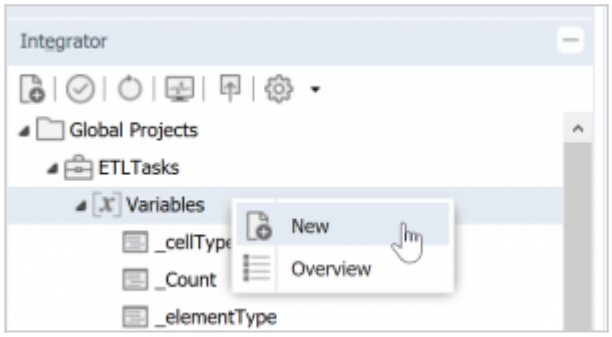
# Integrator Variables

Job execution can be parameterized with variables. For example, you may want to import data for particular years independent of each other, such as all data for the year 2016 and then all monthly data for the year 2017. Without the variable concept, you would have to create extracts, transforms, loads, and jobs separately for each year, or edit these manually each time. With the variable concept, however, only an extract that supports the year as a variable is necessary. To implement the Jedox Integrator process for a particular year, the variable for the job is changed, or a separate job is defined.

Variables can be used anywhere during the modeling, such as in the SQL statement of a relational extract or in the filter of a cube extract.

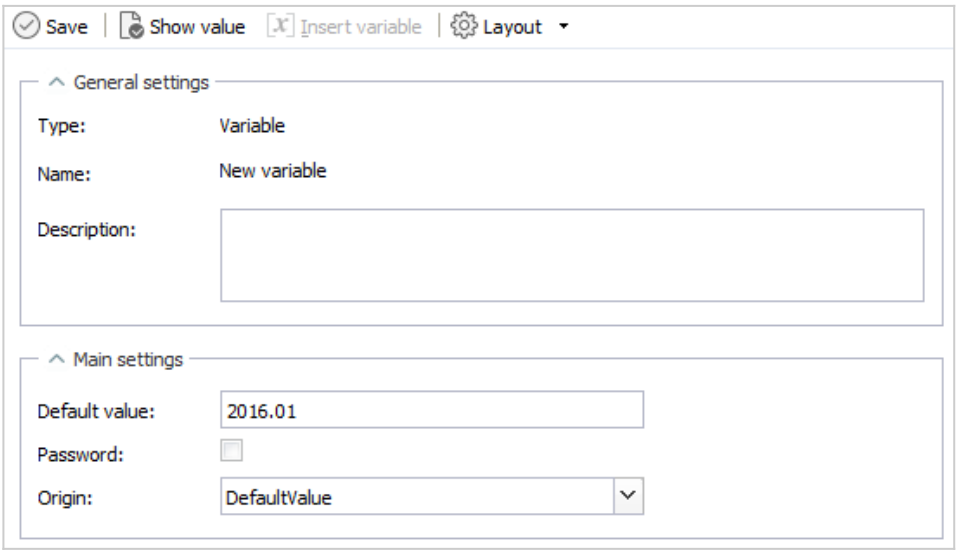
## Defining variables

To create a new variable, look in the Navigation pane under the project name, right-click on Variables, and select New (see screenshot  below). You will be prompted to give the variable a name. Click OK.




Now you can enter a description and a default value for the new

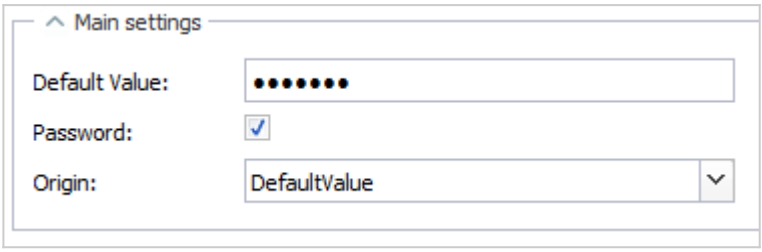
variable.




The variable is defined for the project globally with a name and a Default Value. The variable will be replaced with its current value during the execution of the job.

Variables are initialized as defined by their origin (DefaultValue, Setting, or Groovy) or the Default value.

Selecting the Password checkbox encrypts the Default Value, which can be useful when the value is a password or other sensitive value.

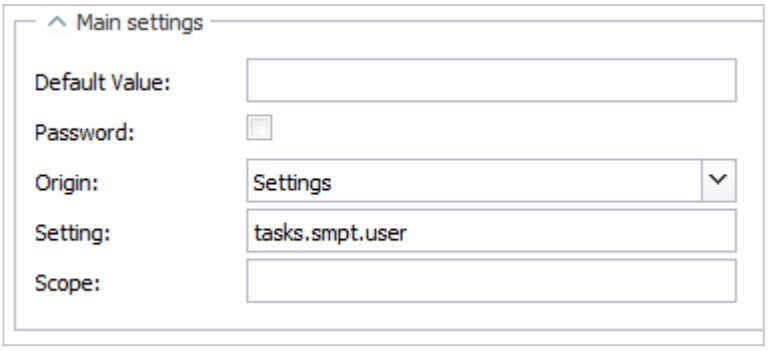


### DefaultValue origin

If the origin is set to DefaultValue, then the variable is always initialized with the Default Value.

