Critical manufacturing
an ASM PT company

Connect IoT - Control Flow Configuration Tutorial

11.2

February 2026

DOCUMENT ACCESS

Public

Connect IoT - Control Flow Configuration Tutorial

The goal of this tutorial is to showcase the use of the control flow designer in constructing integrations. It will use a file raw integration. The goal of the integration will be whenever a new file is created with a content of Test followed by a number, the integration will create a number of files equal to the number specified in the content with the file name being the number and the content being success. If the content does not contain the keyword Test, log saying file ignored.

> **Note**
>
> During this tutorial, the **Automation Manager** will run in console mode in order to highlight the most important events as they take place.

Automation Protocol

The **Automation Protocol** will be quite simple, select the package `@criticalmanufacturing/connect-iot-driver-fileraw` in order to use the file raw protocol. Use the default configurations.

> **Note**
>
> Note that to do this in a real life context, it is important to have a general knowledge about the protocol, the equipment itself and its related documentation.

Automation Driver Definition

The **Automation Driver Definition** will use the templates for the File Raw, as such it won't require any other configuration, besides depending on the previously created File Raw Protocol.

> **Note**
>
> Note that to do this in a real life context, it is important to have a general knowledge about the protocol, the equipment itself and its related documentation.

Automation Controller

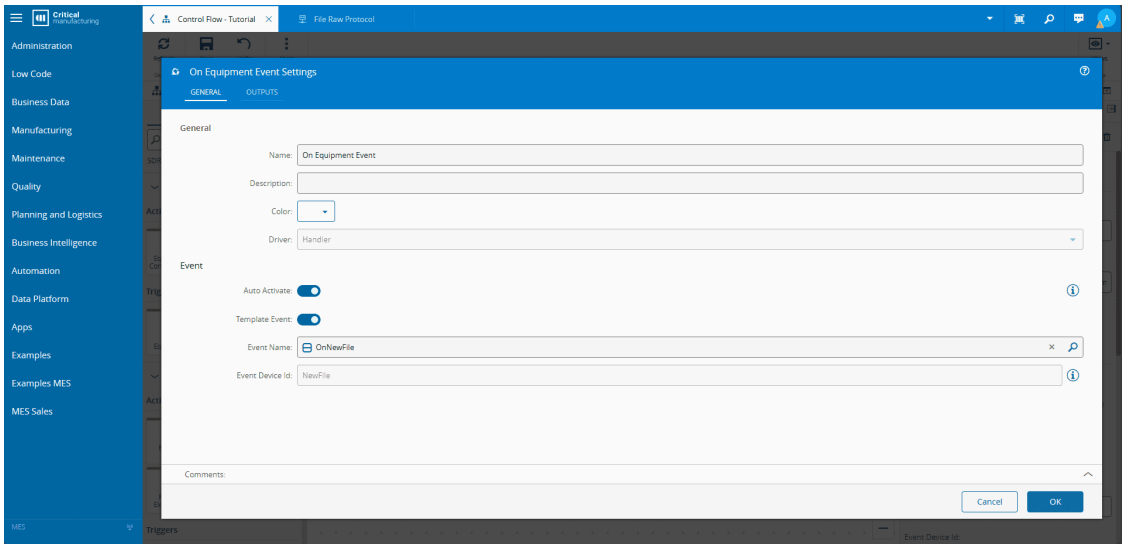
The **Automation Controller** will be of `Workflow Type` **Control Flow**. In the tutorial the entity type used was a of type `Site`. Use the `File Raw Driver Definition` previously created. In the `Tasks`, select also the `File Driver Tasks`.

