March 15, 2022

Guide to Operations

Telefonica/O2

Secure Transport 5.5

File Format Validation

Version 1.0 of March 15, 2022

Status: Official

imagination takes shape

Preface

Purpose of this document

This document describes the functionality, the deployment and configuration as well as the usage steps of the Telefonica/O2 File Format Validation custom AR step for Axway SecureTransport.

Related documentation

This solution package is accompanied by a complete set of documentation, consisting of the following documents:

- Readme
- Design doc
- Guide to operations

Intended audience

The intended audience of this guide is system administrators responsible for deployment, configuration and maintenance of the solution. It is recommended that the readers be familiar with the products and systems listed in the "Requirements" section of this document.

Table of contents

1. Overview

This chapter briefly describes the features of the solution package.

1.1. Functional Overview

The File Format Validation Custom Advanced Routing (AR) step will perform file format validation against a specified file type. The selected file type and result of the validation (*Accepted | Refused*) will be exposed as flow attributes to the file. If after validation the file is considered invalid, a "*.refused"* extension will be appended to the file name and the error message and specified quarantine folder will be exposed as flow attributes to the file. All processed files, both valid and invalid, will be returned as a result from the step. This result can further be used as input to other OOB AR steps.

2. Requirements

This chapter describes the system requirements for the solution package.

2.1. Supported Operating System

Refer to the SecureTransport 5.5 Installation Guide

2.2. Supported Base Product Versions

SecureTransport **5.5-20220227**

3. Building

A simple “mvn clean install” command must be run inside the folder of the project to produce a zip file **securetransport-plugins-step-fileformatvalidation-<version>.zip**. Java version 11 and maven installed are prerequisites. The third-party dependencies that are needed to make a successful build are located in the *lib/* folder.

4. Installation

This chapter describes how to install and uninstall the solution package.

4.1. File Format Validation Installation

To deploy the AR step, perform the following steps on the SecureTransport (ST) Backend server. In case of a clustered environment perform the steps on ALL cluster nodes.

1. Stop the Secure Transport servers
2. Copy following packages into the *$FILEDRIVEHOME/plugins/routingSteps*, where *$FILEDRIVEHOME* represents ST installation directory:

   securetransport-plugins-step-fileformatvalidation-**<version>**.zip

3. Extract the package – e.g. “unzip securetransport-plugins-step-fileformatvalidation-<version>.zip”
4. Start the Secure Transport servers
5. Proceed with the steps in the Configuration section.

4.2. File Format Validation Uninstall

To remove this custom step and return to a previous state, please follow these steps:

1. Stop the Secure Transport servers
2. Navigate to *$FILEDRIVEHOME/plugins/routingSteps* and delete the following files and folders:

   axway-step-fileformatvalidation.jar

   axway-step-fileformatvalidation

3. Start the Secure Transport servers

5. Configuration

This chapter contains the configuration instructions needed for operating the solution.

5.1. File Format Validation Configuration

To create a custom route step, follow the steps below:

