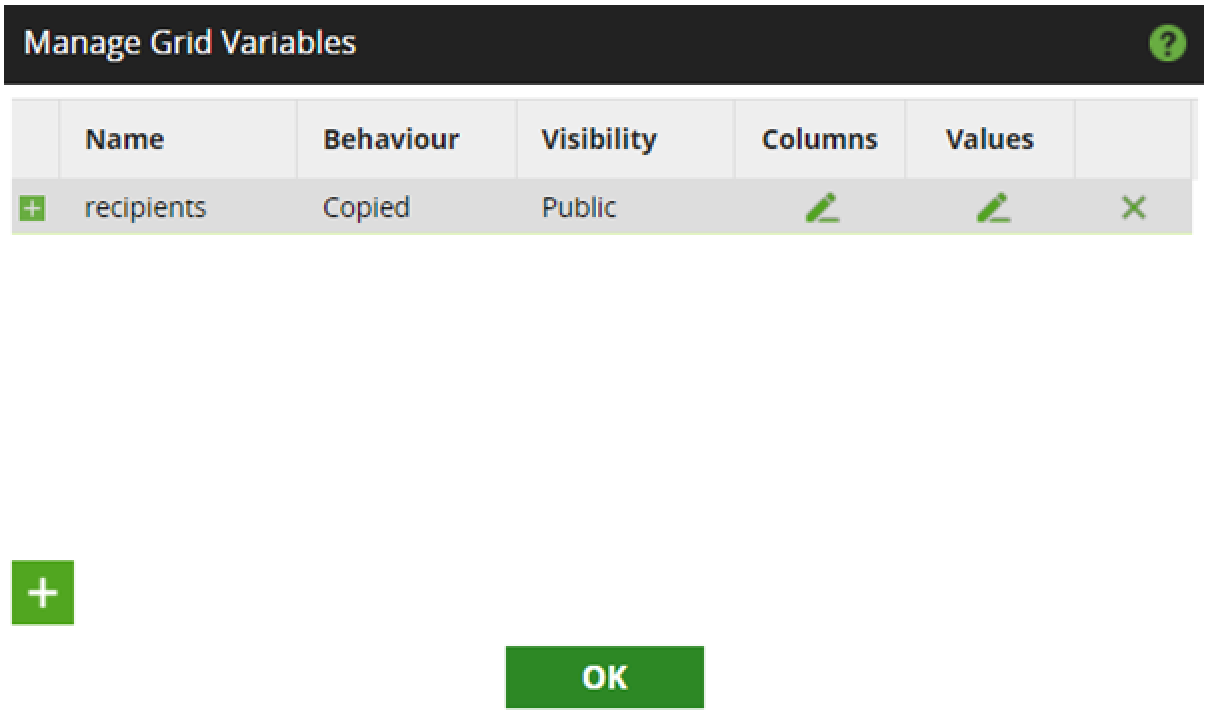
Creating grid variables

1. Right-click a job and click **Variables** → **Manage Grid Variables** to open a dialog listing all grid variables for that particular job. You can also right-click on the job canvas and then click **Manage Grid Variables**.

2. In the **Manage Grid Variables** dialog you can perform the following actions:

- Click at the bottom-left of the dialog to create a new grid variable.
- Click to the left of a variable name to display the variable's **Description** property.
- Click under the **Columns** heading to the right of the variable to edit the variable's properties and columns.
- Click under the **Values** heading to the right of the variable to edit the variable's default values.

- Click to the right of the variable to delete the variable. This cannot be undone, so be sure you want to do this before clicking **Yes** in the confirmation dialog.