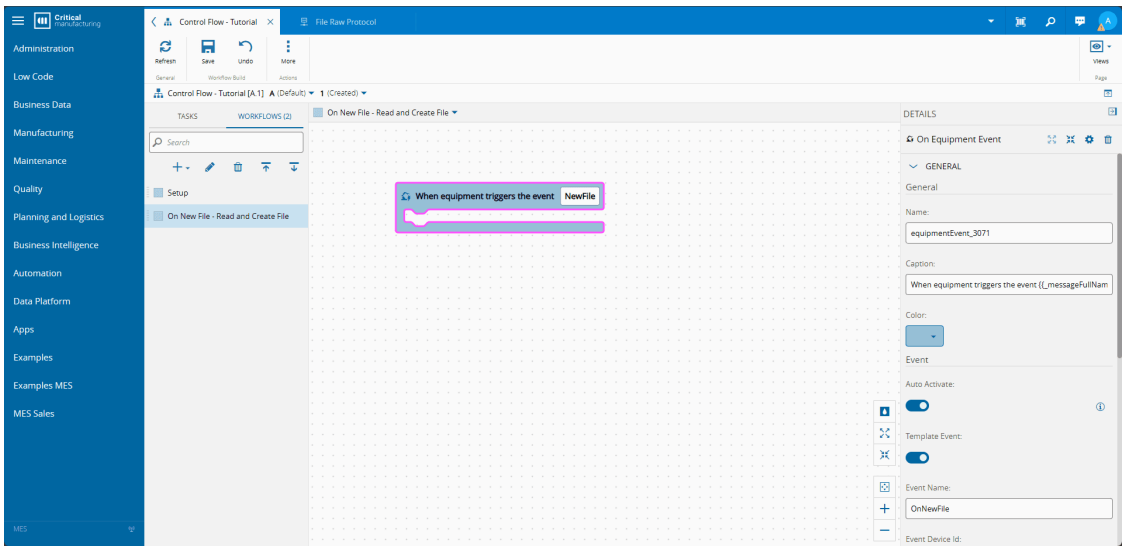
Upon creating the Controller, the GUI will offer a default template for the driver created. This template will have the start cycle of the driver. In the Task `Equipment Configuration`, edit the `path` and provide the path for the watcher. For this tutorial the used path was `c:/temp/Tutorial`.

1. In the page left panel select `Workflows` and then +, select `Control Flow`. Edit the page and rename the page to a more friendly name, for example `On New File - Read and Create File`.
2. Drag and drop the following tasks:
   - `On Equipment Event`: to listen to the Event "OnNewFile", to be notified on a new file

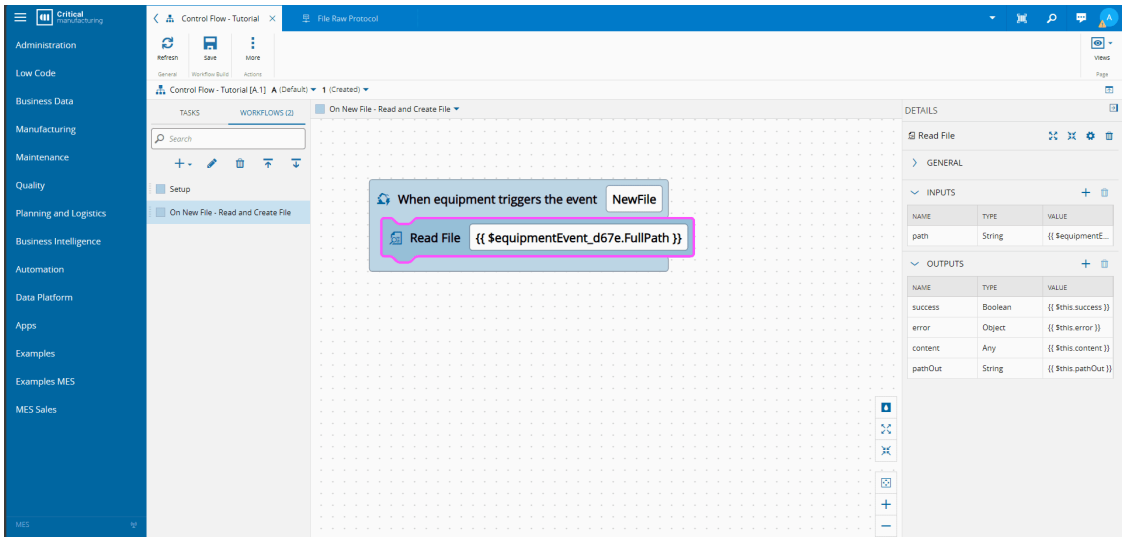
- `Read File`: to read the file that was created
- `Condition`: to perform conditional operations
- `Create File`: to create a new file
- `Log Message`: to print the message into the console, in case the Temperature evaluation result is true

3. Go to the `On Equipment Event` settings and for the Equipment Event, select the Event OnNewFile.





4. Add the Read File Task as the next action. In Control Flow, all the context is available. You can either retrieve it from the general scope, or from a particular `Task`. In this case, the Read Task will require the full path of the New File detected.