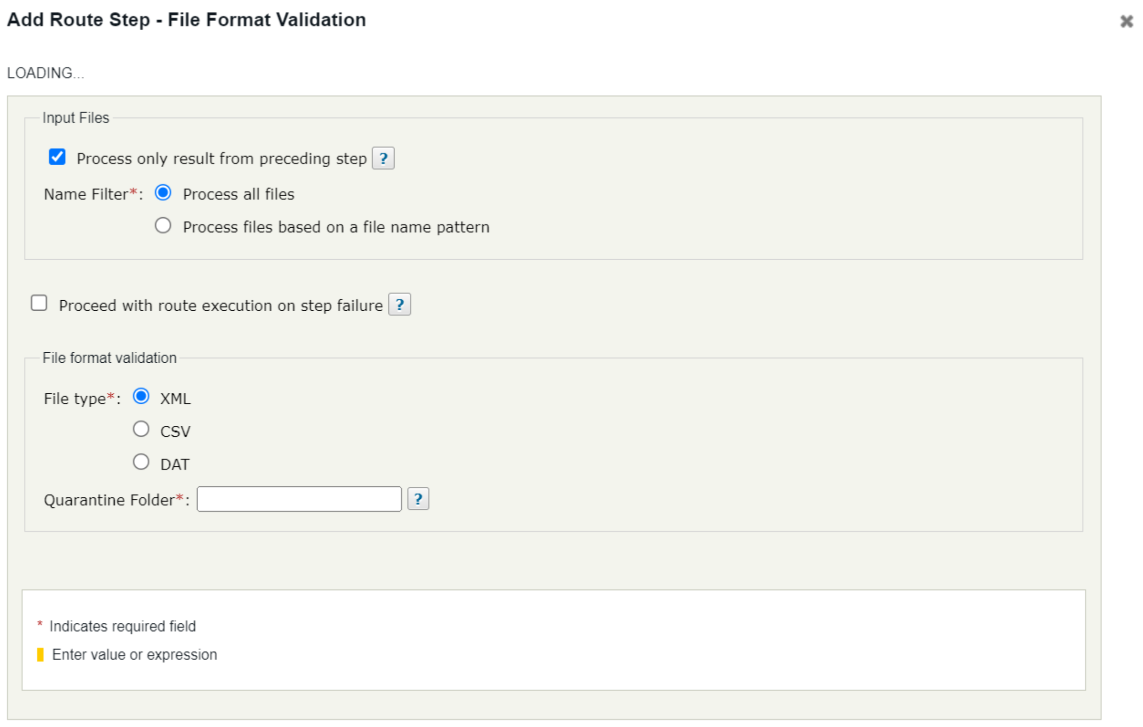
5.1.1. Create an ST user account

To create a route step, start by creating an ST account (or use an existing one). Login to ST Admin UI and navigate to **Accounts** tab. Select User Accounts and click **Add New**. Specify user account attributes and save changes. More information can be found in the chapter "Creating accounts" of Secure Transport Administration Guide.

5.1.2. Create a File Format Validation AR step

Navigate to **Routes** tab and click new Route Package Template. Type the name of the custom route template in the "Route Package Template Name:" field, then click New Route button. Enter a route name in the Route Name: field and select **File Format Validation** from the --Select Step— dropdown list.

The following mockup screen shows a sample definition based on Custom AR Step.



Add Route Step - File Format Validation

LOADING...

Input Files

[✓] Process only result from preceding step ?

Name Filter*: [● Process all files] ○ Process files based on a file name pattern

☐ Proceed with route execution on step failure ?

File format validation

File type*: [● XML] ○ CSV ○ DAT

Quarantine Folder*: ?

* Indicates required field

Enter value or expression

5.1.1. File Format Validation properties

Name	Usage	Description
File type	**Required field.**	The file type, against which the validation will be performed. Valid values are: **xml** | **csv** | **dat**.
Quarantine Folder	**Required field.**	The folder path where the invalid files should be moved. The property value will be set as a file flow attribute.
Separator	Applicable for **CSV** file type. **Required field** for CSV file type**.**	The separator for the file elements.
Element Number	Applicable for **CSV** file type.	Number of expected elements. Each record of the file must contain the specified number of elements. If there is no provided value, each record of the file must have the same number of elements as the number of elements of the first record of the file.
Fixed Length	Applicable for **DAT** file type.	Fixed length of the file records. Each record of the file must contain the specified number of symbols. If there is no provided value, validation against this field will not be performed.
Record Header	Applicable for **DAT** file type.	The specified value will be checked as a record head for each record. If there is no provided value, validation against this field will not be performed.
Forbidden Characters	Applicable for **DAT** file type.	Forbidden characters in the file. The specified symbols must not be present in the file. By default, the comma char ‘,’ is considered forbidden. If there is no provided value, the file will be scanned for comma ‘,’, otherwise – it will be scanned for ‘,’ along with the other specified chars.

5.2. Create File Format Validation using the REST API

File Format Validation custom AR step can be created and modified via Secure Transport REST API. The Secure Transport provides Swagger documentation of its REST resources. With it the user can create, delete or update routes.

To add or modify Custom AR step with the Swagger documentation navigate to:

https://10.10.10.10:444/api/v2.0/docs/index.html

Find the routes resource. Populate the body of the request (POST, PATCH or PUT), specifying the “**fileformatvalidation**” step type and the additional step property values and click ‘Execute’.

