

## HoldJobPlus Version 2

### Description

HoldJobPlus provides additional mechanisms to hold and then release a job. With it you can do the following:

- Delay a job for a designated period
- Utilize a configured date or date-time value and optional positive or negative offset to release the job
- Flush all jobs based on a configured schedule of one or more day and time designations, example *WED@17:00* to release all jobs on Wednesday at 5:00 PM
- Set up two available webhooks. One to inquire about the jobs held and the other to release all or selected jobs. This combination allows you to implement a website to view, select and release held jobs.
- Implement a combination of these methods.
- Prevent duplicate jobs by superseding an existing job with a newer job that has the exact same job name.

### Compatibility

Switch 2021 Spring and higher.

### Connections

At least one incoming connection required. Only one output connection is allowed.

### Properties detailed info

The *Release trigger* property allows you to designate a delay method or specify that no delay method is necessary – for example, you may only want to use the webhooks or flush method for releasing jobs. If you choose *Delay*, select the *Time Unit* and provide a value for *Release date-time offset*. For *Date-time*, configure the *Job release date-time* to resolve to an ISO 8601 formatted date or date-time, examples 2022-02-18 or 2022-02-18T15:30:00. Select the *Time unit* and provide the *Release date-time offset* which can be a positive or negative value. For example, if the time you configured is based on an SLA (service level agreement), and you want to move the file into production 24 hours before this date, specify *hours* for the time unit and *-24* for the offset. Use the *Reset job release* property to control whether the release mechanism and date-time value are reset on flow restart.

*(Note: You may find it more predictable to select [Text](#) as opposed to [Date](#) when configuring the ISO date value for use as the Job release date-time – particularly if you are only returning a month-day-year such as 2022-02-18.)*

*Flush jobs* enables you to designate a list of day/time values in the form *day@time* such as *MON@13:30* to releases jobs every Monday at 1:30PM. *Flush schedule* is a multi-line text field with each day-time designation placed on a separate line. *Reset flush date-time on restart*, if set to *Yes*, instructs the app on a flow restart to reset the currently saved flush date-time to the next available date-time if the current date-time has passed. This may be desirable if you don't want the jobs to be flushed immediately when restarting the flow after the current date-time has passed.

*Enable webhooks* allows you to implement two webhooks. An *inquire* webhook provides the ability to query the app for a list of queued jobs returned as a JSON response. The *release* webhook provides the ability to release all or a list of jobs by their job key. You can protect access to these webhooks by enabling the *Require webhook credential* and providing a value in *Webhook credential*. This credential then must be passed in a header named "password".

If the webhooks option is to be implemented, use the *Job key* property to create a job identifier for each job. The job key can either be unique for all jobs or it can be the same for a group of jobs that you may want to release in unison. If this property is not configured, the job key value is automatically set to the Switch job id.

Enabling *Include job information* allows you to configure additional information that will be passed back in the JSON response of the *inquire* webhook. This is a *multi-line text with variables* editor. Place each *tag:value* pair on a separate line. Each pair will be transformed into a "*key*":"*value*" element in the JSON response within a *jobinformation* element.

Example multi-line text entries:

```
Quantity:[Metadata.Text:Path="/Workflow/Quantity",Dataset="WorkflowXml",Model="XML"]
Priority:[Metadata.Text:Path="/Workflow/Priority",Dataset="WorkflowXml",Model="XML"]
Pages:[Stats.NumberOfPages]
```

Resulting JSON response:

```
{
  "status": "active",
  "jobs": [
    {
      "jobkey": "1103005312983-1",
      "jobName": "1103005312983_Invitations 5.5x7.5 175ptmatte.pdf",
      "releasedate": "2022-03-02T16:00-5",
      "jobinformation": {
        "Pages": "54",
        "Priority": "Normal",
        "Quantity": "25"
      }
    }
  ]
}
```

The **status** value is an indicator for the state of the Switch flow. When a flow is restarted, all existing jobs in the incoming connections must be re-acknowledged. If there are many jobs this can take some time. During this process, **status** will indicate the current restart state: **flow restart** indicates that not all jobs have been re-acknowledged; **purge** indicates that any "orphaned" jobs are being removed from the apps internal job list; **active** indicates that the flow is fully operational.

