Four Pees

Switch

App

FourPees

Buggy

Four Pees nv

Kleemburg 1
9050 Gentbrugge
Belgium
p +32 9 237 10 00
f +32 9 237 10 01
info@fourpees.com
www.fourpees.com

Table of Contents

1. Introduction

How much of your time making Switch flows is spent testing? Probably quite a lot if you're building more complex workflows. This is time-consuming for most flows, with the deactivating the flow, finding test jobs, dragging them on Switch, reactivating... And if your flow is more complex, and especially if it's part of a multi-flow setup, it becomes even more challenging, because often you can't test one flow in isolation as it will expect jobs to already have private data defined, or datasets attached, or email addresses present, and so on.

With all the projects we do where we use Switch, this is pain we felt acutely ourselves. Buggy was born from a set of internal scripts that aimed at alleviating that pain. It turns the "develop, test, correct, test, correct..." cycle into something manageable and reduces our stress levels. We thought we should share that outside of Four Pees as well.

Buggy is a bundle app. This means that you install it from the Enfocus Switch app store and in Switch you find three separate apps to use in your flows. All three apps are described in this documentation. Their basic setup is quite simple, but they have plenty of properties to adjust them to your needs in specific flows. Some would say we have gone a bit overboard perhaps, but obviously that's just wrong!

1.1. Versions

The following is a short version overview:

- version 1: initial version of the app.

2. Buggy Input

The Buggy Input app is designed to be used at the beginning of a workflow (though technically jobs can be injected using it at any point in the workflow as well). The app can be configured to inject jobs (files or folders) into the workflow and those jobs can be configured so that they match what the flow expects in terms of metadata.

The rest of this chapter describes the different functionalities and the app properties that can be used to configure them.

2.1. Injecting jobs

Buggy Input supports three different triggers to inject jobs into a flow, and the choice is made using the "Trigger on timer" and "Trigger by webhook" properties.

2.1.1. Trigger on timer

Using this property, you can ask Buggy Input to inject jobs into the flow based on a timer. The different property values are:

- **Flow start only**
  The configured jobs are injected when the flow is activated, and that's it.
- **Periodically**
  The configured jobs are injected when the flow is activated, and then based on a timer. The additional "Interval" property allows specifying the interval in seconds between the injection of further jobs. The default interval is 10 seconds.
- **Not automatically**
  When this is selected, no jobs are injected using a timer. Using the webhook is then the only way in get jobs into the flow.

2.1.2. Trigger by webhook

Using this property, you can switch on or off (by selecting "Yes" or "No" for the property's value), using a web hook to inject jobs into the flow. This allows you to use something like a Stream Deck appliance or even a Siri shortcut to trigger the injection of jobs.

The additional "Name" property lets you define the "name" of the webhook that the app will listen for. The default webhook name is "debug".

2.2. Disabling job injection

Buggy Input is designed for testing. Perhaps you have arrived at a stage where you feel testing is no longer necessary, but completely removing Buggy Input feels premature. In that case you have two ways to disable Buggy Input, without removing it from the flow completely.

- Use the "Set on hold" property. By default, it is set to "No" which allows Buggy Input to function. If you switch it to "Yes", Buggy Input can remain connected to the flow, but no jobs will ever be injected, regardless of how the other properties are set.
- Delete the connection between Buggy Input and your flow. Buggy Input is perfectly content to sit in your workflow without connections from it, and it's smart enough to disable all activity in that case.

If you trigger a webhook configured on it while the app has been disabled by either method, a debug message is logged to the Switch log so that you understand why nothing happens.

2.3. Job definition

The property "Inject" defines which jobs are injected into your Switch flow. It has three different values.

2.3.1. Dummy job

Selecting this option injects a single job each time the app is triggered. The job consists of a text file, that has the name "Dummy job.txt". The file's content (to make sure it's not a zero-byte file which sometimes could cause problems) is one line of text that says "Generated at: ", followed by a date and time stamp. The name or content of the dummy job cannot be modified.

2.3.2. One job

Selecting this option reveals one additional property called "File or folder" that allows selecting either a file or a folder. Whatever is selected is injected into the flow each time the app is triggered.

2.3.3. Many jobs

Selecting this option reveals two additional properties:

- **From**
  This property allows selecting a folder or specifying a list of files / folders. If it's a folder, the injected jobs are selected from the files or folders found in this folder. If it's a multi-line text editor, the injected jobs are expected to be specified in this dialog, one item on each line.
- **Number of jobs**
  This property specifies how many jobs will be injected. The jobs injected are the first jobs found in the "From" property.