Example configuration:

```
…
"steps": [
  {
   "type": "fileformatvalidation",
   "status": "ENABLED",
   "autostart": false,
   "precedingStep": null,
   "customProperties": {
          "fileType": "dat",
          "quarantineFolder": "/AdvanceRouting/invalid",
          "recordHeader": "",
          "fixedLength": "",
          "forbiddenChars": ",",
          "separator": "",
          "elementNumber": ""
          "usePrecedingStepFiles": "true",
          "fileFilterExpressionType": "GLOB",
          "actionOnStepFailure": "FAIL"
          "fileFilterExpression": "*"
   }
  }
…
```

5.3. Expression Language

The step text fields support all expressions available in out-of-the-box AR steps, prefixed with "plugin".

Sample expressions:

- ${plugin.account.name} - Evaluates to the account name of the route.
- ${plugin.account.attributes['userVars.var1']} - Evaluates to an additional attribute in the account named var1.
- ${plugin.transfer.target} - Evaluates to the current transfer file name.

6. Operation

This chapter outlines the operation details as well as the process flow of the most common use-case of the solution.

6.1. File Format Validation

6.1.1. Overview

The File Format Validation custom step will perform file format validation against a specified file type. The step will rename the invalid files, adding a ".refused" extension to the name of each invalid file. All processed files, both valid and invalid, will be returned as a result from the step. This result can further be used as input to other OOB AR steps. The selected file type, result of the validation (Accepted| Refused), error message (in case of an invalid file) and the specified quarantine folder (in case of an invalid file) will be exposed as flow attributes to the file to be used in subsequent OOB AR steps, as well as the Sentinel reporting.

6.1.2. Exposed flow attributes

- *${flow.attributes['fileType']}* – is evaluated to the value of the File type property – *xml*, *csv* or *dat*;
- *${flow.attributes['status']}* – is evaluated to the status of the validation – *Accepted* or *Refused*;
- *${flow.attributes['errorMessage']}* – is evaluated to the error message of the validation, if there is an error message, or to empty string if there is no error and the file is valid.
- *${flow.attributes['quarantineFolder']}* – is evaluated to the value of the Quarantine folder property, if the file is not valid, or to empty string otherwise.

6.2. Use case flow overview

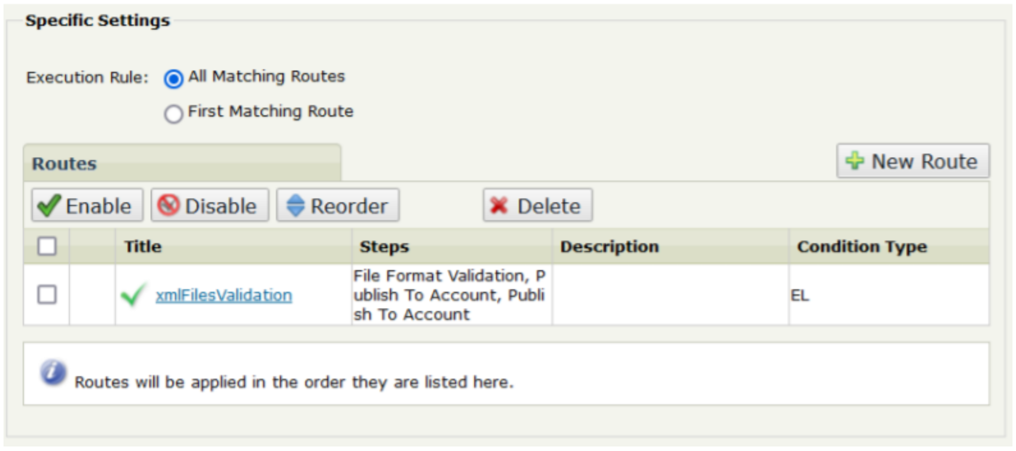
The File Format Validation custom step can be used along with Publish to Account steps to achieve flow in which files are validated against their extension and moved to different folders according to the validation result. The selected file type, result of the validation (Accepted | Refused), error message (in case of an invalid file) and the specified quarantine folder (in case of an invalid file) are exposed as flow attributes to the file and can be used in the Sentinel reporting.

One Route Package can be used to validate files with different extensions – for each file type, new Route can be created, and the Route condition can be changed to match to the desired file extension.

The following Route configuration is an example of XML file validation. Routes for CSV and DAT file validation can be created is a similar way by just modifying the expression for the Route Condition and the File Format Validation custom step configuration.

