

SAS 4248-2020

Best Practices for Scheduling in SAS® Viya®

Ursula H. Polo, SAS Institute Inc.

ABSTRACT

To help you harness the power of the SAS® Platform, SAS® Viya® contains numerous scheduling capabilities. The SAS Viya documentation provides detailed instructions about how to schedule a job or flow. However, to give you a better understanding about which method might be best for you, this paper goes into depth about the five scheduling methods available in SAS Viya. This paper discusses the advantages, disadvantages and alternate approaches, and best practices for each scheduling procedure. This paper details the scheduling types available and helps you take full advantage of the SAS Viya environment.

INTRODUCTION

You can use the following five methods to schedule in SAS Viya: CAS table state management, the **Jobs** page in SAS® Environment Manager, SAS® Job Execution Web Application, SAS® Studio, and the command line interface (CLI). Each scheduling method has unique characteristics although they can perform the same basic functions. These applications are your key to scheduling because they all use the Job Execution microservice or the Scheduling microservice behind the scenes. The Scheduling microservice is used to start the job, which is a synchronous operation to the Job Execution microservice that runs the job. This paper explores each method's characteristics and discusses best practices for scheduling. This paper is organized to discuss the methods based on their popularity with SAS customers.

CAS TABLE STATE MANAGEMENT

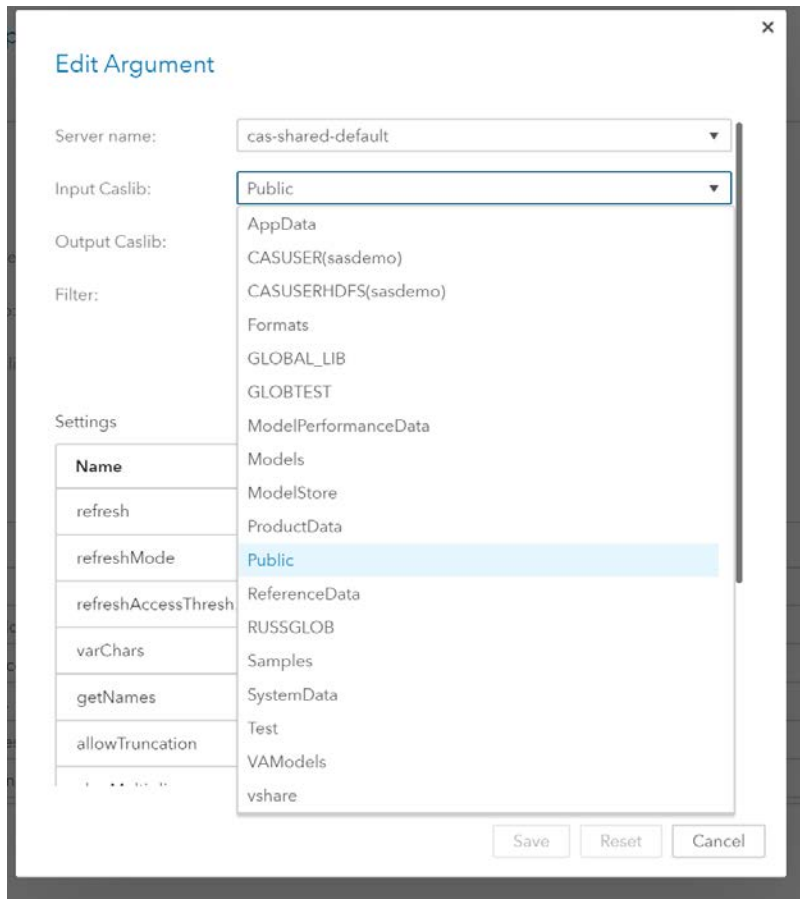
CAS table state management is a feature in SAS® Environment Manager for SAS® Viya®. To access this method, click the **Jobs** page in the left navigation pane in SAS Environment Manager. If you are a SAS Environment Manager administrator, you see three sample jobs on the **Scheduling** tab. You can use these jobs to manage CAS tables. Also, you can modify these jobs as needed for your system.

ADVANTAGES

CAS table state management enables you to schedule jobs to unload, load, and import your data into SAS® Cloud Analytic Services (CAS). This method contains both a graphical user interface (GUI) and the ability to input code that you can use to schedule your jobs. To make it easier, the SAS documentation contains CAS table state management samples that you can modify to provide additional flexibility with unloading, loading, and importing your data into CAS. Also, there are options that refresh the CAS data when the original data source has been modified. Combined with the scheduling capabilities, this option can be a time saver if data is updated often.

