Four Pees

Switch

App

FourPees

Private Data Magician

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

What if the private data I have attached to a job is not the private data I want to have? No, I don't mean my flow wasn't constructed correctly! Let's say you have jobs that have either "BE" or "FR" in a private data field called `fpCountry`... In your database however, you'd like to record that as "Belgium" or "France". How do you do that?

Of course, a custom script can come to the rescue, but it feels like a lot of overkill for such a simple thing. Private Data Magician allows you to do such transformation without any sweat. Either modify the same private data field or define a new private data field without modifying the original.

Perhaps you want to modify multiple private data fields based on that input? Read one private data field, and based on its contents fill create multiple different private data fields? We've got you covered with that too! Wait, wait, what about combinations of input fields, that are remapped to something in multiple new private data fields? Aaaaarggggh. But yes, possible!

Does that sound a little like dark magic right now? The rest of this document describes how to use the app and explains the different properties.

1.1. Versions

The following is a short version overview:

- version 1: initial version of the app.

2. How to use

Using this app in a Switch flow is trivial; it has only one incoming and only one outgoing connection. On top of that, any type of job (file or folder) can flow through the app. The jobs themselves are not modified; only the private data attached to the job can be modified by the script, if the jobs passing through the script satisfy some of the rules you've defined.

There are only two properties. A "`Rules`" property that lets you setup your private data mapping rules, and a "`Separator`" property that lets you tell the app how you want to separate multiple fields in a rule.

2.1. Format of the rules

The real power of the app comes from the rules you can define to transform between private data fields, and the mappings you set up for those rules. The rest of this chapter introduces what possibilities you have. Let's introduce you to some basic principles first though.

2.1.1. Lines of text

Rules are written as lines of text. Each line that is empty is ignored, just as any line that start with a double slash ("`//`"). The latter allows you to add comments to your rules; this is a good thing that will save you headaches six months from now.

```
// This is a comment and will be ignored. Also, penguins rule!
```

These lines of text generally consist of two areas; in the beginning of the line you state what you want to search for or compare with, and at the end of the line what private data field(s) you want to define and with what values. These two areas are divided by an equal sign ("`=`").

```
BE = Belgium
```

Remark that you can have spaces before and after the equal sign. These are ignored. The part before the first equal sign is referred to as the "source" of the line, the part after it as the "target" of the line. And yes, this means the app will not work correctly when the source or target part of the line include spaces at the beginning or end. But this should be a very rare occasion.

2.1.2. Multiple fields

Sometimes the source or target part of the line can consist of multiple fields. In that case these fields are divided by a special separator character. By default, that is the pipe symbol ("`|`"), but you can change that in the separator property of the app.

```
BE = Belgium | euro
A4 | Portrait = 210 x 297
```

The two mappings above show multiple fields in the target and source part of a rule respectively. If pipe symbols could also appear in the private data you're working with, you should use the separator property to select a different separator. The separator can be more complex than just one character; you could make it "`$-$`" for example. The previous example would then become:

```
BE = Belgium $-$ euro
A4 $-$ Portrait = 210 x 297
```

The separator can be as long as you want. Don't go overboard, will you? (Just thinking of your mental health!)

2.1.3. Headers and mappings

A rule always consists of a header, followed by one or more mappings. A header can be recognized because it starts with three asterisk signs ("`***`"). Everything that follows a header, until a new header is found that indicates the start of a new rule, is a mapping.

```
*** fpCountry = fpCountry
BE = Belgium
FR = France
```

The header is where you define what private fields you want to look at, and what private data fields you want to store your results in. The mappings look at the private data in those fields in the job that passes by, and determines the outcome of the app.

2.1.4. Multiple rules

You can have as many rules as you want in the app (no, that is not a challenge). Just remember that having a line with a header starts a new rule. Rules are executed in order; and it is possible to cascade them (use the results of rule 1 in rule 2 for example). Just ask yourself whether your life really isn't complex enough already before you use this feature. Oh, and write comments if you do!

2.2. Examples

The rules above define the overall structure of the rules you'll need for the app. Now let's make things clear by looking at examples of how you can use this.