Note

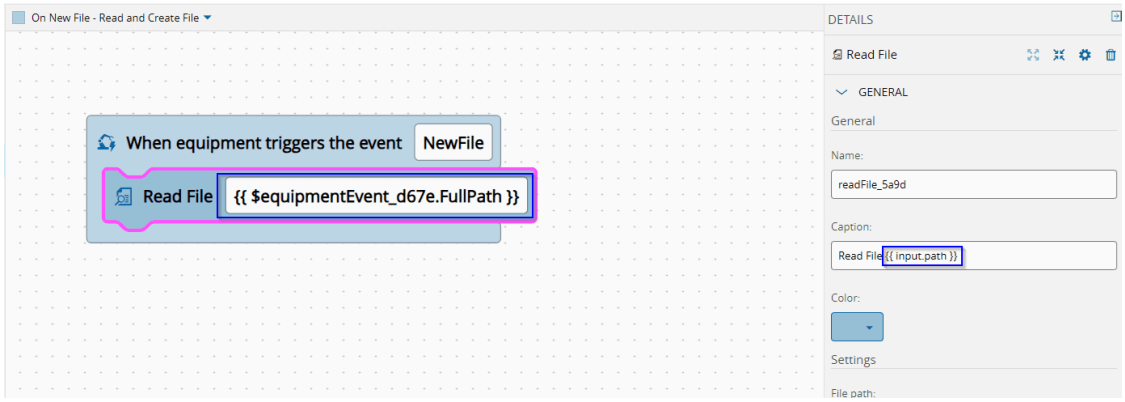
In `Control Flow` items in scope can be used using double curly braces and with dollar scope or dollar name of task, i.e {{\(scope.Test.Name}} or {{\)task_1.Test.Name}}. If the token is in the context of the task, as a setting or input, the token can be accessed directly for a setting or using `input.` for an input i.e {{path}} or {{input.path}}.

Important

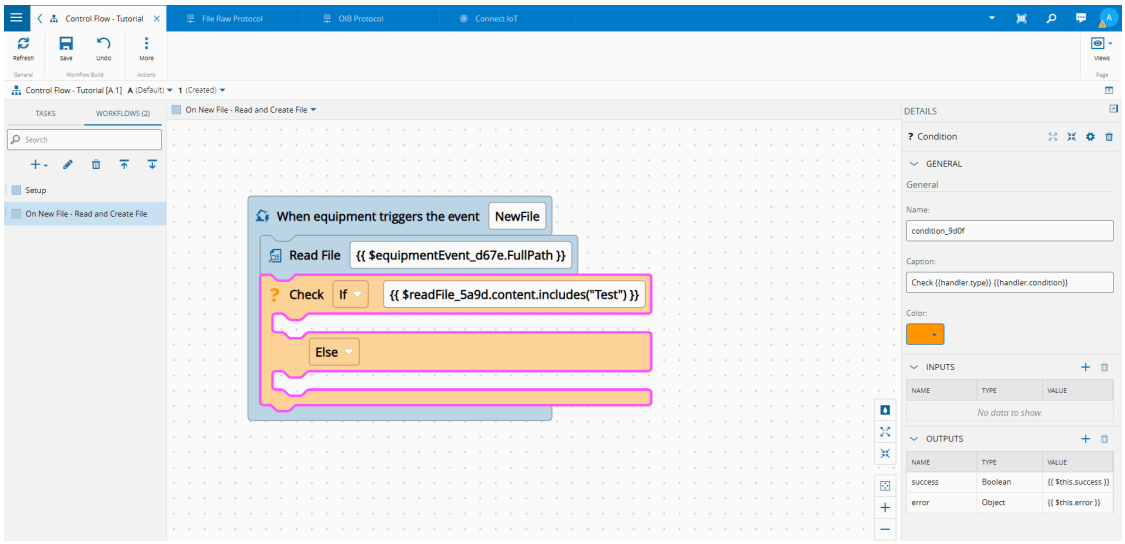
Only inputs and outputs are tokenizable. Currently, settings may not contain tokens, unless specific tasks mention that they can.