DISADVANTAGES

Not all input caslibs are visible from SAS Environment Manager. If you have more than 20 caslibs as shown in Display 1 below, you will not be able to view all of them through the GUI.



Display 1. View Input Caslibs in SAS Environment Manager

One workaround to the caslib display limit is to run code via the CLI to list any input caslibs. For example, here is sample JSON code, with the section where you can modify the inputCaslib highlighted:

Note: For additional details, see [Scheduling: How to \(Command Line Interface\)](#) for SAS Viya 3.3 or [Scheduling Command Line Interface](#) for SAS Viya 3.4.

Load JSON code

```
{
  "enabled" : true,
  "type" : "LOAD",
  "settings" : {
    "refresh" : false,
    "refreshMode" : "newer",
    "refreshAccessThreshold" : 0,
    "varChars" : false,
    "getNames" : true,
    "allowTruncation" : true,
    "charMultiplier" : 2,
    "stripBlanks" : false,
    "guessRows" : 200,
    "scope" : "global",
```

```

    "encoding" : "utf-8",
    "delimiter" : ",",
    "successJobId" : ""
  },
  "selectors" : [ {
    "serverName" : "cas-shared-default",
    "inputCaslib" : "Public",
    "outputCaslib" : "Public",
    "filter" : "or(endsWith(tableReference.sourceTableName, '.sashdat'),
endsWith(tableReference.sourceTableName, '.SASHDAT'))",
    "settings" : { }
  } ]
}

```

In the highlighted section, you can specify the inputCaslib that you are looking for. To learn more about the using the CLI, see the [Command Line Interface section](#) in this paper.

Another possible disadvantage with this method is that it uses the following default file types for importation: CSV, SAS7BDAT, or XLSX. In Display 2 below, the **Filter** field shows the default file types available:

Edit Argument

Server name: cas-shared-default

Input Caslib: Public

Output Caslib: Public

Filter: or(endsWith(name, '.csv'), endsWith(name, '.CSV'), endsWith(name, '.sas7bdat'), endsWith(name, '.SAS7BDAT'), endsWith(name, '.xls'), endsWith(name, '.XLS'), endsWith(name, '.xlsx'), endsWith(name, '.XLSX'))

Settings

Name	Type	Value
varChars	BOOLEAN	false
getNames	BOOLEAN	true
allowTruncation	BOOLEAN	true
charMultiplier	NUMBER	2
stripBlanks	BOOLEAN	false
guessRows	NUMBER	200

Save Reset Cancel

Display 2. Dialog Box Where You Can View Default File Types

If you require a different file type, you can code it as shown by using this example code:

```
{
  "enabled": true,
  "type": "IMPORT",
  "settings": {
    "varChars": false,
    "getNames": true,
    "allowTruncation": true,
    "charMultiplier": 2,
    "stripBlanks": false,
    "guessRows": 200,
    "encoding": "utf-8",
    "delimiter": ",",
    "refresh": true,
    "refreshMode": "newer",
    "successJobId": ""
  },
  "selectors": [{
    "serverName": "cas-shared-default",
    "inputCaslib": "Public",
    "outputCaslib": "Public",
    "filter": "or(endsWith(name, '.csv'),
endsWith(name, '.CSV'),endsWith(name, '.sas7bdat'),endsWith(name, '.SAS7BDAT')
,endsWith(name, '.xls'),endsWith(name, '.XLS'),endsWith(name, '.xlsx'),endsWit
h(name, '.XLSX'))",
    "settings": {}
  }]
}
```

The highlighted section is where you use JSON code to indicate the desired file type.

Here is an example of code modified to enable you to import an index file:

```
{
  "enabled": true,
  "type": "IMPORT",
  "settings": {
    "varChars": false,
    "getNames": true,
    "allowTruncation": true,
    "charMultiplier": 2,
    "stripBlanks": false,
    "guessRows": 200,
    "encoding": "utf-8",
    "delimiter": ",",
    "refresh": true,
    "refreshMode": "newer",
    "successJobId": ""
  },
  "selectors": [{
    "serverName": "cas-shared-default",
    "inputCaslib": "Public",
    "outputCaslib": "Public",
    "filter": "or (endsWith(name, '.idx'))",
    "settings": {}
  }]
}
```

BEST PRACTICE

This best practice saves you time by enabling you to use a single step to load and unload data using the CLI. To do this, you must create a flow that contains both jobs. A *job* is defined as a program that contains tasks that you need to run. A *flow* is defined as a container of jobs. Flows must contain at least one job.