6.2.1 Validating XML files before sending to target destination

- In a newly created or existing Route Package create a new Route - xmlFilesValidation.



Specific Settings

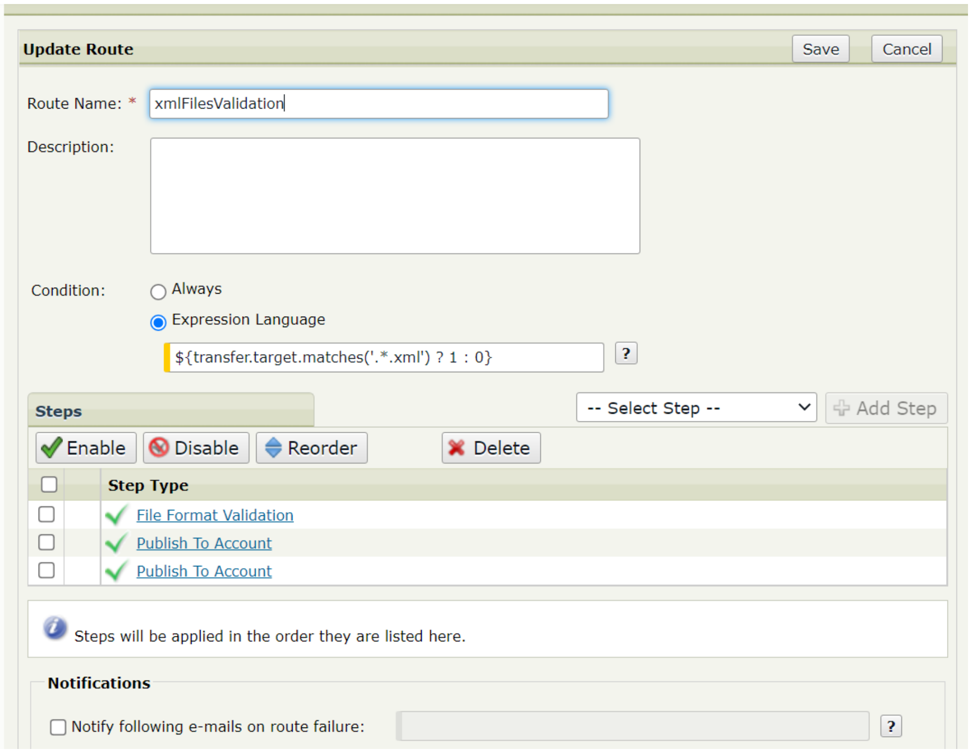
Execution Rule: [✓ All Matching Routes] First Matching Route

Routes

		Title	Steps	Description	Condition Type
☐		✓ xmlFilesValidation	File Format Validation, Publish To Account, Publish To Account		EL

Routes will be applied in the order they are listed here.

- Configure the xmlFilesValidation route to be executed only for files with ".xml" extension. This could be done using condition with the following expression: **${transfer.target.matches('.*.xml') ? 1 : 0}**
- Create 3 steps for the route – File Format Validation, which will validate the XML files and will rename the invalid files adding the ".refused" extension to their file name, Publish To Account – to send the invalid files to a quarantine folder, and another Publish To Account to send the valid files to a desired location.



Update Route

Route Name: * xmlFilesValidation

Description:

Condition: Always [✓ Expression Language]

