Finding Errors

Track down common problems in your flow

Are your variables empty or not showing the values that you expected?
Is Ultradox not generating your documents due to errors in your templates?
This guide described techniques how to find the root cause of them problem so that you can fix them.

Error types

In Ultradox you may face two different categories of errors:

Errors in your flow

The building blocks in your flow rely on each other. They also require the proper input to work correctly.
If you get errors when a building block gets executed or they produce unexpected results, please read the Logging and Debugging sections.

Errors in the template

If you get errors while generating documents from your templates, please check out the Template Errors guide.

Logging

The easiest way to check the value of a variable and to resolve formatting issues is to just print the variable in a comment.
You can see all executions of your flow in the Ultradox editor, even if your flow is triggered from a Google Form or via API.
Adding comments will greatly help you not only to structure your flow, but also to log data to the console to check if everything works as expected.
Let's say you have a variable called `email` that you are receiving from a form and you want to know the entered value, just add a comment building block and enter `\$\{email}`.

> **The entered email is \$\{email}**
> fs-core icon-terminal
> http://help.ultradox.com/en/blocks.html#comment,Jump to the Comment building block

Whenever the flow is getting executed the email will be printed to the console.

> All variables in comments are evaluated just like in your document templates.

You will see something like this in your console:

```
The entered email is user12345@gmail.com
```

As you can also use all rendering options in comments, you can also use this building block to test the proper formatting of text, dates or numbers.

Just create an empty flow and add a comment building block that prints the current date:

Today is \$\{now;date(op=EEEE)}
fs-core icon-terminal
http://help.ultradox.com/en/blocks.html#comment,Jump to the Comment building block

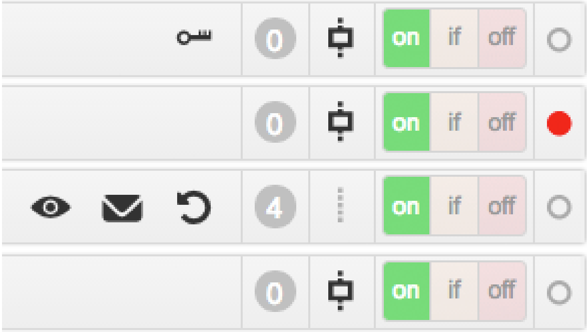
As the current date is formatted using the date renderer with the specified output pattern it will print the name of the current day to the console:

```
Today is Wednesday
```

Debugging

Ultradox comes with a fully featured debugger that will allow you to pause your flow at certain building blocks and to inspect all the available variables before resuming the execution.

You can not only debug flows started in the Ultradox editor, but also if they have been triggered by a timer, on Google Form submissions, by zapier, or via API.

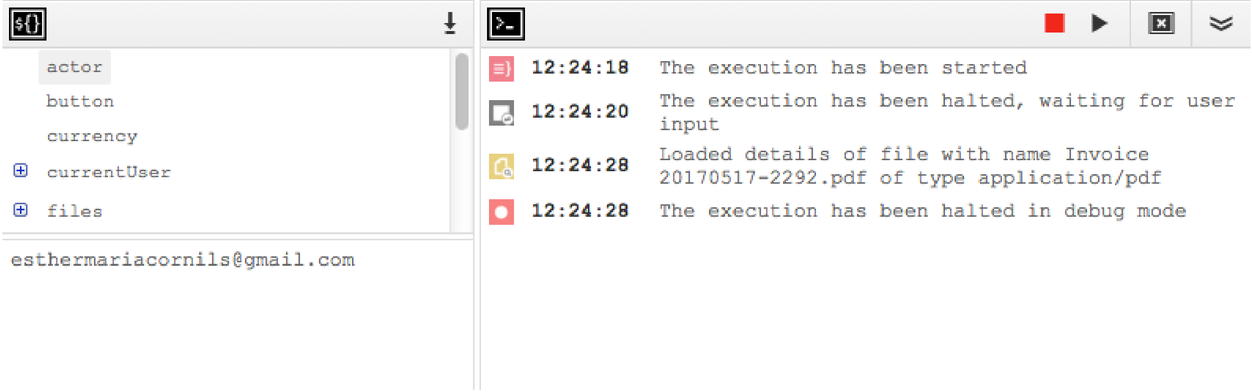


`icon:icon-breakePointOff` To debug your flow double-click on the breakpoint icon on the right of the building block where you want to start debugging.

The breakpoint icon will turn red which means that the next flow will be paused and can be inspected in the debug panel of the Ultradox editor.

The next flow that will run after activating the breakpoint will stop when reaching the building block with an active breakpoint.

The console will open and the debugger panel on the left will allow you to inspect the values of all variables.



```
esthermariacornils@gmail.com
```

```
12:24:18  The execution has been started
12:24:20  The execution has been halted, waiting for user
          input
12:24:28  Loaded details of file with name Invoice
          20170517-2292.pdf of type application/pdf
12:24:28  The execution has been halted in debug mode
```

Toolbar

You can then stop or resume the flow by using the buttons in the toolbar of the debugger panel.

`icon:icon-debugStep` The step button will continue the flow but only next building block will be executed and will then halt again in the debugger.

> Debugging your flow step by step is great to get an understanding of what is really going on as you can inspect all variables right after the execution of each block.

`icon:icon-stop` The stop button is an emergency stop to quit any running flows. This is a great way to end long running flows that may have been started by accident and are eating up your credits.

`icon:icon-play2` The resume button will continue with the execution of your flow and will run to the end or until it reaches another building block with an breakpoint.

`icon:icon-clearConsole` Clicking on the clear console icon will remove all messages from the console.

Template Errors

Detect and fix errors in your templates

With Ultradox you can generate sophisticated documents from your templates by using tables, nested conditions, loops and more.

As you have a lot of powerful features, you will also have a lot of ways to produce errors. If you get errors while generating documents from your templates, please check out the Template Errors guide.