You can schedule flows to run individually or run as dependent flows. The load job can be data dependent on the unload data or vice versa. Here is sample JSON code called LOADTEST that loads data:

```
{
  "enabled" : true,
  "type" : "LOAD",
  "settings" : {
    "refresh" : false,
    "refreshMode" : "newer",
    "refreshAccessThreshold" : 0,
    "varChars" : false,
    "getNames" : true,
    "allowTruncation" : true,
    "charMultiplier" : 2,
    "stripBlanks" : false,
    "guessRows" : 200,
    "scope" : "global",
    "encoding" : "utf-8",
    "delimiter" : ",",
    "successJobId" : ""
  },
  "selectors" : [ {
    "serverName" : "cas-shared-default",
    "inputCaslib" : "Public",
    "outputCaslib" : "Public",
    "filter" : "or(endsWith(tableReference.sourceTableName, '.sashdat'),
endsWith(tableReference.sourceTableName, '.SASHDAT'))",
    "settings" : { }
  } ]
}
```

Here is sample JSON code called UNLOADTEST that unloads the data:

```
{
  "enabled" : true,
  "type" : "UNLOAD",
  "settings" : {
    "unloadAccessThreshold" : "P7D",
    "successJobId" : ""
  },
  "selectors" : [ {
    "serverName" : "cas-shared-default",
    "inputCaslib" : "Public",
    "outputCaslib" : null,
    "filter" : "",
    "settings" : { }
  } ]
}
```

Now, you can put these two jobs in a flow. This example flow called TestFlow contains the two jobs. The UNLOADTEST job should start when the LOADTEST job finishes successfully.

```
{
  "version": 1,
  "id": "",
  "name": "TestFlow",
  "description": "Basic flow to load and unload data.",
  "triggerType": "manual",
  "triggers": [],
  "jobs": [
    "/jobFlowScheduling/jobs/LOADTEST",
    "/jobFlowScheduling/jobs/UNLOADTEST"
  ],
  "controlNodes": [],
  "dependencies": [{
    "target": "Unload Test",
    "event": {
      "type": "jobevent",
      "expression": "success(\"LOADTEST\") // exits with an
exit code of 0)"
    }
  }]
}
```

THE JOBS PAGE IN SAS ENVIRONMENT MANAGER

You can use the **Jobs** page in SAS Environment Manager Jobs to schedule or monitor jobs. This function is available in the left navigation pane of the GUI.

ADVANTAGES

The **Jobs** page in the SAS Environment Manager left navigation pane makes it easy to navigate your scheduled jobs. There are three applications—SAS® Data Explorer, SAS® Data Studio, and SAS® Visual Analytics from SAS® Drive—that you can use to populate the GUI to schedule jobs. Scheduling jobs using the **Jobs** page is easy because you can point and click without needing to code through the CLI. After a job is scheduled, click the **Job Monitor** tab to see whether the job started successfully. Also, you can easily grant non-administrators access to the SAS Environment Manager GUI to schedule jobs. See [SAS Viya 3.4 Administration: Identity Management](#) for information about how to add grants to a Uniform Resource Identifier (URI) using the **Rules** page in the left navigation pane. Here are some example URIs that you would use to grant access to non-administrators:

```
Schedule jobs
/jobExecution/jobRequests/*/
```

