Paper SAS3026-2019

The Power of REST API and CLI with SAS® Visual Analytics 8.3 – Automate Your Tasks with a Batch Job

Rain Kang, SAS Institute Inc., Cary, NC

ABSTRACT

Many customers expect batch and programming approaches to manipulate SAS® Visual Analytics reports in the areas of report creation, customization, localization, and distribution. SAS provides the power to meet these requirements. In this Paper, we demonstrate how to use the REST API and Command-Line Interface (CLI) with SAS® Viya® 3.4 to accomplish business tasks with typical scenarios. You will learn how to create or edit Visual Analytics (VA) reports programmatically, how to localize a VA report based on the standard report, how to customize the email content before distributing the reports in batch, how to change the report distribution recipients automatically with a batch job, and finally how to kick off the distribution right after the data is refreshed automatically. This paper will be helpful for both SAS consultant and customers who work in area of SAS® Visual Analytics.

INTRODUCTION

One of the ways that SAS is more open is it's use of several multiple Batch Interfaces–these are methods of accessing SAS Viya components and services via REST APIs from command-line or programming language interfaces. This feature makes it possible and easy to automate some boring repetitive tasks in Visual Analytics.

SAS Viya REST APIs are organized around REST principles. The APIs are based on resource-oriented URLs, all of these features help you integrate the APIs into your applications and scripts in a standard way, making it easy to integrate the capabilities of SAS Viya into your business processes or to extend and customize SAS Viya to meet specific requirements.

The SAS Viya command-line interface (CLI) allows administrators to perform numerous administrative tasks in batch and automate more complex administration tasks. The CLIs provide a simplified interface to the SAS Viya REST services. They abstract the functionality of the REST services, allowing an administrator to enter commands on a command line and receive a response back from the system. If the CLIs do not surface some functionality, calls to the REST API can be made to fill in the gaps.

This paper assumes that you have knowledge about the basic concepts of CURL and SHELL, with the demonstrate scenario as Figure 1 you will learn how to use REST API and CLI to access the Visual Analytics for automation purposes.

In Figure 1, you are given the task of creating a sales report for different divisions: the task to create localized division report, the task to customize report distribution like the report recipients and email subjects, and the task to schedule the distribution in daily recurrence..., and so on.

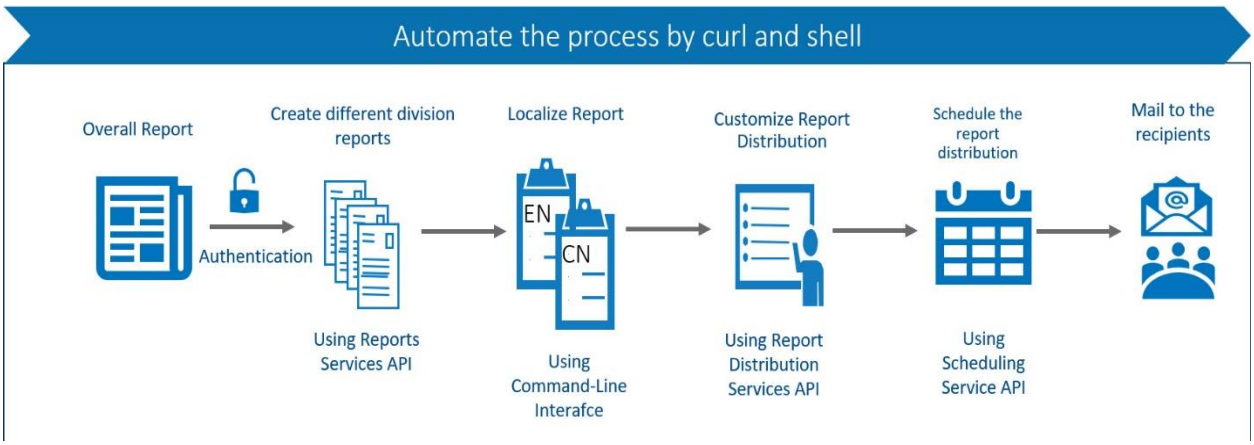

Figure 1. Demonstrate Scenario in this Paper

The scenarios in Figure 1 are presented in Curl and Shell, but all these steps could be written in Python, Java, Lua, and other languages that contain the library for executing HTTP requests and parsing/generating JSON data.

This paper provides example usage for some APIs and CLIs but does not document all possible usages. To learn more, please refer to the references part of this paper.

CREATE AND UPDATE REPORT

SAS Viya REST APIs typically require authentication for all operations. Authentication is a means to verify the identity of the user or agent that is making the REST API request. So before manipulating the report from the REST API, the first challenge for you is to get the authorization token from SAS Logon Manager, which handles the authentication with an OAuth2-baased service.