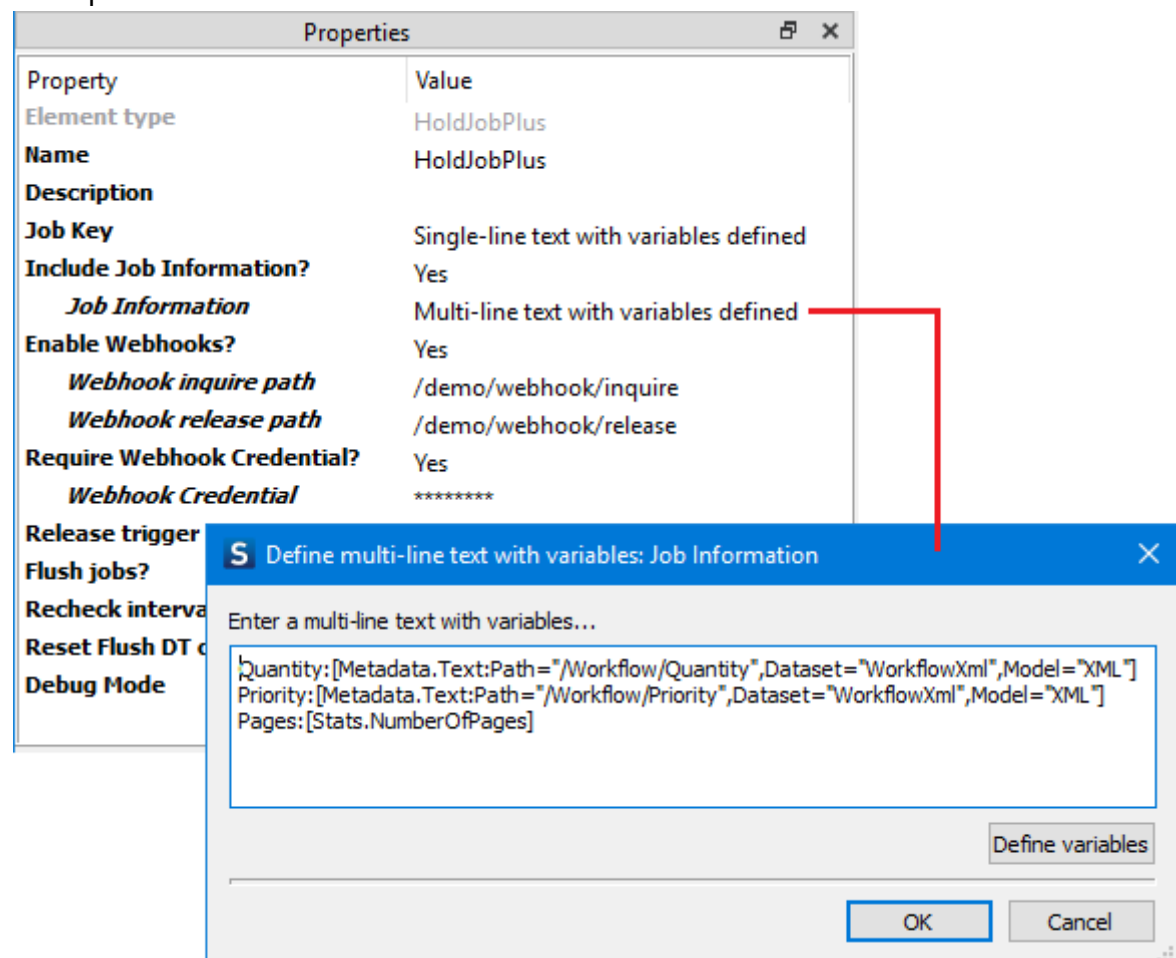
The *Supersede duplicate jobs* property, when set to *Yes*, eliminates duplicate jobs (jobs with the exact same name) by superseding the older job that is waiting with the newer incoming job.

Set the *Debug mode* property to *Yes* to log additional informational messages to help you better understand how the app is interpreting your property settings.

### Implementing the webhooks

Switch webhook requests must be sent to the host name or IP address of your Switch server and use the port identified as the **Port for the switch Web Service** in the *Web Services* section of *Preferences*. Also, *and this is important*, you must add **/scripting** to the beginning of the paths that are configured for the two webhooks in the request URL. When making the request, you should use the **PUT** method and, if you have chosen to require a webhook credential, provide a header named "password" with the configured value. Use the **Job information** property to add additional information to return in the webhook response.