Note

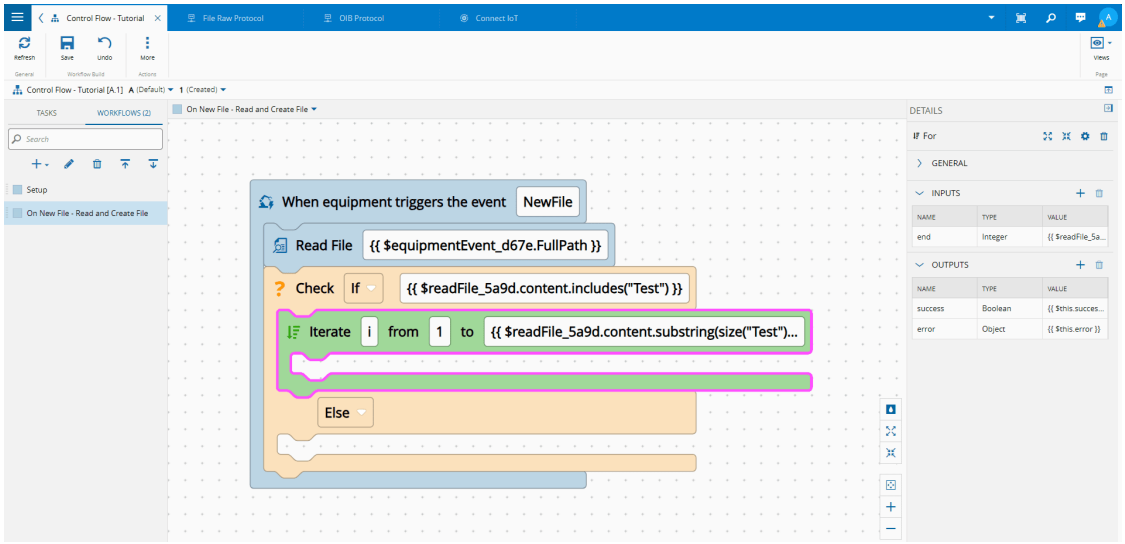
Notice the caption is rendering the content of the input, even though it is described as {{ input.path }}.