${transfer.target.matches('.*.xml') ? 1 : 0}

Steps

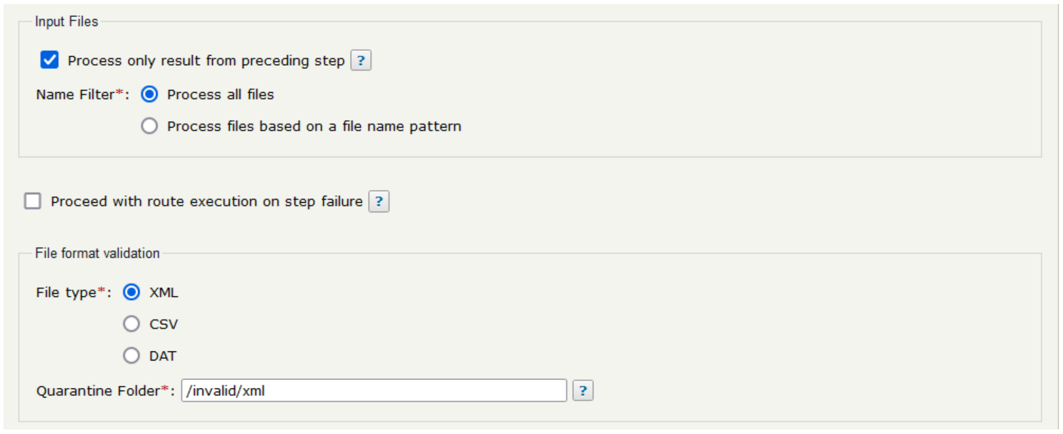
	Step Type
☐	✓ File Format Validation
☐	✓ Publish To Account
☐	✓ Publish To Account

Steps will be applied in the order they are listed here.

Notifications

☐ Notify following e-mails on route failure:

• Configure the File Format Validation step. Select the 'XML' file type and enter the path to the quarantine folder.



Input Files

☑ Process only result from preceding step ?

Name Filter*: [✓ Process all files] Process files based on a file name pattern

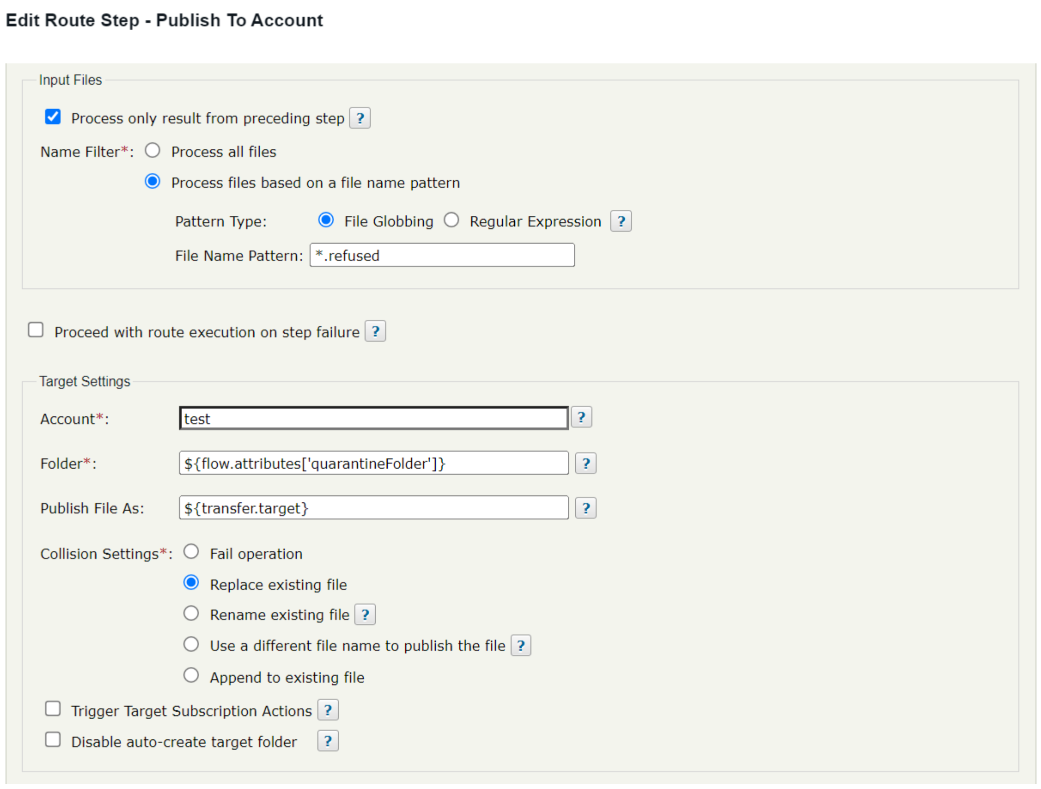
☐ Proceed with route execution on step failure ?

File format validation

File type*: [✓ XML] CSV DAT

Quarantine Folder*: /invalid/xml ?