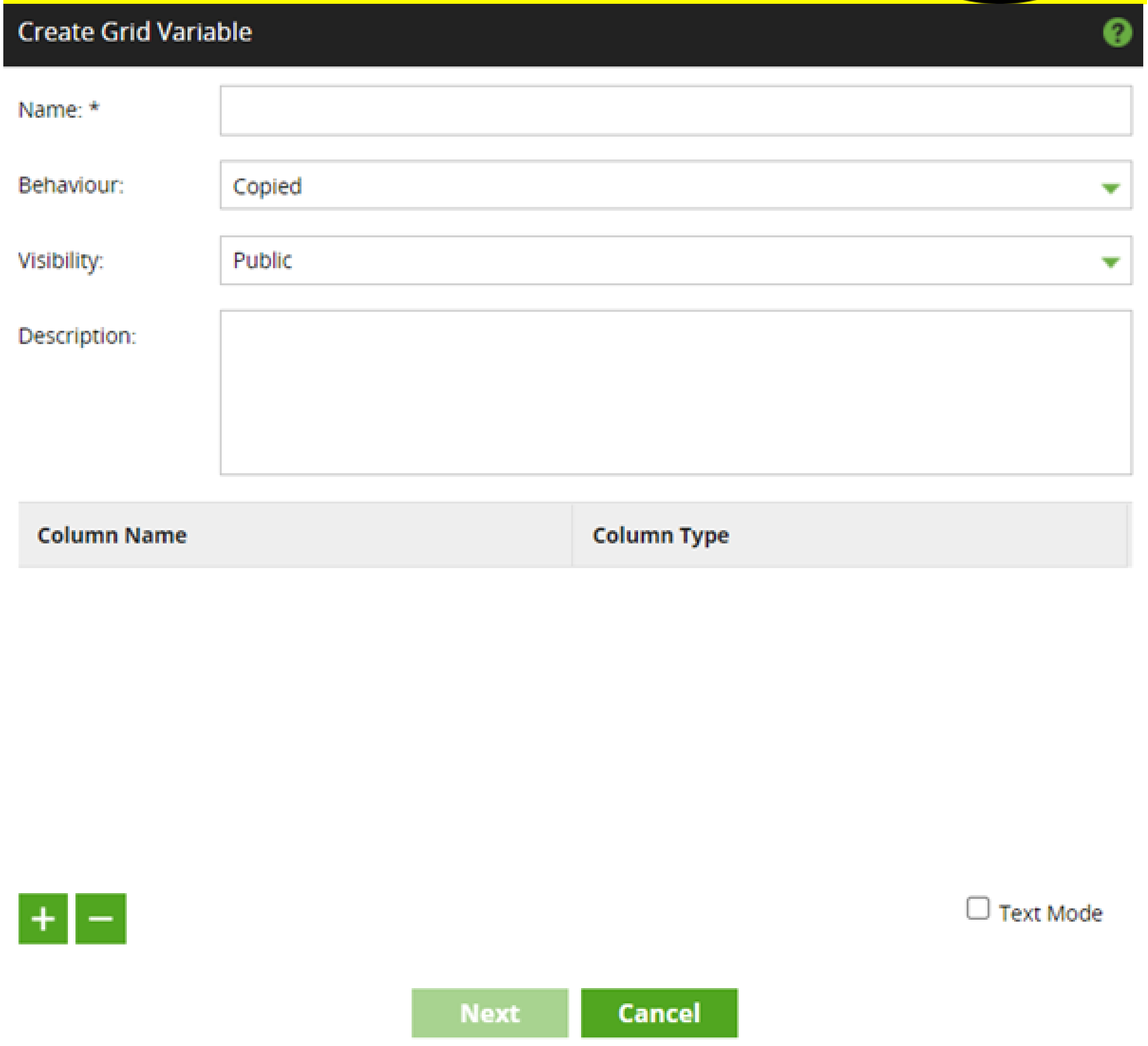


Manage Grid Variables

	Name	Behaviour	Visibility	Columns	Values	
+	recipients	Copied	Public	✎	✎	✕

3. Clicking to add a grid variable will open the **Create Grid Variable** wizard. (Note: editing an existing grid variable will open the same wizard but with the title **Update Grid Variable**.) Provide the following details:

- **Name:** A name to identify the variable in other dialogs and lists.
- **Behaviour:** Determines the variable's branch behavior inside a job. That is, how the variable is updated when more than a single job branch is making use of it. For more information on variable behavior, read this article.
- **Visibility:** Select **Public** or **Private**. If Private, this variable cannot be discovered and overwritten when this job is called from a Run Orchestration or Run Transformation component.
- **Description:** A description of the grid variable. This description has no functionality beyond reminding you what the variable is for.



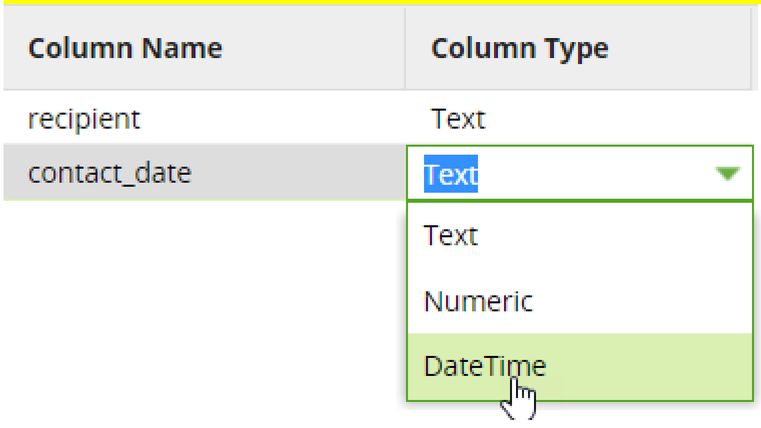
Create Grid Variable

Name: *	
Behaviour:	Copied
Visibility:	Public
Description:	

Column Name	Column Type

4. Below the grid variable properties is a list of columns contained by the grid. Each column has a **Name** and **Type**. To create a new column, click and enter the following details on the blank line created:

- **Column Name:** A name to identify the column. This must be unique within the grid variable but can be the same as column names defined in other grid variables. The name can only contain letters, numbers, underscores (_) and dollar symbols (**$**).
- **Column Type:** The data type that the column will contain. Select **Text**, **Numeric**, or **DateTime**. For more information on variable types, read [this article].