5. Add a condition task. If the read task content includes a value `Test`, it will be to process.

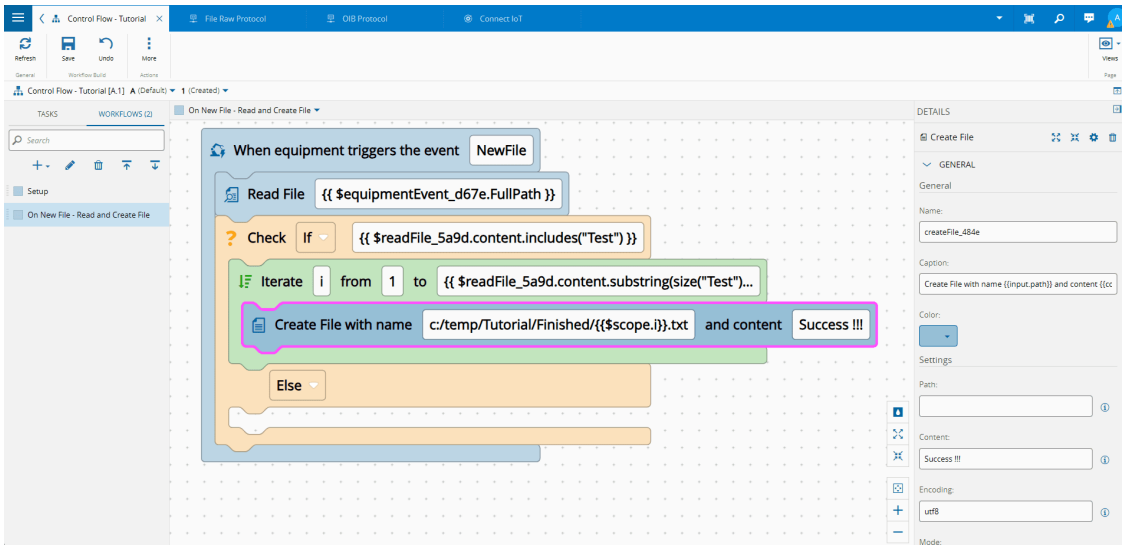


6. In the `If` condition add an iterator task. This task will iterate through the content of the file after the keyword `Test`. To use tokens, we will add an input `end`, with the token for the content of the read file task after the `Test`. The expression used will be `{{ $readFile_5a9d.content.substring(size("Test")[0]) }}`.

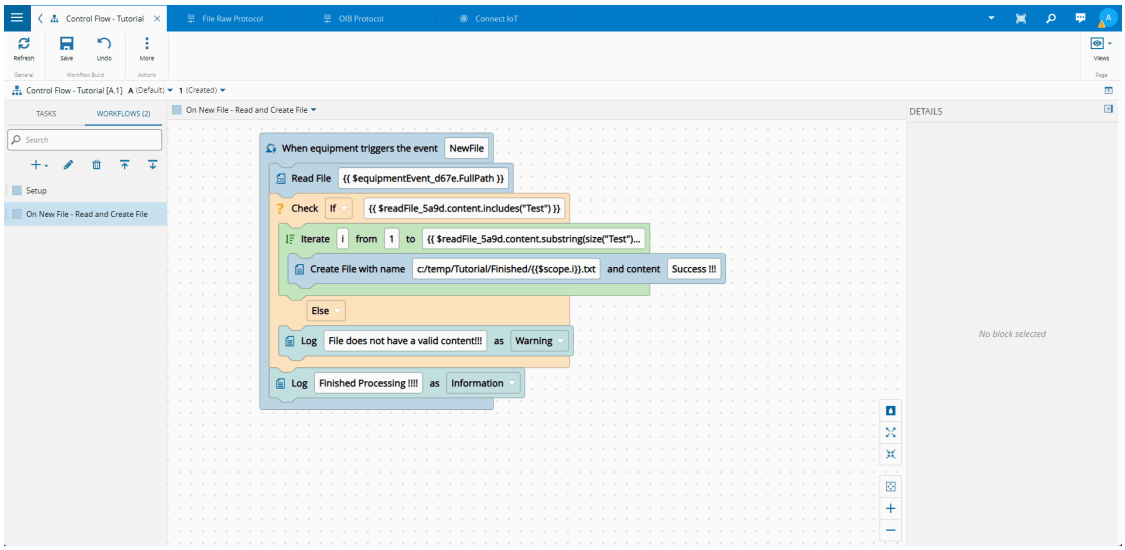


> **Note**
>
> In `Control Flow` token expressions allow for the use of more transformations. To apply transformation the engine applied is the mathjs engine. For indexes, the 0 starting point will be used.

7. In the `Iterator` task we can now add the `Create File`. The goal will be to create a file with name of the index and with content `Success !!!`. This file will be saved in the path `c:/temp/Tutorial/Finished`.