• Configure the first Publish To Account step, which will send the invalid files to a quarantine folder. The invalid files are renamed from the File Format Validation step and have the ".refused" extension. To route only the invalid files set the File Name Pattern to: ***.refused** . Use the expression for the quarantine folder: **${flow.attributes['quarantineFolder']}**, to configure the destination folder. To send the invalid files with their original file name, set the Publish File As to: **${transfer.target}**.



Edit Route Step - Publish To Account

Input Files

☑ Process only result from preceding step ?

Name Filter*: Process all files [✓ Process files based on a file name pattern]

Pattern Type: [✓ File Globbing] Regular Expression ?

File Name Pattern: *.refused

☐ Proceed with route execution on step failure ?

Target Settings

Account*: test ?

Folder*: ${flow.attributes['quarantineFolder']} ?

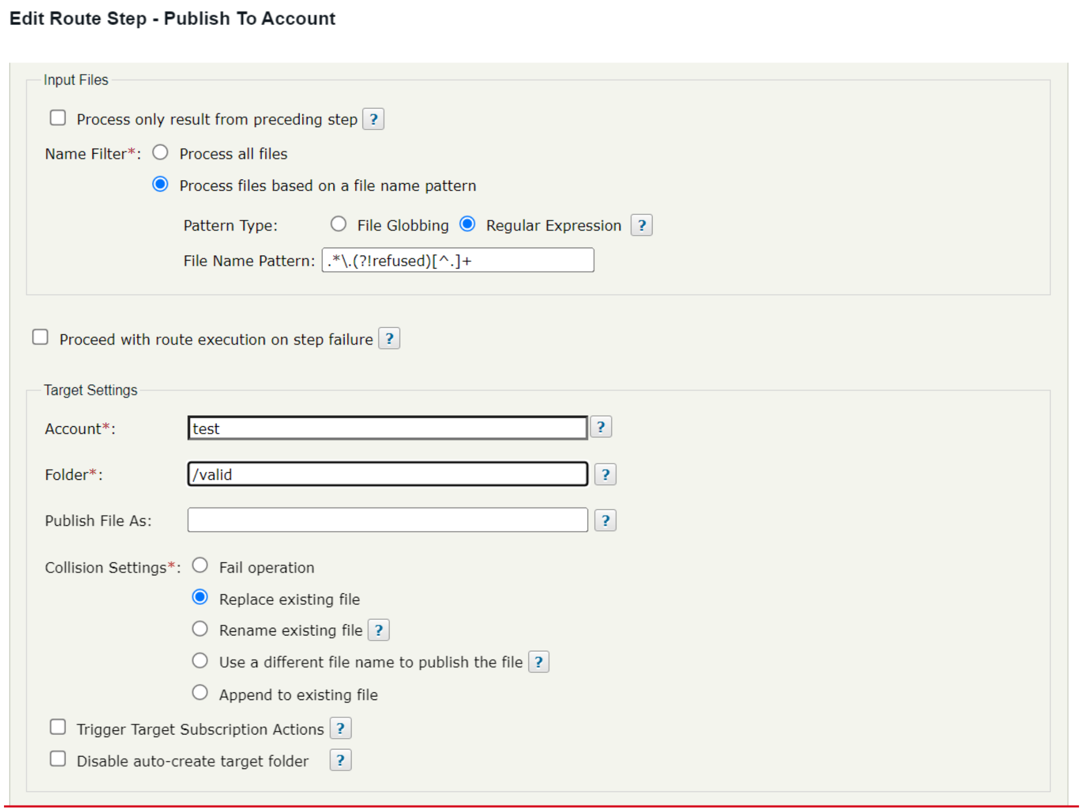
Publish File As: ${transfer.target} ?

Collision Settings*: Fail operation [✓ Replace existing file] Rename existing file ? Use a different file name to publish the file ? Append to existing file

☐ Trigger Target Subscription Actions ?

☐ Disable auto-create target folder ?

• Configure a second Publish To Account step, which will send to target destination only the valid files. The valid files are NOT renamed from the File Format Validation step, so their file name does not end with the ".refused" extension. To route only the valid files set the Pattern Type to Regular Expression and the File Name Pattern to: **.*\.(?!refused)[^.]+**


Edit Route Step - Publish To Account

Input Files

- [ ] Process only result from preceding step ?

Name Filter*: ○ Process all files [✓ Process files based on a file name pattern]

Pattern Type: ○ File Globbing [✓ Regular Expression] ?