Example

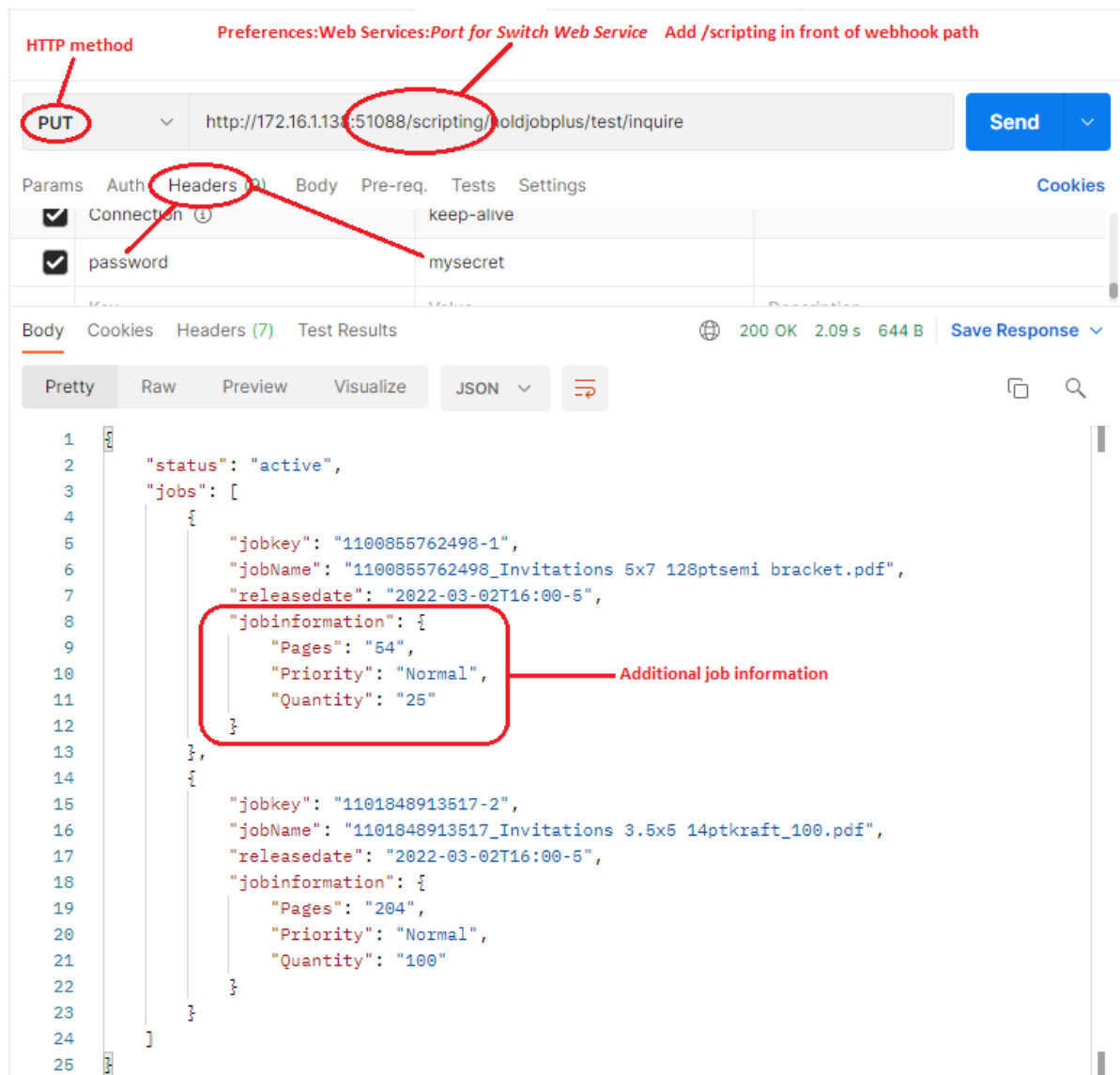


The proper URLs for the webhooks would be...

<http://MySwitchServer:51088/scripting/demo/webhook/inquire>

<http://MySwitchServer:51088/scripting/demo/webhook/release?jobs=ID12345,ID98765>

Note that you should use the protocol (http or https) based on the **Protocol** setting in the **Web Services** preference. Be sure to send the request using the **PUT** method. A great way to test the webhooks is with the **Postman** utility. Below is a screen shot of Postman sending a request based on the properties in the above example and the resulting JSON response:



The screenshot shows the Postman interface. At the top, the HTTP method is set to **PUT** (circled in red). The URL is `http://172.16.113.1:51088/scripting/moldjobplus/test/inquire` (the port and path are circled in red). The **Headers** tab is selected, showing a `password` header with the value `mysecret`. The response status is **200 OK** with a response time of **2.09 s** and a size of **644 B**. The response body is displayed in **JSON** format, showing a list of jobs. The first job's `jobinformation` object is highlighted with a red box and labeled **Additional job information**.