2.4. Job property setup

To make sure your test job(s) function(s) correctly, Buggy Input allows setting all job properties, in a similar (but slightly different) way to SwitchScripter. The following properties exist:

Property name	Explanation
Set hierarchy	Hierarchy information, enter one hierarchy level per line.
Set private data	Private data, one private data definition per line. Each line consists of the name of the private data, followed by an equal sign, followed by the private data value.
Set datasets	Data sets, one dataset per line. Each line consists of the name of the dataset, followed by an equal sign, followed by the full path to the dataset you want to use. The type of dataset is derived from extension of the file. If the extension is not recognized, an opaque dataset is created.
Set status	The status of the job.
Set priority	The priority of the job.
Set email addresses	Any email addresses that should be associated with the job (for example from using the Mail Receive flow element).
Set email body	The email body that should be associated with the job (for example from using the Mail Receive flow element).
Set user name	The user name that should be associated with the job (for example from using a Submit Point flow element).
Set user full name	The user full name that should be associated with the job (for example from using a Submit Point flow element).
Set user email	The user email that should be associated with the job (for example from using a Submit Point flow element).

3. Buggy Reporting

Getting test jobs into a flow is only part of the problem of course. The other part is trying to figure out whether what happens in the flow is expected and correct. To make that easier, Buggy reporting can export a wealth of information for each job passing through a flow (whether it is a test job, or an actual job doesn't matter).

Buggy Reporting is designed to be attached to any folder element in a Switch flow. Jobs do not flow "through" it but are inspected "while they pass the folder". This was an important design goal; you do not have to modify a flow to insert Buggy reporting. Simply attach it at the point in the flow where you want to inspect jobs.

The rest of this chapter describes the different functionalities and the app properties that can be used to configure them.

3.1. Reporting on jobs

The main goal of the app is to show you all details about the jobs that pass through the flow. To do so, you must configure a reporting folder (using the property tellingly named "Reporting folder"). The app automatically creates a sub folder in this reporting folder for each job it reports on. You can set the name of the sub folder yourself by modifying the "Sub folder name" property, or you can leave it blank and then you'll get a time stamp as folder name.

In this sub folder, the app writes all kinds of interesting information. You can be selective, by changing the "Report everything" property to "No", and then manually selecting the information you're interested in, but why would you? All of this information is quite useful and it takes only a second to generate it.

3.1.1. A copy of the job

The app copies the job into the reporting sub folder. This gives you an opportunity to inspect what the job looks like at that point in the flow without putting the flow on hold and intercepting it that way. It also makes it possible to easily intercept a job at different stages in the flow and see how they change.

3.1.2. Job information

In a JSON file, the app captures everything Switch knows about your job. Want to know what hierarchy information it carries? Or what the full list of private data fields is? All of that information is written into a JSON document.

3.1.3. Datasets

If a job carries datasets, you might want to see what is in them. This can be done in Switch by exporting them using the correct flow element, but that means modifying the flow and knowing what the name of the dataset is. The app automatically exports each data set into the reporting sub folder so you can see their names and inspect them.

3.1.4. Job trail

If you have a complex flow where a lot of branching goes on, you might want to see what path a job followed. Have a look at the job trail CSV for this. This shows you all flow elements the job passed through, what outgoing connection it took (did it pass through the success or error connection for example) and it even lists the amount of seconds after the job was first seen by Switch to get to each step in the workflow.

3.2. Disabling reporting

Wouldn't it be cool if you could leave the app in the flow, but in such a way that it consumes a negligible amount of resources? But there to switch on instantly if you want to test what's going on in the flow? That is possible in two different ways:

- Use the "Set on hold" property. By default, it is set to "No" which allows Buggy Reporting to function. If you switch it to "Yes", Buggy Reporting can remain connected to the flow, but no reporting is done.
- Delete the connection between Buggy Reporting and your flow. Buggy Reporting is perfectly content to sit in your workflow without connections to it, and it's smart enough to disable all activity in that case.

If you trigger a webhook configured on it while the app has been disabled by either method, a debug message is logged to the Switch log so that you understand why nothing happens.

3.3. Cleanup

If you're in testing mode, it's easy to get confused and look at old data. That's why the app offers the possibility to automatically clean the reporting folder. It can do so in two different ways:

- Use the "On startup" property to clean up the reporting folder each time you activate the workflow.
- Use the "By webhook" property to register for a webhook with Switch. This allows integrations with appliances such as a Stream Deck or even Siri so you can clean up results on demand. If you leave the name of the webhook empty, this functionality isn't used (the default name is "cleanup".