File Name Pattern: .*\.(?!refused)[^.]+

- [ ] Proceed with route execution on step failure ?

Target Settings

Account*: test ?

Folder*: /valid ?

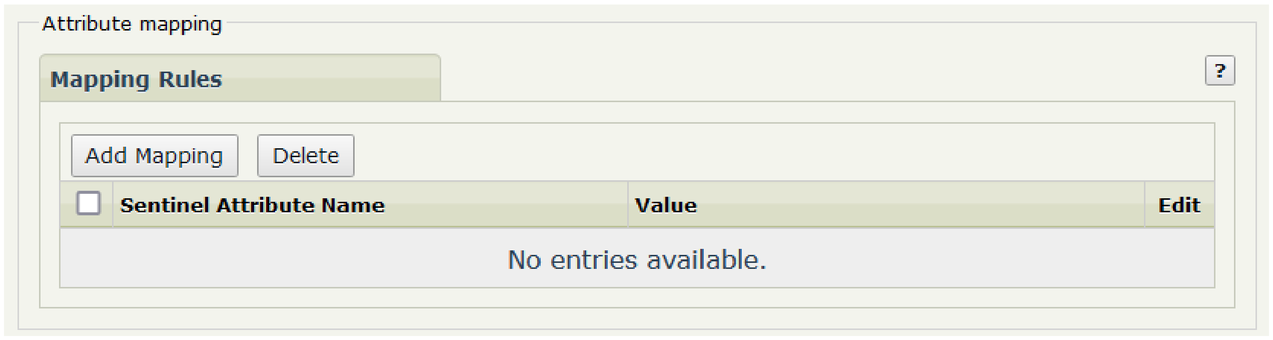
Publish File As: ?

Collision Settings*: ○ Fail operation [✓ Replace existing file] ○ Rename existing file ? ○ Use a different file name to publish the file ? ○ Append to existing file

- [ ] Trigger Target Subscription Actions ?
- [ ] Disable auto-create target folder ?

6.3 Sentinel reporting

Axway Sentinel allows to report custom values in existing Sentinel columns via Attribute Mapping. The Mapping Rules table contains Axway Sentinel column where Secure Transport reports related values. It can report values which contain flow attributes.



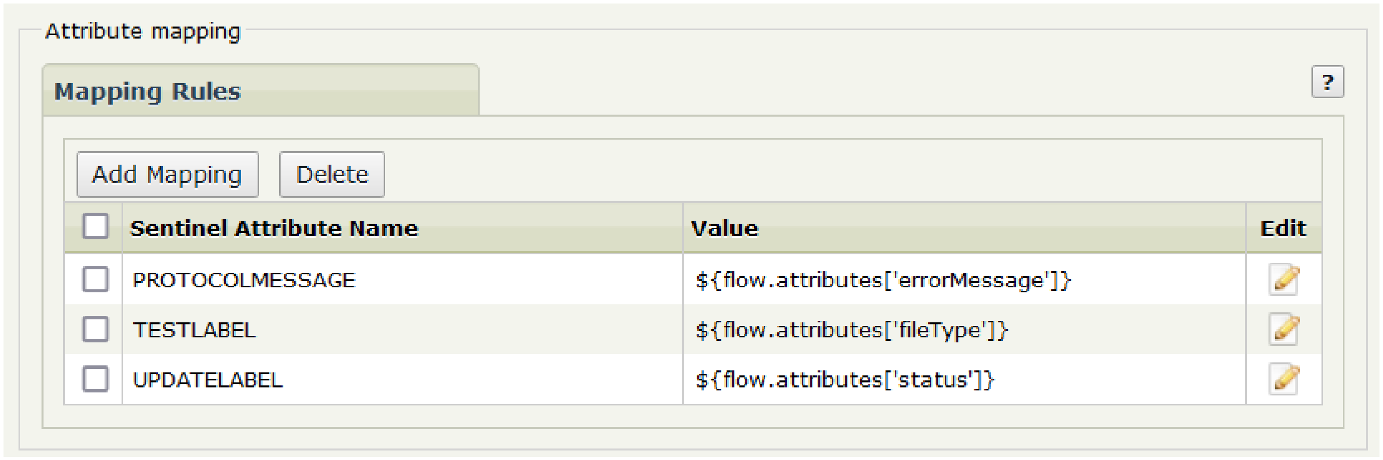
☐	Sentinel Attribute Name	Value	Edit
	No entries available.		