Column Name	Column Type
recipient	Text
contact_date	Text

To edit an existing column name or type, simply click on the value you want to change. To delete a column, click the column name to select it and then click .

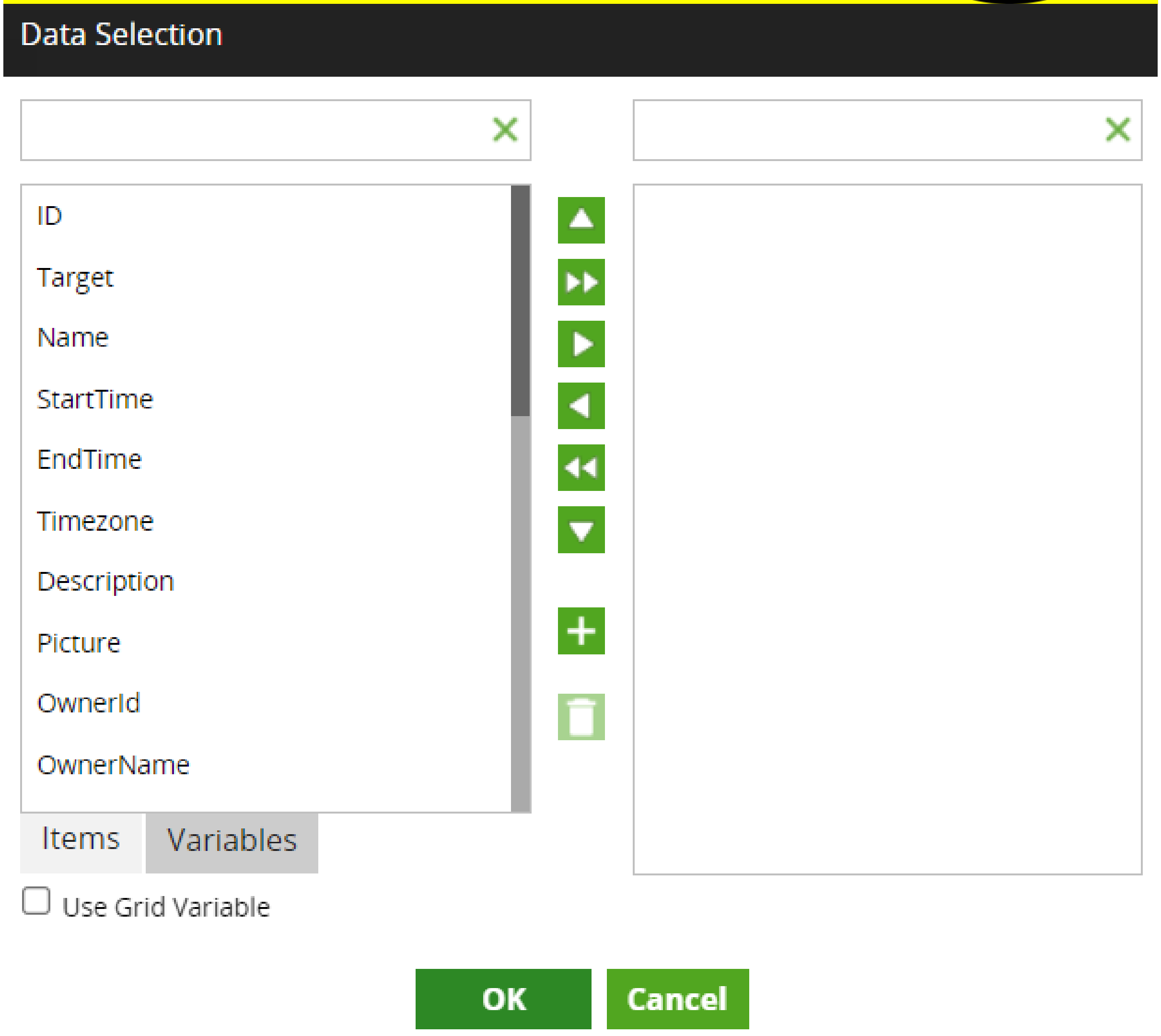
Click **Next** to continue to the second page of the wizard, where you can enter default values for the columns.