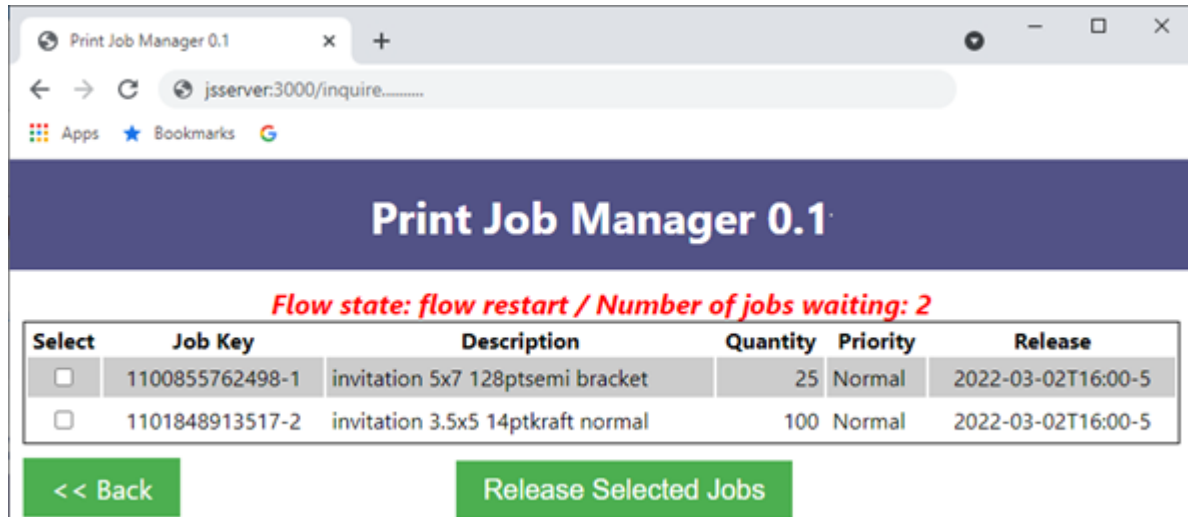
```
1  {
2    "status": "active",
3    "jobs": [
4      {
5        "jobkey": "1100855762498-1",
6        "jobName": "1100855762498_Invitations 5x7 128ptsemi bracket.pdf",
7        "releasedate": "2022-03-02T16:00-5",
8        "jobinformation": {
9          "Pages": "54",
10         "Priority": "Normal",
11         "Quantity": "25"
12       }
13     },
14     {
15       "jobkey": "1101848913517-2",
16       "jobName": "1101848913517_Invitations 3.5x5 14ptkraft_100.pdf",
17       "releasedate": "2022-03-02T16:00-5",
18       "jobinformation": {
19         "Pages": "204",
20         "Priority": "Normal",
21         "Quantity": "100"
22       }
23     }
24   ]
25 }
```

For the **release** webhook, you should supply a **jobs** query parameter set to either **all** or a comma separated list of job keys...

<http://172.14.3.180:51088/scripting/demo/webhook/release?jobs=209405-82476,504854-58211>

<http://172.14.3.180:51088/scripting/demo/webhook/release?jobs=all>

You can download a sample Node.js server app from my web site. It is based on *Express* and *Pug*...



Expand the contents of the zip file into a directory on your system. Install Node.js if you haven't already done so (<https://nodejs.org/en/download/>).

Only a few minor changes should be needed:

- edit the **app.js** file and update *switchPassword* to match the password credential for your webhooks and change the *port* if it is in conflict with another service
- edit **viewswaitingJobs.pug** to edit the table columns to match your job information. The JSON response is passed to *pug* so all you need to do is reference the specific items you want to display as **td** items. (*Note: be careful to maintain the indents as this is important to pug.*) Remember that your custom information is included in the *job.jobinformation* element so if you configured an item called *description* you would reference it as *job.jobinformation.description*.
- edit **control.json** to set **switchHost** to match your host name or ip address. Under **jobTypes**, create an entry for each instance of *HoldJobPlus* that has implemented the webhook option. The value for *name* will appear under *Job Type* and the values for *inquire* and *release* should match the configured webhook paths preceded by /scripting.

To run on Windows, open a command prompt instance, set the working directory to the **jobManager** directory (where **app.js** can be found) and enter **node app.js**. Ctrl-C will stop the server. (*I'm not sure how you do this on a Mac.*)

From a browser, type in the host name or ip address and port to access the server, examples: <http://192.168.1.20:3000> or <http://localhost:3000>

Clicking on the link under *Inquiry* will return a list of the currently held jobs. You can select one or more jobs and click *Release Selected Jobs* which will move the jobs to the outgoing connection. Clicking a link under *Release All Jobs* will do just that.