### Setting origin

Variables can be initialized by a value from the Jedox Web settings.




The Setting field specifies the folders and key of the setting, separated with a dot, e.g. `tasks.smpt.user`.

The settings are organized into scopes. A setting could be in the Global scope, or it could be in a particular Model. The Scope field specifies the location of the setting. It could be empty, Global, or the fully qualified name of a Model (e.g. `com.jedox.model.profitandloss`). The table below shows the possible combinations.

| Setting is Global | Setting is in the current Model | Setting is in another Model |
|---|---|---|

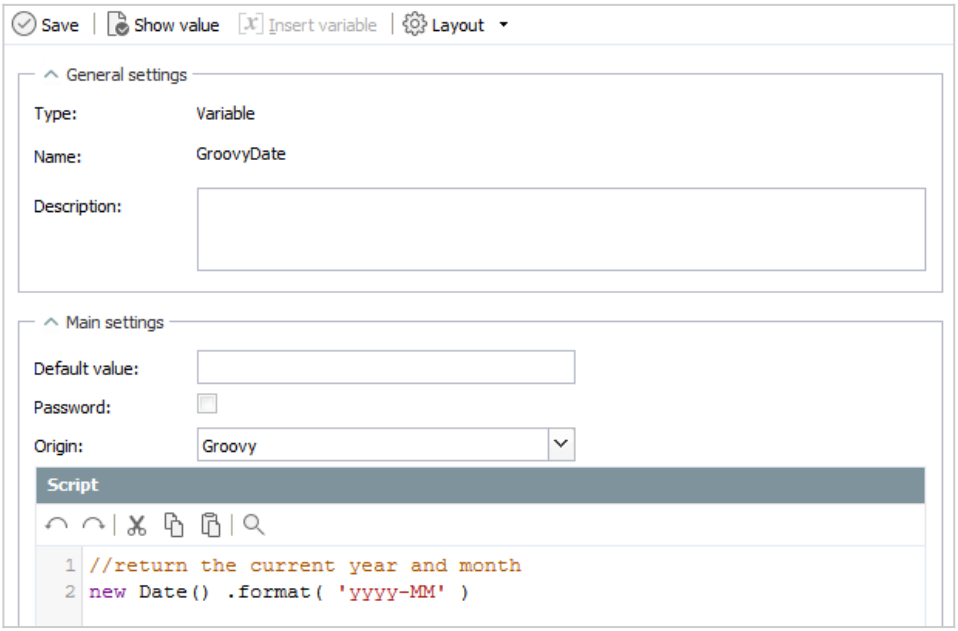
| | | | |
|---|---|---|---|
| **Integrator project is Global** | Scope = “” | N/A | Scope = “com.jedox.model.profitandloss” (example) |
| **Integrator project is in a Model** | Scope = “Global” | Scope = “” | Scope = “com.jedox.model.profitandloss” (example) |

The Default Value field can be left blank, or it can be populated with a fallback value that will be used if the Setting is not found.

### Groovy origin

This option uses a Groovy script to initialize the variable. This option could be used to retrieve the current date, for example.

Selecting Groovy opens a script editor, as shown below:




The Default Value field can be left blank, or it can be populated with a fallback value that will be used if the script returns no result.

Starting with 2019.1, Groovy Origin variables can reference values from a Global Connection, such as Jedox Cube or relational connection. This is useful if you need to retrieve stored connection variables, such as tokens, when you connect via REST.

**Note:** You cannot pass an Integrator variable to a Groovy scripted variable.

Example:

[crayon-5dce3ef8a0038110209961/]