5. On the second page of the **Create Grid Variable** wizard, you can enter a default value for each column you have defined in the grid variable. This is not always required, as the grid variable can be populated in a variety of ways that do not require default values to be defined first (for example, using the Table Metadata to Grid component.

When you have finished entering default values, click **OK**.

Using to include columns from data sources.

Grid variables can be used to populate choices in multiple properties within Data Staging components. The **Data Selection** property, for example, which is used to select which columns are returned from a query:

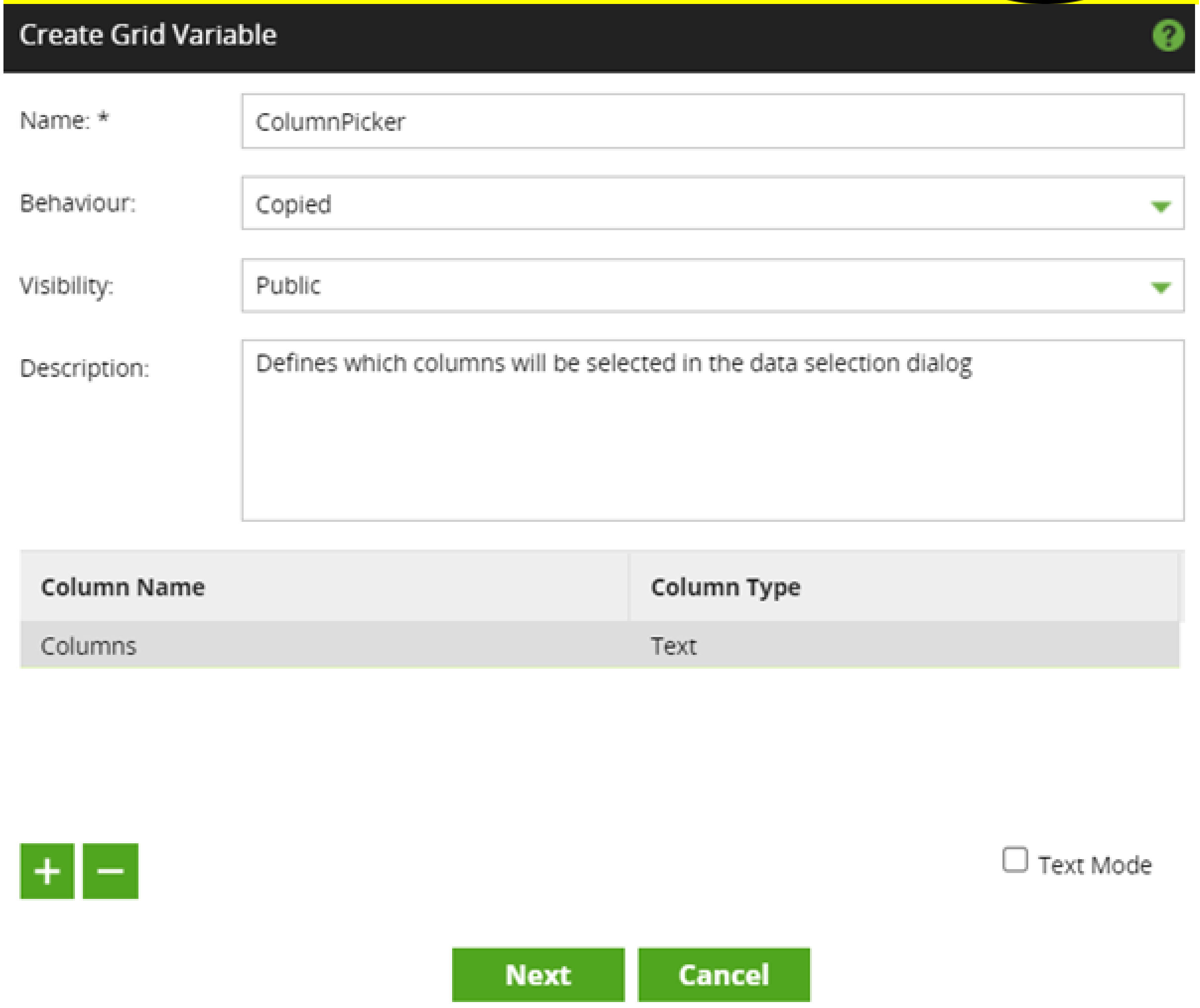

Data Selection

ID
Target
Name
StartTime
EndTime
Timezone
Description
Picture
OwnerId
OwnerName

If you select **Use Grid Variable** in this dialog, you can use a pre-defined grid variable to select the columns. You would do this as follows:

1. Create a new grid variable, as described above, which contains a single column named "Columns".

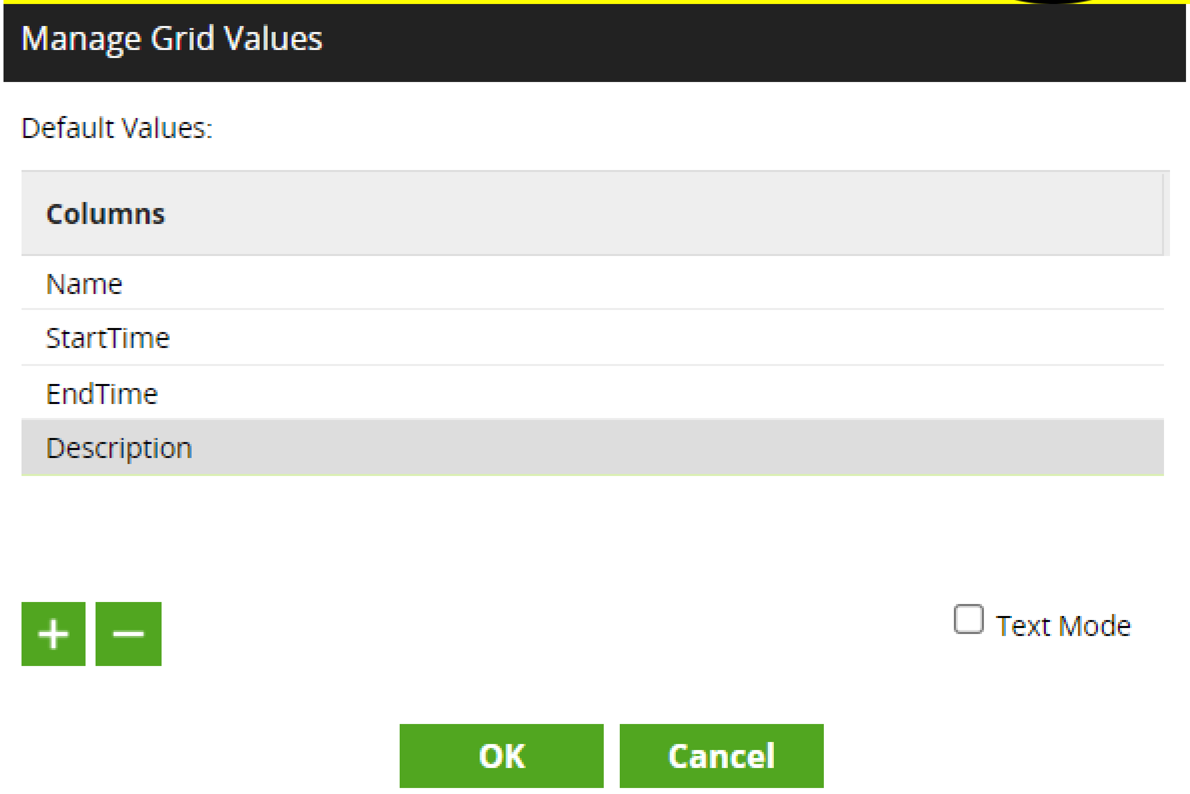
VARIABLES IN MATILLION


Create Grid Variable

Name: *	ColumnPicker
Behaviour:	Copied
Visibility:	Public
Description:	Defines which columns will be selected in the data selection dialog

Column Name	Column Type
Columns	Text

2. Populate the **Default Values** of the grid variable with the names of the columns you want to include in the **Data Selection** property.