GETTING AUTHORIZATION TOKEN IN SAS VIYA

Here is the normal sequence to get authorization token in SAS Viya, these steps only need to be done once within the validity period for token

1. Use the Consul token to obtain an ID Token to register a new Client ID
2. Use the ID token to register the new client ID and secret (with a validity period)
3. Acquire the access OAuth token of our Client ID
4. Call the SAS Viya microservice using the access token for the authentication.

Figure 2 is shows the process for getting authentication tokens.

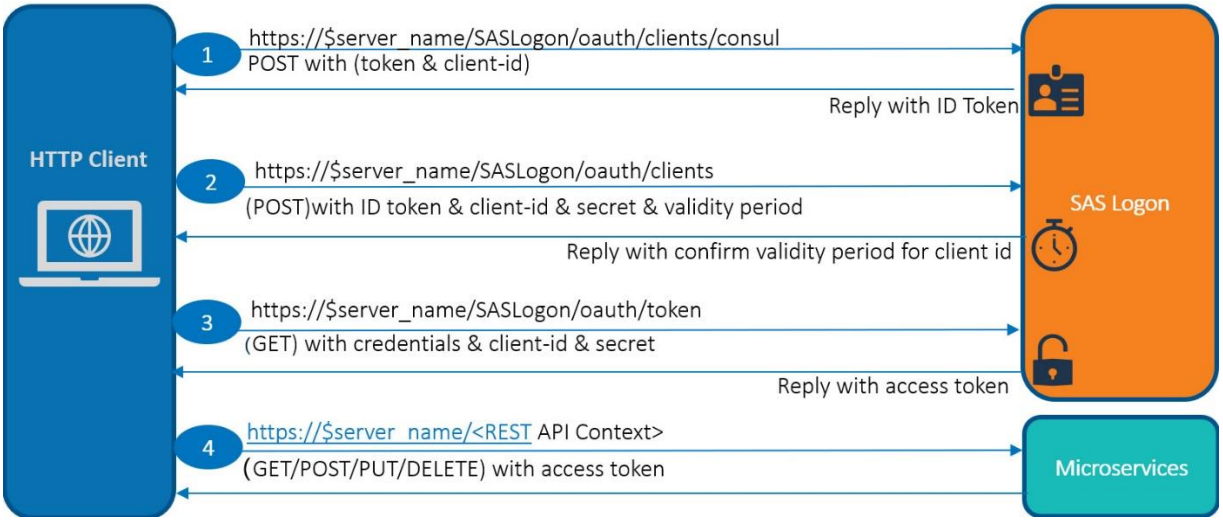

Figure 2. Process of Getting Authentication Tokens

Below are the detailed steps for getting the authentication token.

1. Use the Consul token to obtain an ID Token to register a New Client ID

   The Consul Client Token is available in your SAS Viya configuration directory structure in a file named client.token in the /etc/SASSecurityCertificateFramework/tokens/consul/default. The Consul Client Token is unique to each deployment.

   You have to copy the Consul Client Token and use it in the next query as the value of X-Consul-Token to obtain a client authorization token to register a new client.

   ```
   export consul_token=`sudo cat
   /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/def
   ault/client.token`
   ```

   Using the Consul Client Token, you can now pass through the process to Get Authorization Token from the SASLogon Service. Please note the server_name variable is resolved to your SAS Viya host name.

   ```
   export server_name=<your Viya host name>
   export client_token=`curl -sk -X POST
   "https://$server_name/SASLogon/oauth/clients/consul?callback=false&serviceI
   d=test" -H "X-Consul-Token:$consul_token " | cut -d '"' -f 4`
   ```

2. Use the client authorization token to register the new client

   Table 1 describes the client attributes used in the curl command that follows.

Attribute	Description
client_id	unique ID assigned to this client
client_secret	secret (password)
scope	List of scopes the client should receive by default in access tokens (* is the wildcard for all scopes)

Attribute	Description
authorities	scopes that access tokens should have when using a client_credentials grant_type
authorized_grant_types	ways by which the client can obtain access tokens, refer to the OAuth 2.0 specification
access_token_validity	time (in seconds) that access tokens should be valid for when issued

Table 1. Description of client attributes that are used to register the new client

```
curl -sk -X POST "https://$server_name/SASLogon/oauth/clients" -H "Content-
Type: application/json" -H "Authorization: Bearer $client_token " -d
'{"client_id": "james.client", "client_secret": "james.secret", "scope":
["openid", "*"], "resource_ids": "none", "authorities": ["uaa.none"],
"authorized_grant_types": ["password"],"access_token_validity": 3600000}'
```

3.Get the authorization token

```
export access_token=`curl https://$server_name/SASLogon/oauth/token -H
"Accept: application/json" -H "Content-Type application/x-www-form-
urlencoded" -d "grant_type=password&username=james&password=******" -u
"james.client:james.secret" | cut -d '"' -f 4`
```

Once the token is generated and exported, you can use it as parameter in the subsequent commands.