2.2.1. Changing a private data field in place

What if I have one private data field that contains the correct information, just in a rather cryptic form. So I want to clean it up. The following two rules do the same thing:

```
*** fpCountry = fpCountry
BE = Belgium
FR = France
```

```
*** fpCountry
BE = Belgium
FR = France
```

The only difference is that the second variant is a slightly more concise way of writing things. What do these rules do? Translated into English, this means the following: "Look at the private data field called "`fpCountry`" for our incoming job. Does it contain the value "`BE`"? If so, change that to "`Belgium`". Does it contain the value "`FR`"? If so, change that to "`France`".

What happens if the `fpCountry` private data field doesn't exist or contains a different value like "`DE`"? Nothing! We don't have a mapping for that, so the app will not change a thing in this case.

2.2.2. Adding an "empty value" mapping and a default value mapping

Look at the following rule:

```
*** fpCountry
BE = Belgium
FR = France
= Unknown Country
```

This is very similar as the previous example, but an additional line has been added that will be triggered if the `fpCountry` private data field didn't exist or was empty for a job. After the app, the `fpCountry` private data field then contains the value "`Unknown Country`".

To complete things, we can add one last line:

```
*** fpCountry
BE = Belgium
```

```
FR = France
= Unknown Country
.= Some Other Country
```

A mapping that has a dot ("`.`") in the source part is a default mapping. It is triggered if none of the other mappings matches our job. So now if my job's `fpCountry` private data field contains "`DE`" on entry, that will be changed to "`Some other country`" after it goes through the app.

Two more things about the default mapping:

- There can be only one default mapping per rule. It's like rings; one to rule them all!
- The default mapping can be anywhere in the mapping lines, it doesn't need to be the last line.

2.2.3. Defining a new private data field

What if we don't want to overwrite our fpCountry private data field, but instead define a new one? Easy enough:

```
*** fpCountry = fpFriendlyCountry
BE = Belgium
FR = France
= Unknown Country
.= Some Other Country
```

Instead of overwriting `fpCountry`, the app now creates a new data field called `fpFriendlyCountry` and stores our result in there. This also adds one more trick we can play:

```
*** fpCountry = fpFriendlyCountry
BE = Belgium
FR = France
= Unknown Country
.=.
```

Remark that we changed the default mapping so the target part now also contains a dot ("`.`"). This means "take whatever the source contained". So, a job that has `fpCountry` set to "`BE`", will have `fpFriendlyCountry` set to "`Belgium`". But a job where `fpCountry` is "`DE`", will have `fpFriendlyCountry` set to … "`DE`"! Whatever is in your source private data field, is simply copied to the target private data field.

2.2.4. Setting multiple target fields based on one source field

Maybe I want to not only set a user-friendly version of my country code, but also a currency name based on the country? That's possible by adding multiple target fields:

```
*** fpCountry = fpFriendlyCountry | fpCurrency
BE = Belgium | euro
FR = France | euro
US = United States | USD
```

The principle is exactly the same, it's still the same private data field (`fpCountry`) that is read and of which the value is compared to our mappings. But now two private data fields are filled in based on the mapping that was found.

Remark that if your header contains multiple target private data fields, your mappings must contain the same number of values. You can still use empty or default mappings; they are not in this example to save some space (and they to must have the same number of values)

2.2.5. Using multiple source fields to define one target field

What if I want to define a private data field based on multiple inputs? A nice example would be that I want to define the width and height of a paper, based on the name of the paper and an orientation. The name is stored in a private data field called "`fpPaper`" and the orientation in… you guessed it "`fpOrientation`". So… if my job contains "`A4`" and "`Landscape`", I want the outcome to be "`297 x 210`" for example. The following would accomplish this:

```
*** fpPaper | fpOrientation = fpSize
A4 | Landscape = 297 x 210
A4 | Portrait = 210 x 297
```

Or maybe even more functional

```
*** fpPaper | fpOrientation = fpSize | fpWidth | fpHeight
A4 | Landscape = 297 x 210 | 297 | 210
A4 | Portrait = 210 x 297 | 210 | 297
```

This example rule looks at two private data fields, and if it finds a match sets 3 other private data fields. Life is beautiful!

3. More information

3.1. 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".

3.2. 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.

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.