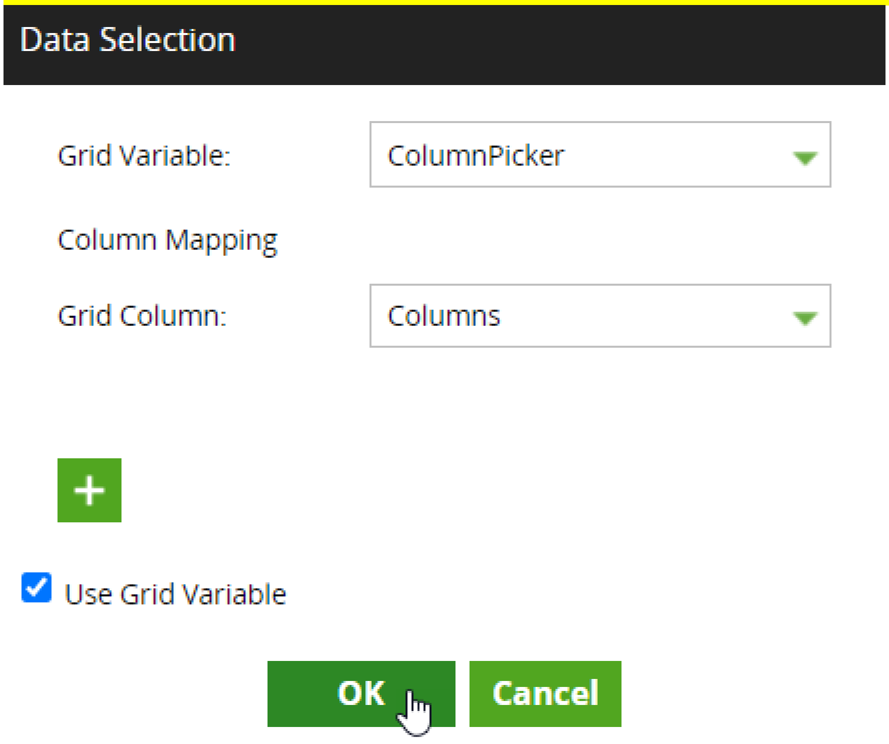


Manage Grid Values

Default Values:

Columns
Name
StartTime
EndTime
Description

3. In the **Data Selection** property of your job component, specify which grid variable and which column from the variable to use. The property will then be populated with the default values you entered in the grid variable. You can use the same grid variable across all components in the job where you need to specify the same data selection, making it easier to consistently set the property and allow you to make global changes to all such properties in the future.


Data Selection

Grid Variable: ColumnPicker

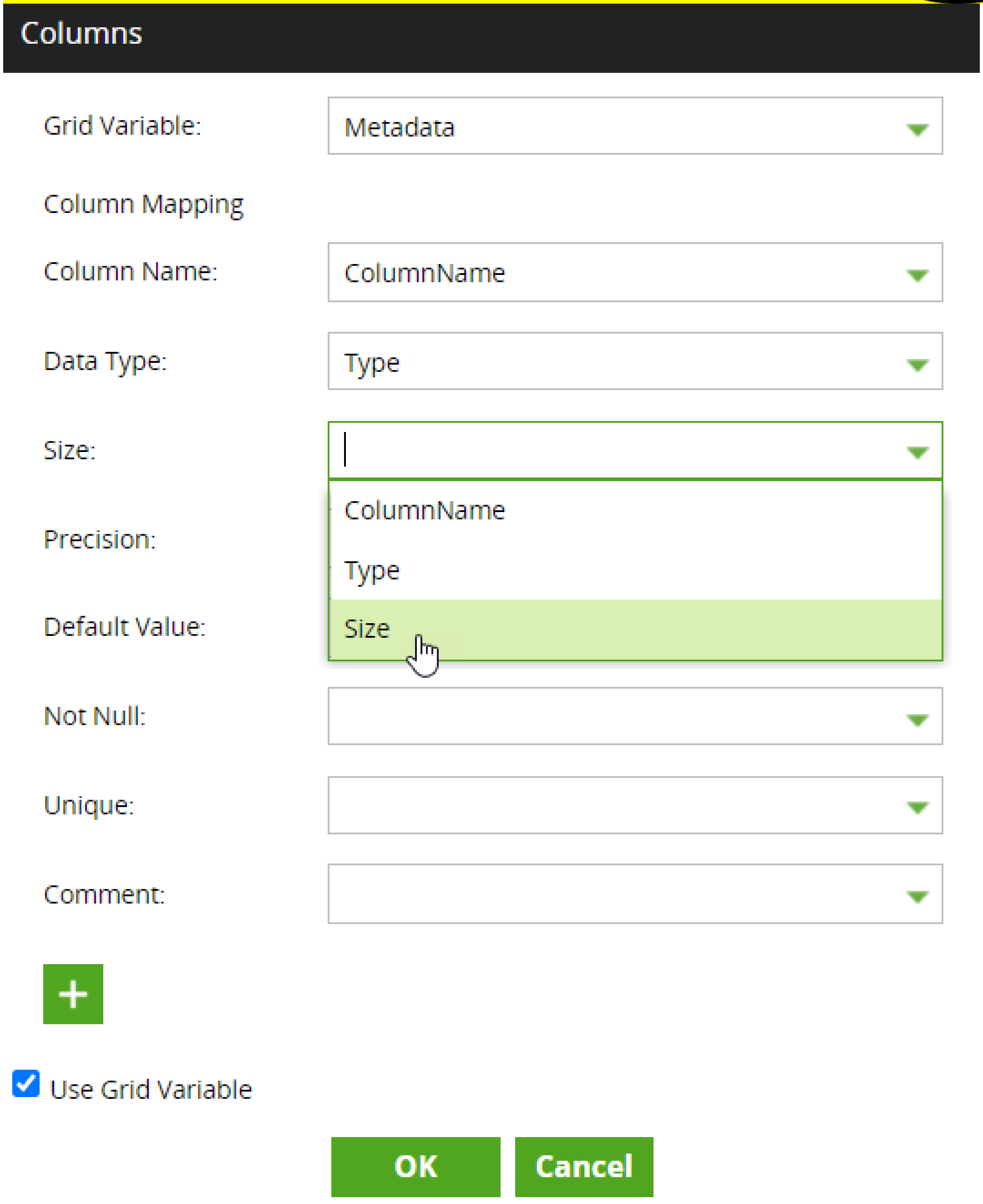
Column Mapping

Grid Column: Columns

☑ Use Grid Variable

Using to populate Metadata

In the Create Table and Create External Table components, table metadata can be assigned from a grid variable by selecting **Use Grid Variable**. You must first create a new grid variable, as described above, with columns that contain the data you want to populate the metadata with. You then specify which grid variable the property will use, and which columns in the grid variable map to each piece of metadata:



Columns

Grid Variable:	Metadata
Column Mapping	
Column Name:	ColumnName
Data Type:	Type
Size:	
Precision:	
Default Value:	
Not Null:	
Unique:	
Comment:	

☑ Use Grid Variable

Using to populate variables

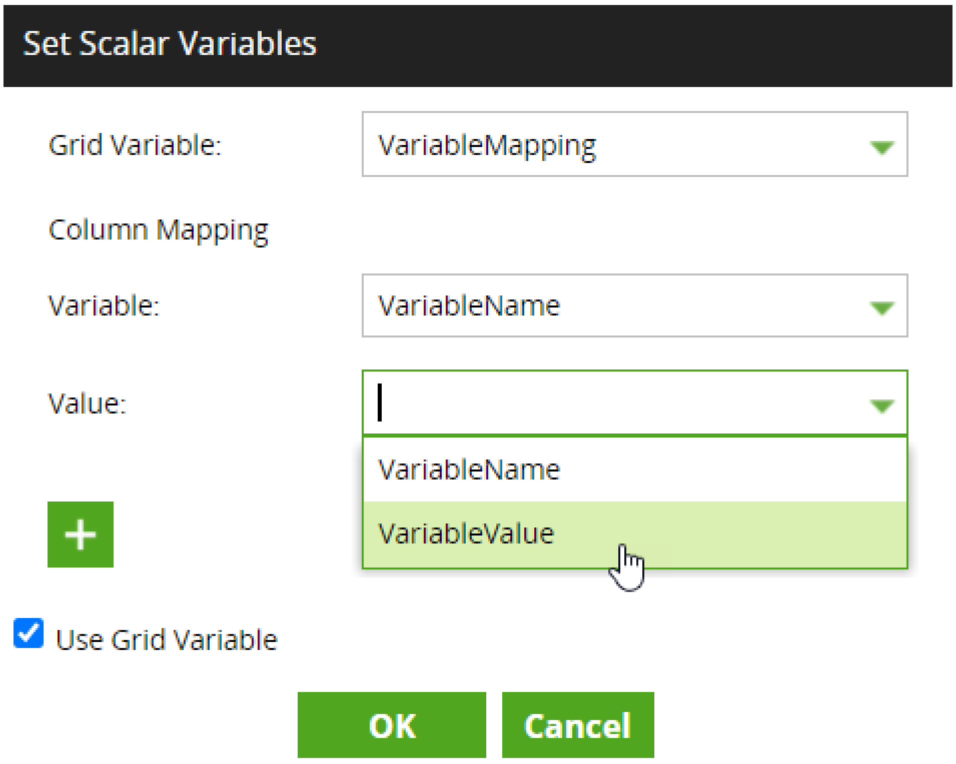
If you have a large number of variables to pass into a job along your production line, it can be convenient to make a grid variable that can then populate the variables for you.

In the Run Transformation and Run Orchestration components, the **Set Scalar Variables** property is used to set variables used by the component. You can use a grid variable to assign these properties.

1. First create a new grid variable, as described above, with one column that contains variable names and another that contains the values you want to put into each variable.

2. In the **Set Scalar Variables** dialog of your Run Transformation or Run Orchestration component, select **Use Grid Variable**.

3. Select the grid variable you have defined and specify which of its columns contains the variable names and which contains the values.