CREATE NEW REPORT

You are assigned the task of making the daily sales report available to different division, such as education and consumer division. You have similar data sets for each department in the company; the substituted data source is compatible with how the original data source is used in the report, and the report should be periodically replicated for new or updated data sets. Figure 3 shows the sample that creates the sales report for both the education division and the consumer division based on the overall sales report.

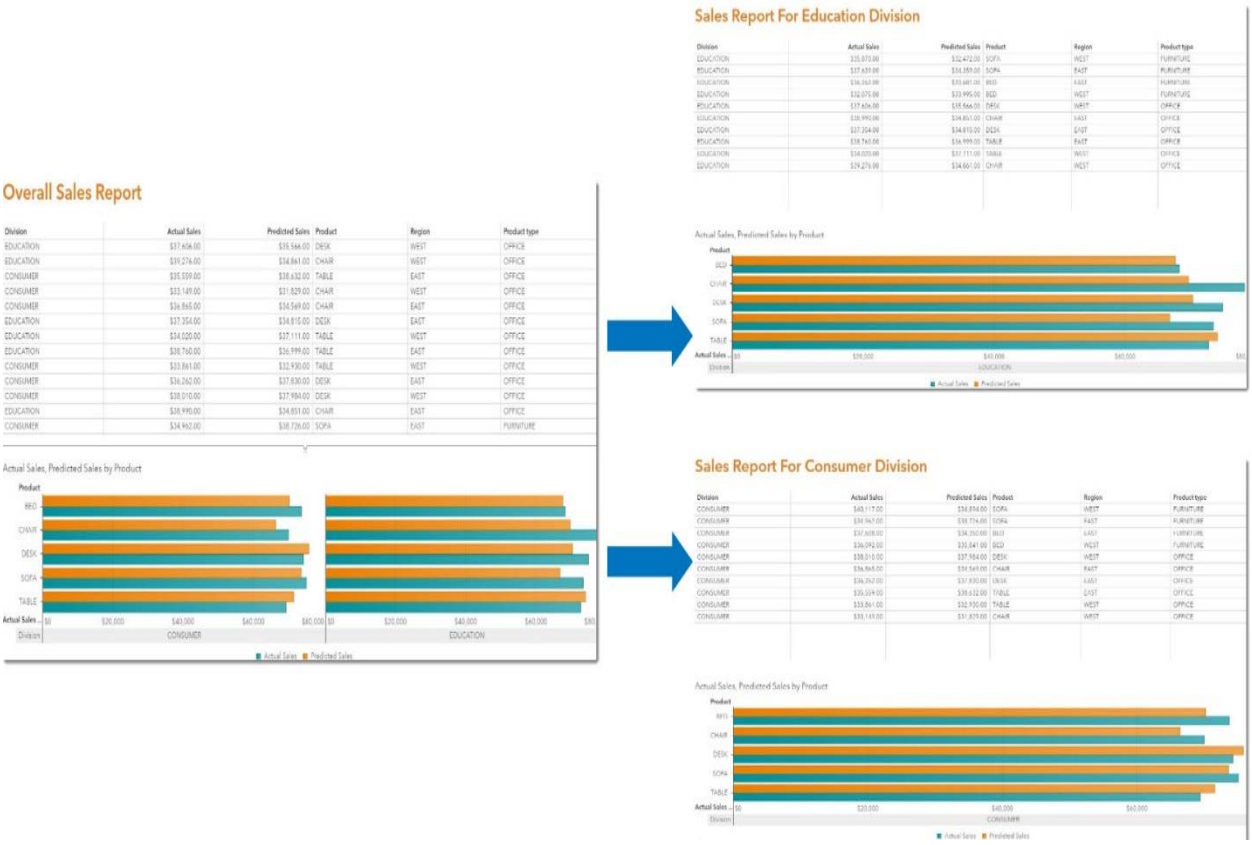

Figure 3. Create a Different Division Report Based on the Overall Report

In following step, you will learn how to use the reports service API to create the report based on the standard report and then use an automated process to replicate it for each department.

Note: If you would like to go through the samples you should have the data sets and base report available.

About Reports Service

The Reports API provides persistent service of reports and report content, as well as validation for report content. Typical consumers are those who treat the report as an object, such as those who want to move or rename the object, and those who read, write, and update the content, such as report editors and viewers.