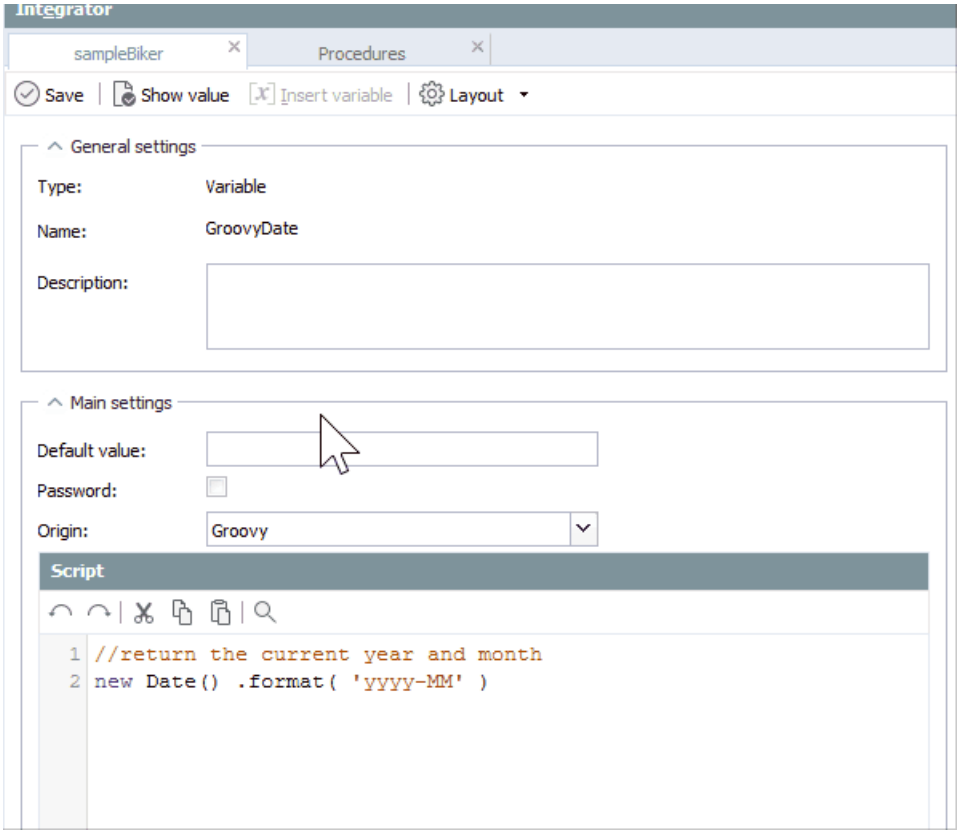
You can test the call by running the following Groovy job:

To test, you can run the following Groovy job (replace ‘test’ with the name of your variable)

[crayon-5dce3ef8a003f651854084/]

## Show value button

As of Jedox 2019.2, you can inspect the current value of a variable by clicking **Show value** in the toolbar, as shown below. This option is useful for variables sourced from the Settings or a Groovy script.




## Changing variables in Integrator processes

You can also set the value of a variable for a Jedox Integrator job that has been started via a web service call or a batch file. The command line would then look like this:

```
etlclient.bat -s localhost -p Biker default -c Year=2007
```

Variables can be changed in a variety of ways according to an order of precedence, as outlined below:

1. Individual values that are set in Jedox Integrator Script Job

or Function (type Groovy or JavaScript) with API.setProperty method, or via loop sources in TableLoop Transform and Loop Job.

2. Variables set in the surrounding job of a sub-job.
3. SOAP interface of the Jedox Integrator with `addExecution` and `runExecution` methods, e.g.
   • from Jedox Web via PHP script in the Macro Engine.
   • Context variables in the command line client with option `-c`
4. Fixed values of variables set in a External Job definition.
5. Fixed values of variables set in a Standard Job definition.
6. Initial values of Settings or Groovy origin in the variable definition of the Jedox Integrator project.
7. Default values in the variable definition of the Jedox Integrator project.

An example of how variables are used in the Jedox Integrator process can be found in the sample Jedox Integrator projects “sampleVariables” and “sampleLoopJob”.

## Escaping of variable definitions

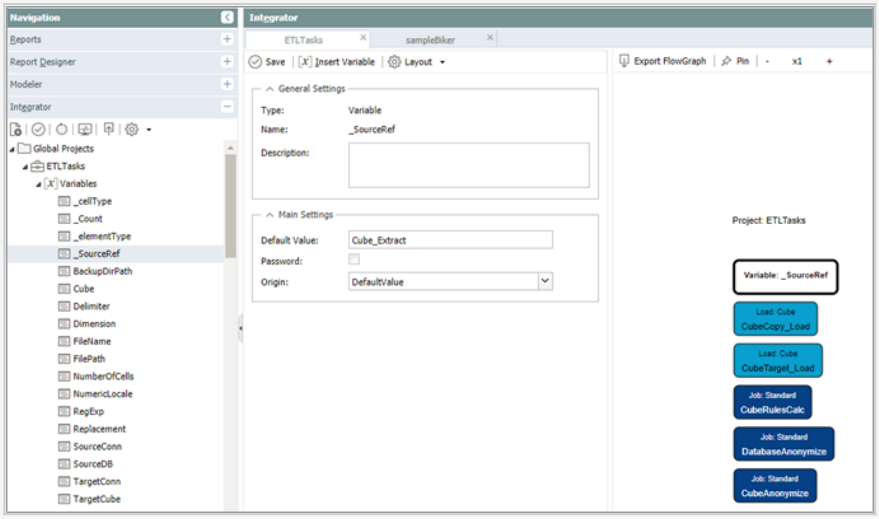
Integrator variables, which are expressed in the form **${*varname*}**, can now be escaped, or treated as literal characters, by using the form **$${{*varname*}}**. For example, the expression $${{xy}} would be

interpreted literally as ${xy}.

This feature can be especially useful in rule templates that are created by an Integrator job.

## Variable tracking

The Flowgraph displays where a specific variable is used in a project:




After selecting the variable “_SourceRef”, the flow graph displays all modules that use this variable.

**Note:** the use of variables to reference other components has some limitations:

- The flow graph is shown based on default variable values only. Other dependencies during runtime are not

visualized.

- Value help on columns for the Target input field is not fully functional for variable values.