The flow attributes set by the File Format Validation custom AR step can be reported and available under the **AVAILABLE** event state when using a Sentinel attributes mapping.

Important notes:

- No additional mapping is required for the filename, because it will be reported under the XFBTransfer Tracking Object **PROTOCOLFILENAME** column on **AVAILABLE** event state.
- No additional mapping is required for the quarantine file path, because it will be reported under the XFBTransfer Tracking Object **FILENAME** column on **AVAILABLE** event state, where the final file destination after the Publish to Account step completes is populated. In case of an invalid file this field will hold the file path to the quarantine location, where the file was sent with the configured for invalid files processing Publish To Account step. In case of a valid file this field will hold the file path to the desired location, specified in the configured for valid files processing Publish To Account step.

Some of the System attributes columns, which are not used by Secure Transport for reporting, are: TESTLABEL (attribute length: 50), UPDATELABEL (attribute length: 250), PROTOCOLMESSAGE (attribute length: 4000) and COMMENTACK (attribute length: 512), so the following mapping could be used:



Sentinel Attribute Name	Value	Edit
PROTOCOLMESSAGE	${flow.attributes['errorMessage']}	
TESTLABEL	${flow.attributes['fileType']}	
UPDATELABEL	${flow.attributes['status']}	

Where:

- *${flow.attributes['fileType']}* – is evaluated to the file type – *xml*, *csv* or *dat*;
- *${flow.attributes['status']}* – is evaluated to the status of the validation of the file – *Accepted* or *Refused*;
- *${flow.attributes['errorMessage']}* – is evaluated to the error message of the validation, if there is an error message, or to empty string if there is no error and the file is valid.

Other possible expressions:

- *${flow.attributes['quarantineFolder']}* – is evaluated to the value of the Quarantine folder set in the File Format Validation AR step configuration;
- *${stenv.homedir}* – is evaluated to the account home directory;
- *${stenv.target}* – is evaluated to the filename;
- *${stenv.fulltarget}* – is evaluated to the full file path;
- *${empty flow.attributes['errorMessage'] ? 'NA' : flow.attributes['errorMessage']}* – is evaluated to the error message of the validation, if there is an error message, or to 'NA' if there is no error and the file is valid.

7. Custom code maintenance

The purpose of this section is to explain the basic parts of the Telefonica/O2 custom code for File format validation custom AR step.

1. There are two files in the **fileformatvalidation-distribution/src/assembly**: *jar.xml* and *bin.xml*. Their main goal is to handle the production of the jar file and the packaging.
2. In **fileformatvalidation-spi-1-1/src/main/java** resides the backend logic.
3. In **fileformatvalidation-spi-1-1/src/main/resources/html** resides the html UI of the Custom AR step.

8. Troubleshooting

This chapter outlines the solution troubleshooting steps as well as some general troubleshooting recommendations for the Customer to maintain the code on its own.

8.1. General Troubleshooting Steps

If you encounter any problem with this solution, please follow these steps:

Check if the observed issue is listed under the “known issues” section in the Readme file document shipped with the installed package.

Restart the failing server(s) or client(s).

You can easily go to “File Tracking” to check the statuses and information about the file transfers.

“Server log” section is used for troubleshooting actions done by SecureTransport with its logs.

Both sections can be found in the Admin UI of SecureTransport.

Reproduce the issue with enabled Debug logging if applicable (Refer to the “Debug Logging” section for instructions).

8.2. Debug Logging

To enable the debug logs for the Custom AR Steps, edit the file *FILEDRIVEHOME/conf/tm-log4j.xml* with a text editor. Add the AR step logger element in the loggers section and set the level value to “debug” as shown below:

```
<Logger name="com.axway.st.plugins.routing" level="DEBUG" additivity="false">
        <AppenderRef ref="ServerLog"/>
</Logger>
```

The log level could be *INFO*, *ERROR* or *DEBUG*. The *INFO* level is recommended for production environment.

When ready, restart the TM service for the change to take effect.

imagination takes shape

9. Training and Consulting services

For more information about Axway training and consulting services, visit the Axway website: www.axway.com