A report is identified uniquely by its ID and has sub-components which are its content and, optionally, its states. A report can have only one content, which is a document that can be presented in either XML or JSON format but is saved only in XML format. Report content is stored separately from the report. A new report has no content. Storing report content overwrites the previously stored content.

Creating a New Report

In this step, you create the different division report under 'public' folder. Firstly, you need to get the public folder ID with below URI:

Method	URI	Task
GET	/folders/folders?filter=eq(name,' ')	Getting the folder content based on name

About Filtering:

Collections can be sorted and filtered using the ?sortBy= and ?filter= query parameters.

Filtering and sorting can use the following members of the Report: name, creationTimeStamp, createdBy, modifiedBy, modifiedTimeStamp, type, and description.

```
export folder=`curl -sk -X GET
"https://$server_name/folders/folders?filter=eq(name,'Public')" \
     -H "Accept: application/json"\
     -H "Authorization: Bearer $access_token " `
```

The response string that saved to folder variable contains much information, you need to split the ID from it.

```
export folder_id=`echo ${folder#*'"'ID'"':'"'} | cut -d'"' -f1 `
```

- After getting the folder ID, now you can pass it to below curl command.

Method	URI	Task
POST	/reports/reports?parentFolderUri	Create report under the specified folder

```
export temp_str=`curl -sk -X POST
"https://$server_name/reports/reports?parentFolderUri=/folders/folders/$fol
der_id" \
 --include \
     -H "Accept: application/json" \
     -H "Authorization: Bearer  $access_token " \
      -H "Content-Type: application/json" \
     --data '{"name":"eduReport"}'`
```

Note: Curl parameter –include must be appended in your request, this parameter forces the response to generate the HTTP-header which contains the ETag.

SAS Viya REST APIs use conditional operations to manage concurrent updates to resources.

An Etag (entity tag) is an opaque string that marks (or "tags") the current state of a resource. The entity tag changes each time the resource is updated and are returned in the ETag response headers, shown in Table 2, from supporting operations. SAS APIs that use entity tags must pass the If-Match request header using the value of the entity tag received from the latest operation on the resource.

Header	Description	Example
ETag	An identifier for a specific version of a resource. RFC 7232. Note: SAS typically uses a single strong ETag Note: the double quotation marks are part of the ETag syntax.	ETag: "xyzzy"
Last-Modified	The last modified date for the requested object	Last-Modified: Wed, 14 Mar 2018 19:43:31 GMT
Location	The Location response header is used both whenever a redirect occurs and when a resource is created.	Location: /folders/folders/5c20b80c-def
Content-Type	The Media type of the body of the request or response.	Content-Type: application/vnd.sas.report
Content-Item-Type	(SAS specific) For collection and composite resources, the media type of the contained item(s). Infrequently used as Collections member accept is typically sufficient. Used for content negotiation, especially where the specific collection supports more than one media type,	Accept-Item: application/vnd.sas.data.table

Table 2. Common response headers list

Since the new report ID and ETag will be used in following steps, you need to get them from the response header with below command:

```
export new_report_id=`echo ${temp_str#*'"'ID'":'"'} | cut -d'"' -f1`
export etag=`echo ${temp_str#*ETag:} | cut -d'"' -f2`
```

Note: The ETag can be obtained at any time with the report ID:

```
export etag=`curl -sk -X HEAD
"https://$server_name/reports/reports/$new_report_id#standard" --
include -H "accept: application/vnd.sas.report+json" -H "Authorization:
Bearer $access_token" | grep ETag | cut -d'"' -f2`
```

UPDATE THE REPORT

Now an empty report named eduReport is generated under public folder. Next you need to write report content to it. Since new report will be based on the overall sales report, you first need to get the overall report content and then update it as-needed.

Getting the report ID for overall sales report

1. Getting the report ID for overall sales report

```
export temp_original_report=`curl -sk -X GET
"https://$server_name/reports/reports?filter=eq(name,'overall_sales_rep
ort')"  \
     -H "accept: application/json"  \
     -H "Authorization: Bearer $access_token" `
```

```
export original_report_id=`echo ${temp_original_report#*'"'ID'"':'"'} |
cut -d'"' -f1`
```

2. Writing the overall sales report content to file original_reportcontent.txt, and this file will be used as input in next step.

Method	URI	Task
GET	/reports/reports/$original_report_id/content	Get the content of specified report

```
curl -sk -X GET
"https://$server_name/reports/reports/$original_report_id/content" -H
"accept: application/vnd.sas.report.content+json" -H "Authorization:
Bearer $access_token " -o original_reportcontent.txt
```

Note: The curl option – k means turn off the verification of the certificate for curl.