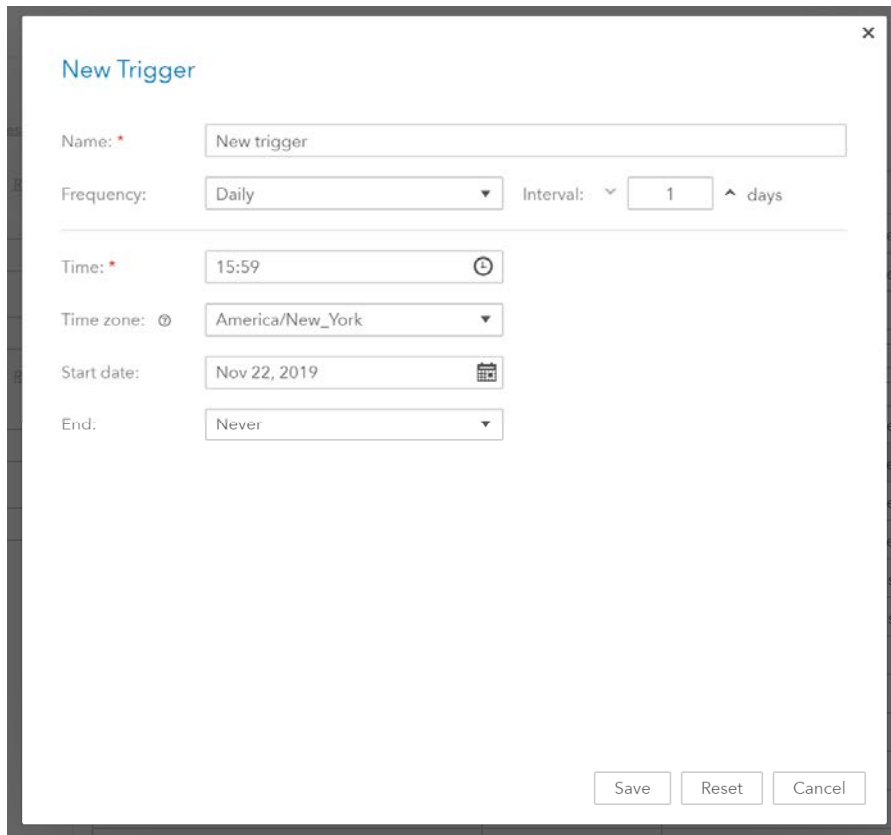
```
Edit scheduled jobs
/scheduler/jobs/**
```

```
Monitor jobs
/jobExecution/jobs/**
```

DISADVANTAGES

You can populate the **Jobs** page only from certain applications, which correspond to the following CAS actions: Manage Data works with SAS Data Explorer, Prepare Data works with SAS Data Studio, and Explore and Visualize Data works with SAS Visual Analytics. You can access these applications from the side menu on SAS Drive. The workaround is to use the CLI or another application and make sure that you use **Job execution** as the type of job. This workaround enables you to see the job in the GUI on the **Monitoring** tab of the **Jobs** page.

The **Jobs** page in SAS Environment Manager uses time-based criteria only. Display 3 shows the New Trigger dialog box where you put in the time-based criteria for your job:

A screenshot of the 'New Trigger' dialog box in SAS Environment Manager. The dialog has a title bar with a close button (X). The form contains several fields: 'Name' with a red asterisk and a text input containing 'New trigger'; 'Frequency' with a dropdown set to 'Daily'; 'Interval' with a dropdown arrow and a text input containing '1', followed by an up arrow, a text input containing 'days', and a down arrow; 'Time' with a red asterisk and a time input containing '15:59' with a clock icon; 'Time zone' with a globe icon and a dropdown set to 'America/New_York'; 'Start date' with a calendar icon and a date input containing 'Nov 22, 2019'; and 'End' with a dropdown set to 'Never'. At the bottom right are three buttons: 'Save', 'Reset', and 'Cancel'.

Display 3. Dialog Box Where You Input Time-Based Criteria for the Job

If you want the job to be dependent on something other than a time event, then you can use the CLI to create a flow. From the flow, you can set job dependencies on other jobs. The code in the [Best Practice](#) section of CAS table state management contains a snippet of dependencies. The [Command Line Interface](#) section of this paper also has helpful information.

For running jobs, the token has a 12-hour limit. The token is not refreshed once a user is logged in to SAS Environment Manager. For example, if you run a job at 8 a.m. and run another job as the same user at 9 p.m., the second job is likely to fail because the token has expired. The workaround is to use the SAS Job Execution Web Application, which uses a fresh token for each job.

Note: The **Monitoring** tab for SAS Environment Manager has a limitation. It shows only the status of jobs that are starting, not jobs that are in progress or finished.