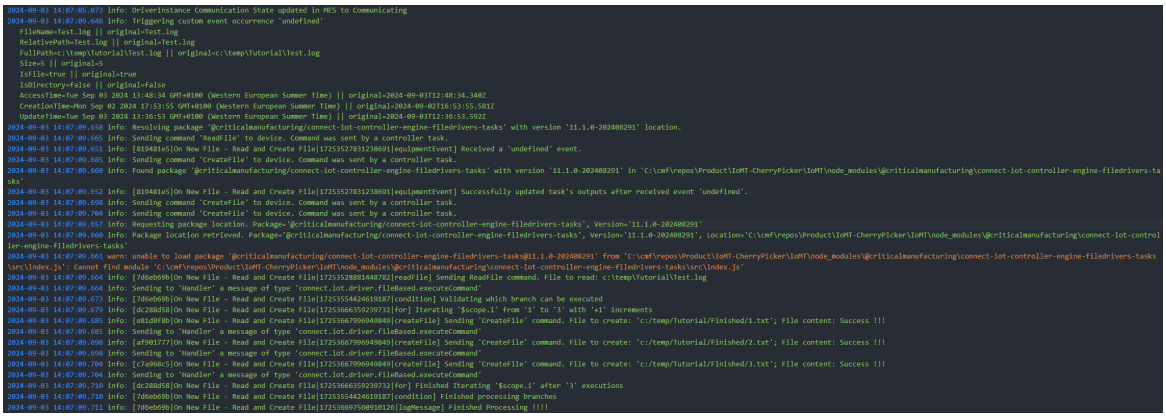
8. To finish our integration let us add a log message both for the else condition and for the end of the execution.



Execution

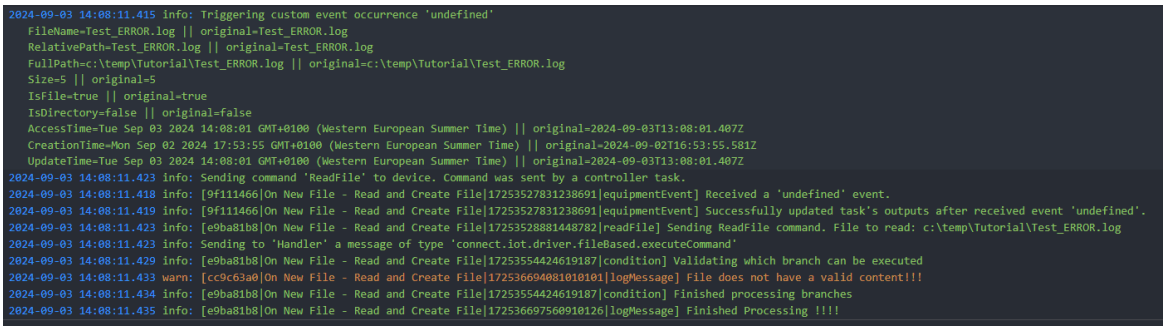
Let`s now execute a run of our integration.

Drop a file with content `Test3` in the c:/temp/Tutorial folder:



Notice that we see a new file being detected, the iterator being activated and creating three new files.

If we drop an invalid file:



It will not create any file and will log a warning notifying the user that it was not processed.

This tutorial tries to illustrate a simple use case for the use of control flow and how dynamic information can be propagated.

Legal Information

Disclaimer

The information contained in this document represents the current view of Critical Manufacturing on the issues discussed as of the date of publication. Because Critical Manufacturing must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Critical Manufacturing, and Critical Manufacturing cannot guarantee the accuracy of any information presented after the date of publication. This document is for informational purposes only.

Critical Manufacturing makes no warranties, express, implied or statutory, as to the information herein contained.

Confidentiality Notice

All materials and information included herein are being provided by Critical Manufacturing to its Customer solely for Customer internal use for its business purposes. Critical Manufacturing retains all rights, titles, interests in and copyrights to the materials and information herein. The materials and information contained herein constitute confidential information of Critical Manufacturing and the Customer must not disclose or transfer by any means any of these materials or information, whether total or partial, to any third party without the prior explicit consent by Critical Manufacturing.

Copyright Information

All title and copyrights in and to the Software (including but not limited to any source code, binaries, designs, specifications, models, documents, layouts, images, photographs, animations, video, audio, music, text incorporated into the Software), the accompanying printed materials, and any copies of the Software, and any trademarks or service marks of Critical Manufacturing are owned by Critical Manufacturing unless explicitly stated otherwise. All title and intellectual property rights in and to the content that may be accessed through use of the Software is the property of the respective content owner and is protected by applicable copyright or other intellectual property laws and treaties.

Trademark Information

Critical Manufacturing is a registered trademark of Critical Manufacturing.

All other trademarks are property of their respective owners.