Writing the report content to new report

1. When you createthe education division report, the datatable 'PRDSALE' will be replaced with 'PRDSALEEDU', and the report title also will be replaced. Below is the script to update the original_reportcontent.txt.

```
#change the dataset name for report.
old_data_table=\"PRDSALE\"
new_data_table=\"PRDSALEEDU\"
sed -i "s/$old_data_table/$new_data_table/g" original_reportcontent.txt

#change the report title.
old_report_name="\"Overall Sales Report\""
new_report_name="\"Education Division Sales Report\""
sed -i "s/$old_report_name/$new_report_name/g"
original_reportcontent.txt
```

2. Write the content to education division report.

```
curl -sk -X PUT
"https://$server_name/reports/reports/$new_report_id/content" \
     -H "accept: application/vnd.sas.report.content+json" \
     -H "content-type: application/vnd.sas.report.content+json" \
     -H "If-Match: $etag" \
     -H "Authorization: Bearer $access_token" \
   -d @original_reportcontent.txt
```

LOCALIZE THE REPORT

Now you get the division report in English locale. But if you also like to send the report in other locales, how can you do it? Starting in the 8.3 release, you can use the SAS Viya Command-Line Interface (CLI) to complete this task, it provides the interfaces to export and import the translation worksheets for reports. Table 3 lists the common interfaces for CLI.

Interface	Scope
admin	The top-level administrative command-line interface that is used to initialize, authenticate, and execute other plug-ins.
audit	Gets SAS audit information.
authorization	Gets general authorization information and manages rules.
backup	Manages backups.
restore	Manages restore operations.
cas	Manages CAS administration and authorization.
configuration	Manages the operations of the configuration service.
reports	Manages SAS Visual Analytics 8.2 reports.
tenant	Manages tenants in a multi-tenant deployment.
transfer	Promotes SAS content.
identities	Gets identity information and manages custom groups.
folders	Gets and manages SAS folders.

Table 3. Common interfaces for CLI

Preliminary steps required to use the CLI

You can download the admin CLI directly from the SAS Support website and install the plug-ins that you need. Alternatively, the admin CLI, along with the plug-ins, is installed on the SAS Viya server during deployment, you can run the admin CLI directly from this location on the SAS Viya server:

Linux: /opt/sas/viya/home/bin Windows: \Program Files\SAS\Viya\bin

There are two preliminary steps required to use the command-line interface you need to create a profile and authenticate.

```
#Create a profile, input service endpoint: http://host:port, then input
#json and Yes.

/opt/sas/viya/ home/bin/sas-admin profile init

#Authenticate with your credentials
/opt/sas/viya/home/bin/sas-admin auth login
```

Preparing your translation worksheets before import

Before localizing your report, you need to prepare your translation sheet. To do this, determine what reports need to be translated and into which languages the reports need to be translated. You also need to identify the base language in which the reports were created. The translation worksheets that are exported will contain strings to translate, and these strings are written in the base language in which the report was created. You must save the translation worksheets using the UTF-8 encoding.

In this sample, the original report is in English locale and you want to translate it to Chinese.

1. First you need to export the English locale template from original report.

```
/opt/sas/viya/home/bin/sas-admin reports translations export --report-
id $new_report_id --output-location /tmp --report-locale zh-CN
```

Figure 4 shows the explanation for the syntax of the command to export the translation worksheet

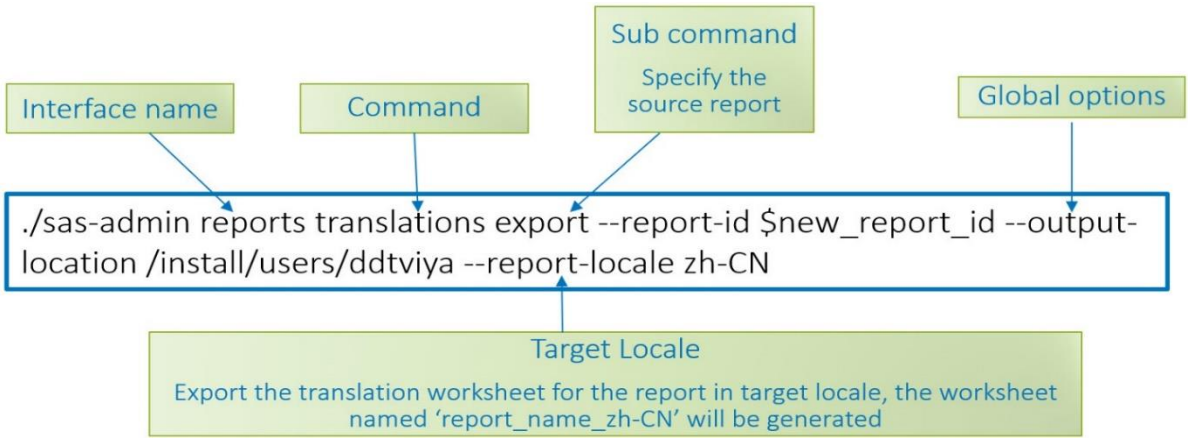

Figure 4. Export Translation Worksheet Command

2. Translated the strings in exported template to the target language.

   Figure 5 shows a sample display for a translated sheet.

