Text pickup

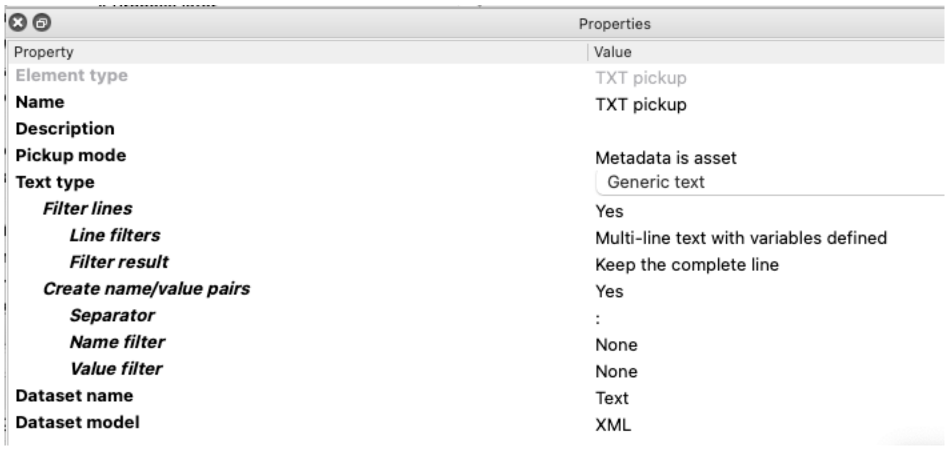
Introduction

Metadata is sometimes delivered in plain text form. As plain text is not structured it cannot be browsed and accessed in the same way as XML or JSON. This app takes a text file and makes it available as a dataset so the information in it is available as Switch variables.

There are additional features to give more structure by adding name/value pairs making it easier to extract useful metadata.

There is also support for two types of structured text files, YAML and INI files. As they are text files they cannot be used natively as metadata in Switch, but this app converts them to XML or JSON.

Properties



Properties

Property	Value
Element type	TXT pickup
Name	TXT pickup
Description	
Pickup mode	Metadata is asset
Text type	Generic text
Filter lines	Yes
Line filters	Multi-line text with variables defined
Filter result	Keep the complete line
Create name/value pairs	Yes
Separator	:
Name filter	None
Value filter	None
Dataset name	Text
Dataset model	XML

- **Pickup mode:** “Metadata is asset” should be used when the text file is the input file. “Metadata is an opaque dataset” should be used when the text is attached to the job as an opaque dataset. This can be the result of a previous “Opaque pickup”, or the result of a flow element that attached an opaque dataset, e.g. the server response of “HTTP request”. When the latter is chosen, there is an additional property to specify the name of the opaque dataset.

- **Text type:** a dropdown list of “Generic text”, “YAML” or “INI”. The YAML and INI formats are text formats, but they are structured so they can be converted to a structured format. YAML files can be converted to both XML and JSON, INI files only to JSON.

When "Generic" text is chosen a set of extra properties become available that allow you to give structure to the contents of the text.

- **Filter lines:** a Yes/No choice. Empty lines in the input text file are always ignored, but to reduce the number of lines that end up in the dataset even further, it is possible to filter lines with a certain content. In the property "Line filters" you can define a list of regular expressions. Lines that match one of the regular expressions are kept, either completely or just the matching part, depending on the setting of the "Filter result" property.

- **Create name/value pairs:** a Yes/No choice. With this property it is possible to give more structure to the XML. Instead of having all the lines as they are, the lines can be broken up into name/value pairs. This is useful when there is information in the text file like this:
  Filename: abc.pdf
  E-mail: somebody@company.com

    - **Separator:** here you can specify the character that serves as the separator between the name and the value. That will often be a character like : or =, but the use of multiple characters is allowed. The use of a regular expression is especially useful when the separator is a special character like a space or a tab, \s or \t respectively, or when the separator could be either a colon or an equal sign in which case it would be :|=
    - **Name filter:** in the same way as lines can be filtered (see above) it is also possible to specify filters for the names to be kept.
    - **Value filter:** and the same applies for values. In other words, if simply based on the separator there could be dozens of name/value pairs, you can further reduce the number of lines that end up in the dataset. The smaller the dataset, the easier it is to locate the desired information.

- **Dataset name:** the name of the dataset that is attached to the job.

- **Dataset model:** a choice of XML or JSON. Note that for INI files only JSON is allowed.

Example

Given the following text file

```
From: “Somebody” <somebody@acme.com>
Date: Tuesday 10 November 2015 09:48
To: "Enfocus support" <support@enfocus.com>
Subject: File upload from Acme: Brochure.pdf

Dear Recipient,
Please note, that files for your attention have been uploaded to the Acme file exchange
server.
*********************************************************************************
Filename: Brochure.pdf
File availability Period: 20 days
Downloadlink: https://fileserver.acme.com/getfile.php?id=MTQzMjkzNDQ3
*********************************************************************************
Kind regards,
Somebody at Acme
```

With the settings that it is a generic text and that name/value pairs have to be created with the separator :, the resulting XML will look like this:

```
<textfile>
    <line>From: “Somebody” &lt;somebody@acme.com&gt;</line>
    <line>Date: Tuesday 10 November 2015 09:48</line>
    <line>To: &quot;Enfocus support&quot; &lt;support@enfocus.com&gt;</line>
    <line>Subject: File upload from Acme: Brochure.pdf</line>
… some lines deleted …
    <name-value-pair>
        <name>Filename</name>
        <value>Brochure 1.pdf</value>
    </name-value-pair>
    <name-value-pair>
        <name>Downloadlink</name>
        <value>https://fileserver.acme.com/getfile.php?id=MTQzMjkzNDQ3</value>
    </name-value-pair>
</textfile>
```

If you wanted to use the download link in the “HTTP request” the XPath would be:
/textfile/name-value-pair[name='Downloadlink']/value

An INI file and the resulting JSON:

```
[section1]
;a comment
key1=value1
key2=value2
key3= value3
key4 =value4
key5 = value5
key6 = "value6"
key7 =" value7"
key8=VALUE8
```

```
{
  "INI": {
    "section1": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3",
      "key4": "value4",
      "key5": "value5",
      "key6": "value6",
      "key7": " value7",
      "key8": "VALUE8"
    }
  }
}
```

A YAML file and the resulting XML:

```
environment:
  matrix:
    - nodejs_version: "4"
    - nodejs_version: "5"
    - nodejs_version: "6"
    - nodejs_version: "8"
    - nodejs_version: "10"
    - nodejs_version: "" # latest

install:
  - ps: "Install-Product"
  - "npm install"

test_script:
  - "node --version"
  - "npm --version"
  - "npm test"

build: off
```

```
<YAML>
    <environment>
        <matrix>
            <nodejs_version>4</nodejs_version>
            <nodejs_version>5</nodejs_version>
            <nodejs_version>6</nodejs_version>
            <nodejs_version>8</nodejs_version>
            <nodejs_version>10</nodejs_version>
            <nodejs_version/>
        </matrix>
    </environment>
    <install>
        <ps>Install-Product</ps>npm install</install>
    <test_script>node --version</test_script>
    <test_script>npm --version</test_script>
    <test_script>npm test</test_script>
    <build>off</build>
</YAML>
```