BEST PRACTICE

This best practice gives you flexibility for how you use the time trigger and also saves time. When you use the **Jobs** page in the SAS Environment Manager left navigation pane, there are multiple ways to schedule jobs using various time patterns. You can create different triggers, each with an optional start and end date. For example, if you want to schedule a job to run three weeks, skip one week, and then run three weeks, you could write it as three triggers in the New Trigger dialog box:

Start on X , run one week, skip 3 weeks
Start on $X+1$ week, run one week, skip 3 weeks
Start on $X+2$ weeks, run one week, skip 3 weeks

For this pattern, you would need to use the **weekly** TimeEvent trigger, specify **every weekday** in the daysOfWeek array, and specify **skipCount=3**.

Instead of specifying days, you can also run the job with list of dates.

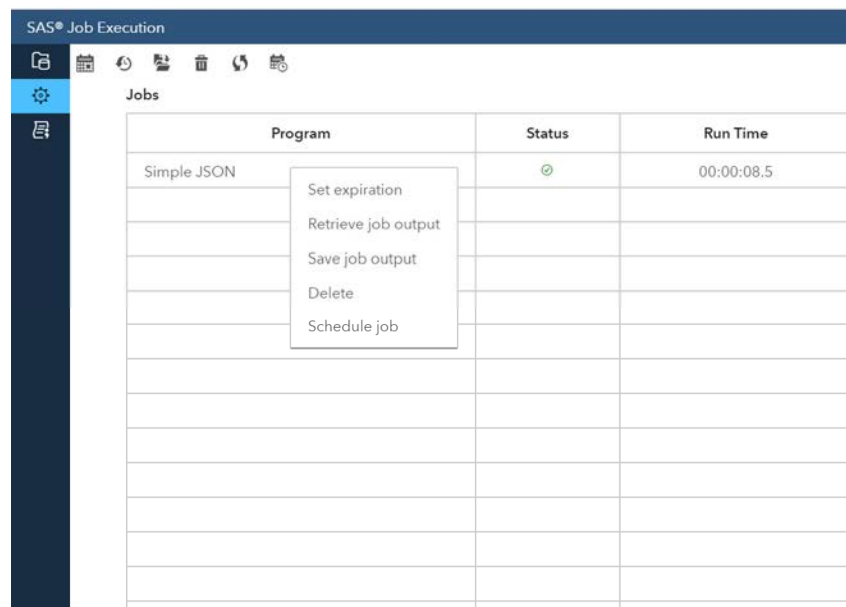
SAS JOB EXECUTION WEB APPLICATION

SAS Job Execution Web Application is comparable to a SAS®9 stored process. You can access it from a web browser with the SASJobExecution extension.

ADVANTAGES

When you submit a job in the SAS Job Execution Web Application, it uses a fresh 12-hour token unlike the **Jobs** page in SAS Environment Manager. A REST request has a token on it, which is based on whomever authenticated the application that is driving the REST request. This is an important feature for long-running jobs. It can increase the time-out value so that jobs can run longer. This setting is published in the [Job Execution Time-out](#) section of SAS® *Job Execution Web Application 2.1: User's Guide*.

This application can schedule a wider variety of jobs than the previous two methods discussed in this paper. From the SAS Job Execution Web Application, you can create jobs by right-clicking and selecting **Schedule Job** as shown in Display 4.



Display 4. Page Where You Can Schedule Your Job

This action launches another web browser tab with the **Jobs** page from SAS Environment Manager. On this tab, you can schedule and monitor the job that you created in the SAS Job Execution Web Application.

DISADVANTAGES

The entry point to the SAS Job Execution Web Application is separate from SAS Drive. You must either launch a separate web browser or you need to enter another URL to get to the SAS Job Execution Web Application.

BEST PRACTICE

This practice can help improve the performance of long-running jobs. For longer-running jobs, the fresh 12-hour token is key. If you have many long-running jobs, SAS recommends running half of your jobs on the SAS Job Execution Web Application and the other half from the **Jobs** page in SAS Environment Manager. This practice could help improve the performance of running jobs on your server. However, if your environment performs better running all the jobs through one application, then you should choose the best one for your system's needs.

SAS STUDIO

SAS Studio is web-based programming environment for creating SAS code that you can access from SAS Drive.

ADVANTAGES

If you want to use SAS®9 jobs in the SAS Viya environment, then this method is the one that you should use. SAS Studio enables you to migrate SAS®9 jobs into SAS Viya for scheduling. For details about how to use this method, see the [SAS Viya 3.4: Schedule SAS Studio 5.1 \(Studio V\) Program Files](#) post in SAS Communities.