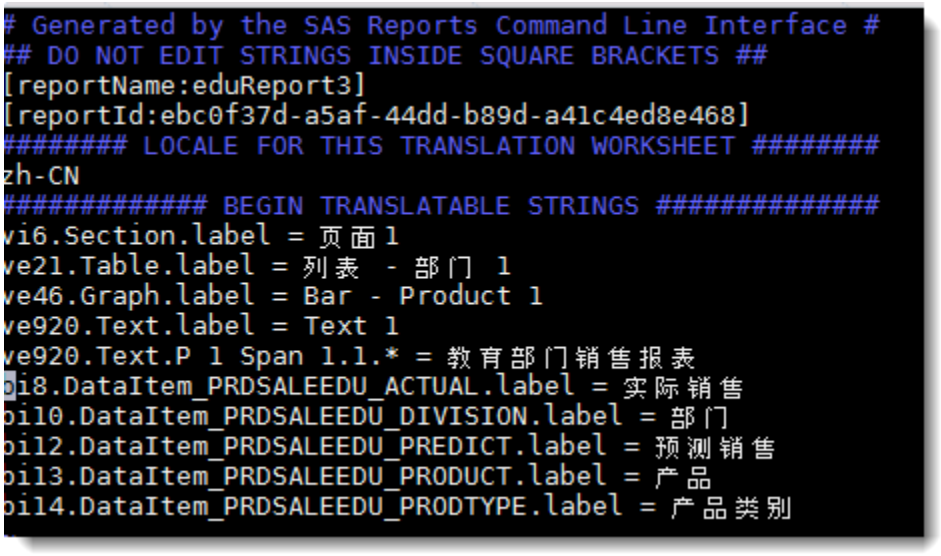

Figure 5. Sample Translated Sheet for Simplified Chinese

Importing translation worksheets

Import the translated worksheet to the newly created division report.

```
/opt/sas/viya/home/bin/sas-admin reports translations import --source-
file /tmp/eduReport zh-CN.reports
```

Figure 6. Education Division Report After Importing the Translated Sheet

CREATE AND CUSTOMIZE REPORT DISTRIBUTION

Next you need to complete the assigned task of making the daily sales report to the education and the consumer department. The report distribution service will be used to accomplish this.

About Report Distribution Service

The Report Distribution Service is used to construct distribution requests, which are descriptions of a set of reports to be emailed to recipients; the email contains links and an optional PDF report. This service schedules distribution requests in the Scheduler Service on a one-time or recurring basis. The tasks performed by the distribution service include:

- Managing distribution requests.
- Scheduling or updating the scheduling of a distribution request in the Scheduler Service.
- Running the report job to produce email with links and optional printed output attachments

This service schedules distribution requests in the Scheduler Service in order to assist the Visual Analytics client. The reports produced are customized to the recipients, showing only the data that the recipient is allowed to see.

Customizing the report distribution information

Distribution requests are the primary resource for client use to create and manipulate the details of the distribution request.

The media type is application/vnd.sas.report.distribution.request . It's a collection of information needed to generate reports with the recipients' view of the data and email PDF copies of those reports to the recipients. Below is a template of the JSON representation of this media type. In this sample you can save it to distribution_request.txt.

```
{
  "name": "<distribution_name>",
  "recipients": [
    "/identities/users/<username>"
  ],
  "reportRequests": [
     {"reportUri": "/reports/reports/<reportid>"}
  ],
  "subject": "<distribution_subject>",
  "body": "<distribution_body>",
  "includePdf": "true",
  "includeScreenplay": "false",
  "version": 2
}
```

Below is the shell script to replace the strings in request template file.

```
sed -i "s/<username>/new_username/g" distribution_request.txt
sed -i "s/<distribution_name>/distribution for education disvision/g"
distribution_request.txt
sed -i "s/<reportid>/$new_report_id/g" distribution_request.txt
sed -i "s/<distribution_subject>/Sales report for education division/g"
distribution_request.txt
sed -i "s/<distribution_body>/Dear colleagues, this is the sales report
for education division. /g" distribution_request.txt
```

Creating the report distribution

Now you can create the distribution with below curl command and get the distribution ID. The distribution ID will be used for scheduling in next section.