4. Buggy Reception

Very similar to Buggy Reporting, Buggy Reception focuses on the end of a flow. At the end of the flow, you also want to be able to see all the metadata, and Buggy Reception gives you the same capabilities as Buggy Reporting on that front. But you also would like to be able to clean up the jobs that reach your output folder. But maybe not all of them, because having some live jobs in the flow makes it easy to use the variable designer window in Switch.

The rest of this chapter describes the different functionalities and the app properties that can be used to configure them.

4.1. Reporting on jobs

The main goal of the app is to show you all details about the jobs that pass through the flow. To do so, you must configure a reporting folder (using the property tellingly named "Reporting folder"). The app automatically creates a sub folder in this reporting folder for each job it reports on. You can set the name of the sub folder yourself by modifying the "Sub folder name" property, or you can leave it blank and then you'll get a time stamp as folder name.

In this sub folder, the app writes all kinds of interesting information. You can be selective, by changing the "Report everything" property to "No", and then manually selecting the information you're interested in, but why would you? All of this information is quite useful and it takes only a second to generate it.

4.1.1. A copy of the job

The app copies the job into the reporting sub folder. This gives you an opportunity to inspect what the job looks like at that point in the flow without putting the flow on hold and intercepting it that way. It also makes it possible to easily intercept a job at different stages in the flow and see how they change.

4.1.2. Job information

In a JSON file, the app captures everything Switch knows about your job. Want to know what hierarchy information it carries? Or what the full list of private data fields is? All of that information is written into a JSON document.

4.1.3. Datasets

If a job carries datasets, you might want to see what is in them. This can be done in Switch by exporting them using the correct flow element, but that means modifying the flow and knowing what the name of the dataset is. The app automatically exports each data set into the reporting sub folder so you can see their names and inspect them.

4.1.4. Job trail

If you have a complex flow where a lot of branching goes on, you might want to see what path a job followed. Have a look at the job trail CSV for this. This shows you all flow elements the job passed through, what outgoing connection it took (did it pass through the success or error connection for example) and it even lists the amount of seconds after the job was first seen by Switch to get to each step in the workflow.

4.2. Disabling reporting

Wouldn't it be cool if you could leave the app in the flow, but in such a way that it consumes a negligible amount of resources? But there to switch on instantly if you want to test what's going on in the flow? Buggy Reception does not come with a property to toggle the app off or on (that would have adverse effects on the flow), but it does let you delete the connection between your output folder and the app. You can keep the app in the flow and it won't do anything until you reconnect it.

4.3. Cleanup

If you're in testing mode, it's easy to get confused and look at old data. That's why the app offers the possibility to automatically clean the reporting folder. It can do so in two different ways:

- Use the "On startup" property to clean up the reporting folder each time you activate the workflow.

- Use the “By webhook” property to register for a webhook with Switch. This allows integrations with appliances such as a Stream Deck or even Siri so you can clean up results on demand. If you leave the name of the webhook empty, this functionality isn’t used (the default name is “cleanup”.

If you want to cleanup every job, leave the “Keep last x jobs” property set to 0 and your wish will come true. If you want to keep the last couple of jobs, set that number in this property. Remember that having live jobs in the flow is an advantage if you want to setup variables in Switch.

5. More information

5.1. Four Pees ~ feel the good flow

This app was created by Four Pees. You can find more information about our company here: http://www.fourpees.com. We created this app based on the experience we have with projects where Switch is used, but of course that is not a guarantee that the app will be suitable for every project out there.

If you run into a problem, or this app doesn't completely cover what you had hoped it would, don't hesitate to send us feedback. There are multiple ways you can do this:

- Go to our website and use the contact page: https://www.fourpees.com/en/contact.
- Send us an email at support@fourpees.com. You'll get a confirmation message and we'll be with you before you can say "Automation".

5.2. Free apps

If you're using one of our free apps, please keep in mind that our support on those is limited. We believe this is fair as free apps can't be handled the same way as payable project work. If you're using a payable app, we provide support on using the apps.

That having been said, we of course will try to help you as best as we can! Just get in contact and we'll have a conversation on how we can help you. If you think an app misses an essential feature, please let us know that as well so we can evaluate whether it would be possible and would make sense. Of course, we can't promise to implement all requests, but if you don't let us know, it's certainly never going to happen...