DISADVANTAGES

The main drawback to this method is that you must use both the SAS Job Execution Web Application and SAS Environment Manager. This method requires a combination of programming and pointing and clicking. Currently, there is no way around using two applications.

BEST PRACTICE

SAS Studio is an application that can efficiently load a large amount of data from external data sources. Instead of manually loading the data, you can schedule the load using the link in the **Advantages** section above.

COMMAND LINE INTERFACE

The CLI is an administrative tool for programmers, which can be accessed directly from the operating system on which the CLI is installed.

ADVANTAGES

The CLI offers the most flexibility and customization of these methods. It provides options to schedule and unschedule flows, supports flows that contain jobs, and supports job dependencies.

DISADVANTAGES

Familiarity with JSON is required to use this method. If you are not a JSON programmer, there are several other scheduling methods that you can use, which are described in this paper.

The CLI has a 12-hour authentication limitation for flows, which is documented in the [Command-Line Interface: Preliminary Instructions](#) section of *SAS® Viya® 3.4 Administration: Using the Command-Line Interfaces*. Therefore, flows that run longer than 12 hours will not complete using this method. For longer-running jobs, it is best to use the SAS Job Execution Web Application tool.

BEST PRACTICE

This method provides more flexibility with manipulating flows. There are different actions and commands to schedule and unschedule flows. Display 5 shows a sample of usage and commands of job flows for the CLI.

```
[sas@viya bin]$ ./sas-admin job flows
NAME:
  sas-job-cli flows - Update Job Flow Scheduling service objects

USAGE:
  sas-job-cli flows command [command options] [arguments...]

COMMANDS:
  create          Create a flow
  delete          Delete a flow
  execute         Execute a flow
  generate-template Generate a template for a flow
  help, h         Shows a list of commands or help for one command.
  list            List flows
  list-history    List the flow's' history
  reschedule      Reschedule a flow
  schedule        Schedule a flow
  show            Show a flow
  show-history    Show the history for a flow instance
  unschedule      Unschedule a flow
  update          Update a flow

OPTIONS:
  --help, -h  Shows help.

[sas@viya bin]$
```

Display 5. Sample of Usage and Commands of Job Flows for the CLI

Here is an example of how to use the CLI to schedule a flow:

1. Before you attempt to use the CLI, make sure that you have a profile that is authenticated based on the [Command-Line Interface: Preliminary Instructions](#) section of *SAS® Viya® 3.4 Administration: Using the Command-Line Interfaces*.
2. To list all the jobs in your SAS Viya environment, run the `./sas-admin job`. The output of the job requests list shows that `aabd293c-4a5f-43b4-ab66-9c908c19723a` is the ID used in the example JSON code for jobs below:

```
sas> ./sas-admin job requests list
Id                                     Name
Version  Description
bda2ce72-feb2-4d3f-96d7-24908c5027a4  BINARY_BACKUP_SCHEDULE
3      This job is created by sas.deploymentBackup to run binary
backup first Saturday at 5AM every month
```

```

625f1059-67ba-4e2c-9377-68032eb4023d  DEFAULT_BACKUP_SCHEDULE
3      This job is created by sas.deploymentBackup to run default
backup every Sunday at 1AM.
6df962fb-6b26-4f14-b46c-181ba8e62781  Sample: Import cas-shared-
default Public data 3      Imports csv, sas7bdat, and excel
files to sashdat files in cas-shared-default:Public
09459f84-1feb-4e28-83ca-96be5c36452c  Sample: Load cas-shared-
default Public data 3      Loads data in cas-shared-
default:Public
35793268-d12c-4cd0-934e-77f0d3608757  Sample: Unload cas-shared-
default Public data 3      Unloads infrequently accessed data in
cas-shared-default:Public
0b059266-667b-4701-alca-f60df6a182f6  Simple JSON
3      JSON file created using PROC JSON.
aabd293c-4a5f-43b4-ab66-9c908c19723a  Simple JSON
3      JSON file created using PROC JSON.

