Switch2Switch

Description

Switch2Switch is a bundle of two apps, one that sends a file from Switch Server A to Switch Server B, and one that receives the file on Switch Server B and sends an acknowledgement back to Switch Server A.

The apps allow the definition of a channel so several instances of *Send to Switch* can be linked to several instances of *Receive from Switch* in the correct way.

There is a property that determines whether the private data and the datasets are also transferred or not.

Technical background

The two apps communicate using the remote processing feature that was added in the Switch 2019 Spring Release. This means there are four communication exchanges taking place.

First of all, the local Switch Server sends a webhook to the remote Switch Server with what is called a payload. This payload contains information about the job and about the local server.

The remote server then contacts the REST API of the local server and downloads the job. In other words, the local server does not push the file to the remote server, it is the remote server that pulls it.

The remote server sends back an acknowledgement that the job has been successfully received to the REST API of the local server.

Finally, the local server sends an internal webhook to the *Send to Switch* app instance that the job was transferred, and the job is then sent to the Success output connection.

Compatibility

Switch 2019 Fall Release and higher. Windows or Mac OSX.

Compatibility third-party applications

No third-party applications are used by this app.

Connections

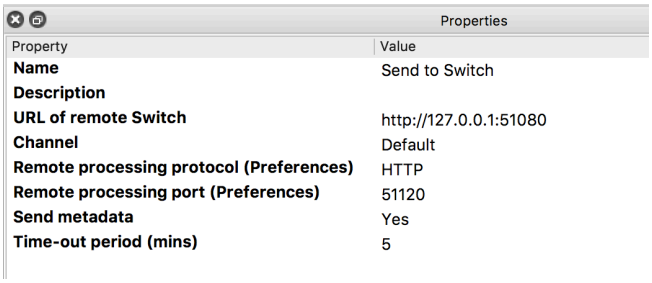
Send to Switch has incoming connections and traffic-light outgoing connections, Success obviously for when the server B confirms having received the file, and Error when it was not

possible to send the file or when the acknowledgement timed out. The latter situation can occur when the flow on server B is not active.

The outgoing connections on *Send to Switch* are optional. When no outgoing connections are defined and there is an error situation, the job will be sent to *Problem jobs*. When there is only a *Success* outgoing connection and no *Error* connection, the job will likewise end up in *Problem jobs*.

Receive from Switch has no incoming connections and one outgoing connection.

Properties detailed info



Flow element properties Send to Switch

- **URL of remote Switch:** https://127.0.0.1:51080
  The app sends a webhook to the remote Switch, so the URL has to be built up of the protocol (http or https), the IP address or the name of the remote server and the port on which webhooks are received. The protocol and the port number are defined in the Preferences of the **remote** Switch Server in the section Webhooks.
- **Channel:**
  In order to be able to send different jobs to different instances of the *Receive from Switch* apps on the remote server, you can define a Channel name. E.g. PDF files are sent to the channel “PDF” and images to the channel “Image”. This name can be anything, you just have to make sure that it is the same on both ends.
  The technical background: the channel name becomes a part of the path of the webhook, so it becomes unique. Note that when there are multiple instances of *Receive from Switch* on the remote server using the same channel name, they will all receive the file.
- **Remote processing protocol (Preferences)**
  The two apps communicate using the remote processing feature that was added in the Switch 2019 Spring Release. This property defines the protocol to be used by the remote server when downloading the file. It must be the same as the one that is defined in the Preferences in the Remote processing section of the **local** Switch Server.
  The reason this is a property is because the Switch scripting API does not yet give access to the preference settings.
- **Remote processing port (Preferences)**
  The same as for the protocol.
- **Send metadata**
  When set to “Yes” (default) the private data and the datasets are also transferred to the remote system. The job ticket information (priority, hierarchy, e-mail addresses,

e-mail body, and job state) is always transferred.

- **Time-out period**
  When the remote Switch Server is not running, it is immediately clear that something is wrong because the webhook cannot be sent. However, when the remote Switch is up and running but the flow in which the *Receive from Switch* app is used is not, then it will not be immediately clear. This property sets a time-out period after which the job will be sent to the Error connection.

Flow element properties Receive from Switch

- Channel
  There is only one property required and that is the channel name. It defines a string that links a certain instance of this app with a certain instance of the Send to Switch app.

Using job folders

Please note that the Switch2Switch apps do NOT support the transfer of job folders. If transferring job folders is a requirement you can overcome this limitation by packing the job before transferring it and unpacking it after the transfer.

The metadata and job ticket information of the job folder can be added to the packed job and restored in *Unpack job*. In that case it is recommended not to let *Send to Switch* send any metadata or that work is being done twice.

What's new in v2

The first version was compatible with the Switch 2019 Spring release. In that version it was not yet possible to transfer the private data and the datasets of the job. The Switch 2019 Fall release added this functionality and this version of the app uses that. It is therefore no longer necessary to pack job files before sending them and unpacking them after transfer.

What's new in v3

Now all the properties allow the use of single-line text with variables and script expressions.

What's new in v4

It is now possible not to use any outgoing connections on *Send to Switch.* If there are no outgoing connections and there is an error situation, the job will be sent to *Problem jobs*. The same is true if there is no *Error* outgoing connection, but only a *Success* one.

Credits
This app bundle was a joint development during a scripting training in April 2019 at Enfocus in which the following people participated:

Aleksey Safronov, AM Labs
Arsen Manukyan, AM Labs
Cedric Sintes, Pixel Tech
Chris De Clercq, Lab9
Danny Roso, Wifac
Freddy Pieters, Enfocus
Horațiu Slăvescu, Printman
Ingo Rösseler, SNAP Innovation
Jan De Brabanter, Enfocus
Jan Suhr, ColorConsult
Laurent De Wilde, Enfocus
Loïc Aigon, Agile Streams
Malcolm McKenzie, Colour Engine
Marc De Blanck, Sagam
Martin Owen, Agfa
Martin Thomann, Topix
Muharrem Altintaș, WIT
Pascal Miquet, Pixel Tech
Stéphane Allain, Neosa
Tim Melis, Catena Company
Tjeerd Schutte, ISI
Vinod Kirouchenamourty, Agile Streams
Wim Deliveyne, Esko