Method	URI	Task
POST	/reportDistribution/distributionRequests	create a distribution request

```
 export distribution_id=`curl -sk -X POST
"https://$server_name/reportDistribution/distributionRequests" \
 -H "accept: application/vnd.sas.report.distribution.request+json" \
 -H "content-type:
application/vnd.sas.report.distribution.request+json" \
 -H "Authorization: Bearer $access_token" \
 -d @distribution_request.txt | cut -d'"' -f6`
```

Once the distribution is created, if you would like to kick off the distribution immediately without a schedule, you can use below command.

Method	URI	Task
POST	/distributions/{distributionRequestId}	Executes a distribution request.

```
curl -sk -X POST
"https://$server_name/reportDistribution/distributions/$distribution_id
" \
-H "Authorization: Bearer $access_token" \
-H "accept: application/json"
```

After you executed above command, the recipients will receive the email with report, as shown in Figure 7.

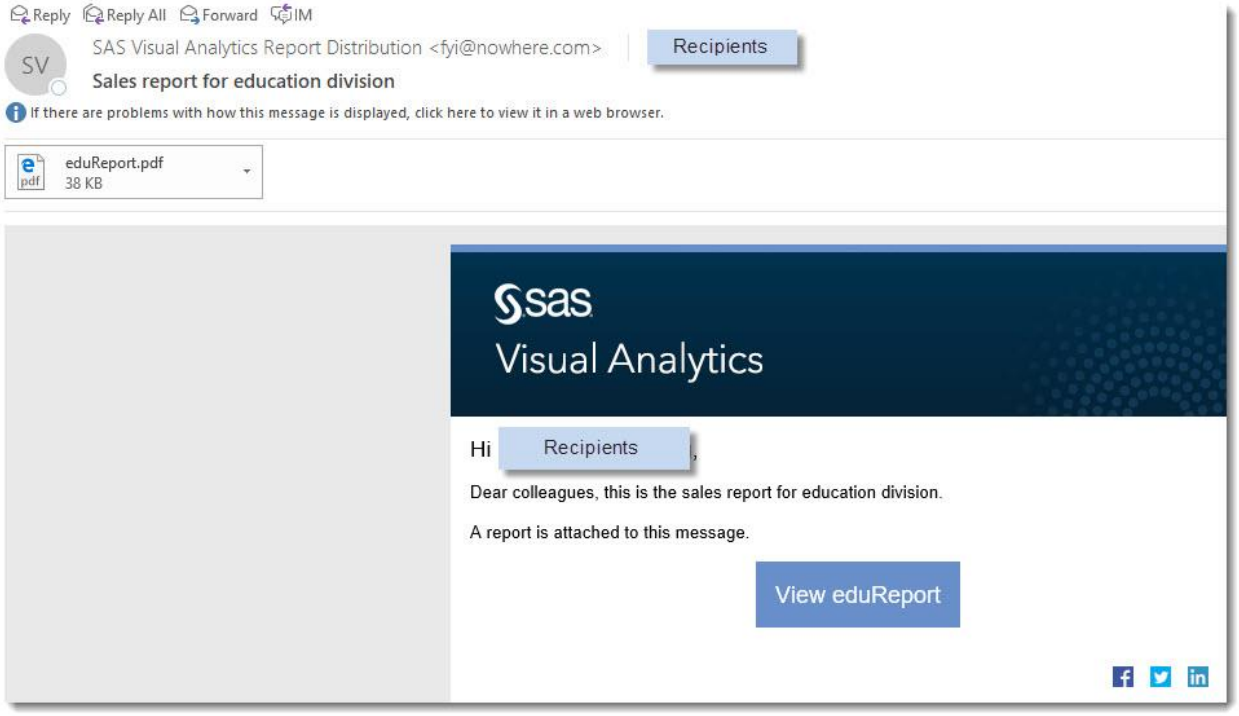

Figure 7. Customized Email with the Report

SCHEDULE THE CREATED DISTRIBUTION

After creating the distribution, the next step is to schedule the new created distribution in daily recurrence. The report distribution service provides the endpoints to schedule a distribution request and manage scheduled distribution requests in the Scheduling Service.

About Scheduling Service

The Scheduling Service schedules jobs to be executed according to a time schedule. At a minimum, a job is a combination of a name, a request, and a collection of triggers. A trigger is a set of conditions that are associated with a job in the Scheduler service and determines when and how often a job should be run. A trigger can run immediately, run one time, or run on a recurring basis.

A trigger comes in four different flavors:

- immediate - runs immediately

- cron - provides a cron schedule string to represent invocation periods.
- simple - allows for simple time interval specifications.
- timeEvent - allows for more complex time interval specifications.