Set Scalar Variables

Grid Variable: VariableMapping

Column Mapping

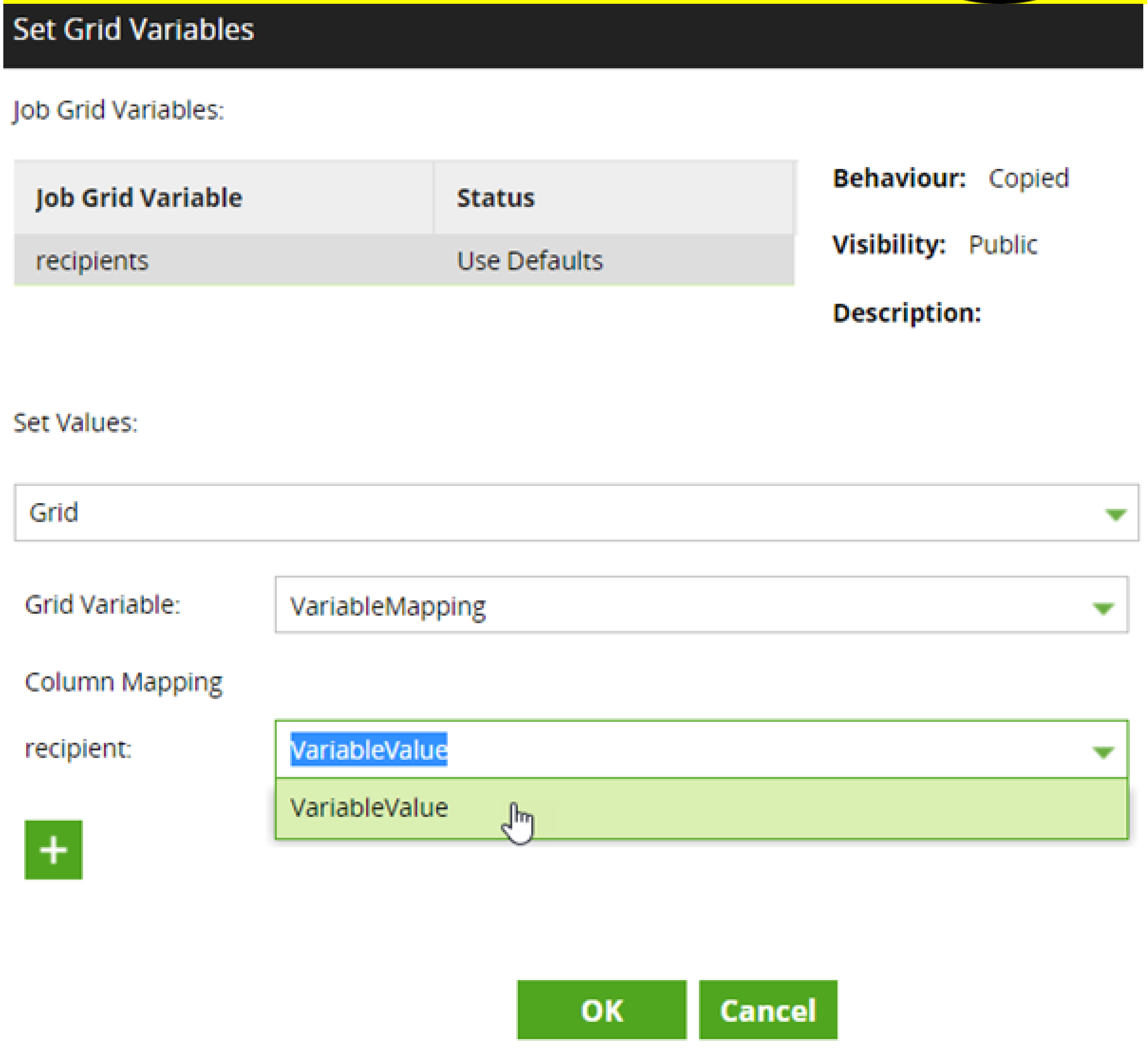
Variable: VariableName

Value:

Using to pass grids

Similar to passing regular variables, grid variables can also be passed on to jobs that use the Run Orchestration and Run Transformation components, using the **Set Grid Variables** property. The orchestration or transformation job that you are running with the component must have a grid variable defined in it for this operation to succeed.

To pass the grid variable, in the **Set Grid Variables** dialog, select **Grid** from the **Set Values** drop-down. This will expand the dialog to give you options to select the grid variable you want to use and the grid variable columns you will use. In this way, grid columns can be mapped from the calling job to the called job.



Set Grid Variables

Job Grid Variables:

Job Grid Variable	Status
recipients	Use Defaults

Behaviour: Copied

Visibility: Public

Description:

Set Values: Grid

Grid Variable: VariableMapping

Column Mapping

recipient: VariableValue

Using grid variables in Python

Warning

Variables become first class variables in Python and Bash scripts, and care should be taken to avoid naming them in a manner that clashes with key words in either language. We recommend using a prefix (for example, v_) to ensure no such conflicts occur.

VARIABLES IN MATILLION

It's possible to access and update grid variables in Python scripts. To get grid variable data and load it into a Python array, use:

```
context.getGridVariable('<GridName>')
```

To place Python array data into a grid variable use:

```
context.updateGridVariable('<GridName>', <values>)
```

The following is an example Python 3 script that takes data from one grid variable, "people", puts the data into an array, and it the variable before updating a different grid variable, "names", using that array.

```
array = context.getGridVariable('people')
for data in array:
  print(data)
context.updateGridVariable('names', array)
```