```

3. To list all the schedulers in your SAS Viya environment, run the sas-admin job. The output of job schedulers list shows that 6e573f08-7498-443a-bd7c-3b0101ef5d75 is the ID used in the example JSON code for scheduleID:

```

sas> ./sas-admin job schedulers list
Id                                     Name
Version  Description
6e573f08-7498-443a-bd7c-3b0101ef5d75  Default SAS Job Flow Scheduler
1      SAS Job Flow Scheduler

```

4. Here is an example of how to create a flow using a time-event trigger. You can use Notepad to create the JSON code.

```

{
  "name": "opaque",
  "description": "Test object with time event opaque",
  "triggerType": "event",
  "triggers": [{
    "type": "timeevent",
    "active": true,
    "event": {
      "recurrence": {
        "type": "daily"
      },
      "hours": "12",
      "minutes": "0",
      "duration": "1"
    }
  ]},
  "triggerCondition": "any",

```

```

    "jobs": [
        "/jobFlowScheduling/jobs/aabd293c-4a5f-43b4-ab66-9c908c19723a"
    ],
    "dependencies": [],
    "schedulerId": "6e573f08-7498-443a-bd7c-3b0101ef5d75"
}

```

This code is contained in a file called createflow.json and is saved on the UNIX operating system.

5. In the CLI, the **sas-admin** command creates the flow in this example:

```

sas> ./sas-admin job flows create --file-in ~urpolo/createflow.json
POST /jobFlowScheduling/flows from
"/r/ge.unx.sas.com/vol/vol620/u62/urpolo/createflow.json completed."

sas> ./sas-admin job flows list
Id                               Name      Version  Description
86a2551e-84e6-4421-a013-3145a28bfce2  opaque   <nil>    Test object
with time event opaque

sas> ./sas-admin job flows schedule --id 86a2551e-84e6-4421-a013-3145a28bfce2
Scheduled the flow "86a2551e-84e6-4421-a013-3145a28bfce2"

```

In this example, Opaque is a schedule flow using a time event as a trigger.

CONCLUSION

As outlined in this paper, there are numerous options to meet your needs for scheduling data. Scheduling options exist for specific tasks in SAS Viya as well as flexible and customizable options for general job and flow submission. This paper has detailed the advantages, disadvantages, and best practices for each method, enabling you to create the most effective and efficient environment for your needs. SAS Viya makes it easy to schedule your jobs. Now, it's up to you to decide which is the best method for you.

REFERENCES

SAS Institute Inc. 2018. "Job Execution Time-out." *SAS® Job Execution Web Application 2.1: User's Guide*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/?docsetId=jobexecug&docsetTarget=plct9uz15c7omun1t2zy0gxhlqlc.htm&docsetVersion=2.0&locale=en#n111c94xox7r2vn1bzyp8on38so1

SAS Institute Inc. 2018. *SAS® Viya® 3.4 Administration: Data*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/api/docsets/caldataamgmtcas/3.4/content/caldataamgmtcas.pdf

SAS Institute Inc. 2018. *SAS® Viya® 3.4 Administration: Identity Management*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/api/docsets/calids/3.4/content/calids.pdf

SAS Institute Inc. 2018. *SAS® Viya® 3.4 Administration: Jobs*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/api/docsets/caljobs/3.4/content/caljobs.pdf

MKQueen. "SAS Viya 3.4: Schedule SAS Studio 5.1 (Studio V) Program Files." *SAS Communities Library*. 2019. Available at communities.sas.com/t5/SAS-Communities-Library/SAS-Viya-3-4-Schedule-SAS-Studio-5-1-Studio-V-Program-Files/ta-p/550219

SAS Institute Inc. 2018. "Scheduling Command Line Interface. *SAS® Viya® 3.4 Administration: Jobs*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/?docsetId=caljobs&docsetTarget=plg6z2wnbs791un1n5556kigta2h.htm&docsetVersion=3.4&locale=en

SAS Institute Inc. 2017. "Scheduling: How to (Command Line Interface). *SAS® Viya™ 3.3 Administration: Scheduling*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/?docsetId=calscheduling&docsetTarget=pl08p296wpbvkn1cl76xlyoh8vt.htm&docsetVersion=3.3&locale=en

ACKNOWLEDGMENTS

Thanks to Russell Gonsalves, Jeff House, Bob Maggio, Paul Polo, Rayna Rowell, Casey Thompson, George Wilder, Erica Williams, Randy Williams, and Connie Wu for their support and knowledge.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ursula H. Polo
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: support@sas.com
Web: support.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.