There are Seven options for recurrence: minutely, hourly, daily, weekly, monthly, yearly, and on a specified date.

Defining the trigger of report distribution

As we now know, it is easy to define a schedule through UI. Figure 8 shows the screenshot for creating a schedule in daily recurrence.

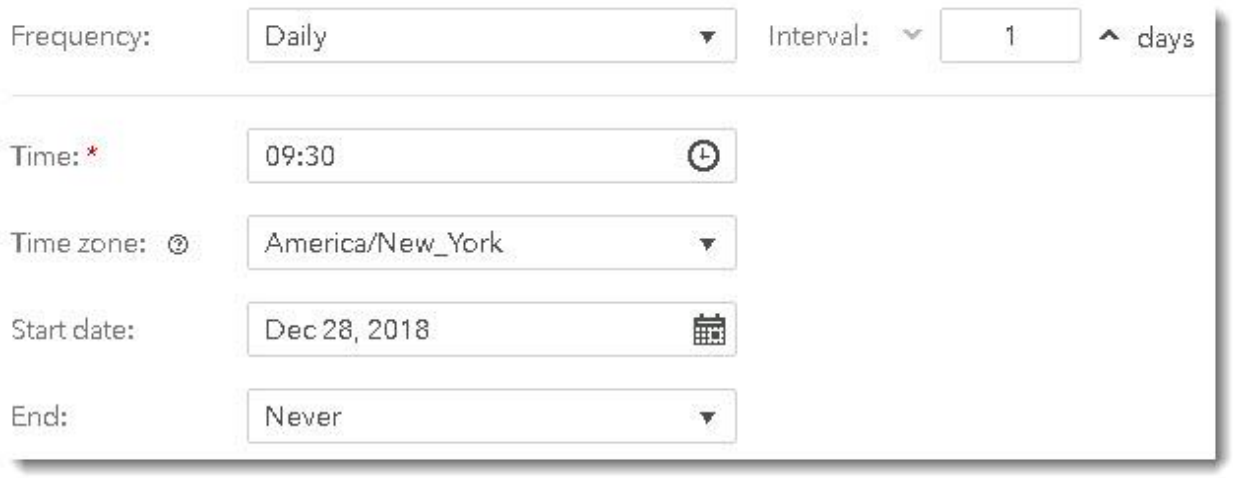

Figure 8. Capture of Daily Trigger in Visual Analytics GUI

Want to know how to define the above trigger for scheduling service in JSON format? Below is the sample daily trigger that starts at 09:30 on Dec 28, 2018.

```
[
        {
                "name": "Daily Trigger",
                "type":"TIMEEVENT",
                "hours":"09",
                "minutes" : "30",
                "timezone": "America/New_York",
                "maxOccurrence" : "-1",
                "recurrence" :
                {
                    "type" : "daily",
                    "startDate": "2018-12-28",
                    "skipCount": 1,
                    "daysOfWeek":[],
                    "dayOfMonth":1,
```

```
                        "dates":[]
                    }
                }
    ]
```

You can put the trigger content in a file named daily_trigger.txt as POST data

Once you create the trigger, the next step is scheduling your distribution request. Remember to provide the file name with @ prefix for the trigger file.

Method	URI	Task
POST	/reportDistribution/scheduler?distributionRequestId={distributionRequestId}	schedule a distribution request in the Scheduler service

```
curl -sk -X POST
"https://$server_name/reportDistribution/scheduler?distributionRequestI
d=$distribution_id" \
-H "accept: application/json" \
-H "content-type: application/vnd.sas.schedule.trigger+json" \
-H "Authorization: Bearer $access_token"
-d @daily_trigger.txt
```

CONCLUSION

By providing REST APIs and command-line interfaces (CLI), SAS Viya makes it possible to support automating your daily tasks about visual analytic with batch jobs and access the power of SAS.

REFERENCES

SAS Institute Inc. SAS® Viya® 3.4 Administration: Command-Line Interfaces. Available at,

https://go.documentation.sas.com/?docsetId=calcli&docsetTarget=n01xwtcatlinzrn1gztsglukb34a.htm&docsetVersion=3.4&locale=en

SAS Institute Inc. "Getting Started with SAS® Viya® REST APIs". Available: https://developer.sas.com/guides/rest.html

SAS Institute Inc. SAS Viya REST APIs. Available at,

https://developer.sas.com/apis/rest

ACKNOWLEDGMENTS

I would like to express my gratitude to Adam Bullock, Jade Walker, Kansun Xia and Wei He for the valuable feedbacks and the support to make this paper possible.

CONTACT INFORMATION

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

Rain Kang
SAS Research and Development (Beijing) Co., Ltd.
Rain.Kang@sas.com