Paper 5048-2020

Automatically Loading CAS Tables from SAS® Data Integration Studio Using SAS® Viya® REST APIs.

Oscar Simionati and Ervin Wilneder, Buenos Aires City Government.

ABSTRACT

Your SAS® Viya installation may have many SAS® Visual Analytics reports in several folders that also need many source data tables. Assuring that the information for each of those reports is loaded at the moment the users want to use them is not an easy task. We developed a comprehensive SAS® Data Integration Studio process that automatically finds the data source needed for each SAS® Visual Analytics report, extracts the tables from the database management system (DBMS), loads them as SAS® Cloud Analytic Services (CAS) tables, and checks that everything is correct for the reports' execution. This is possible thanks to the use of the SAS® Viya® REST API, integrating http requests that refer to the folders on which we want to automate the reports and the relationships between these and their CAS tables, along with the corresponding authorization to access the data.

With this integration of SAS® Viya with SAS® Data Integration Studio, we achieve a result in which both the developers and end users of the reports don't need to update the information or submit a request to their respective technical support, making the workflow of data analytics faster and more continuous.

INTRODUCTION

Faced with the need to give analytical information to different areas of an organization, SAS® Viya provides an architecture that allows to distribute both reports and data in a controlled and continuous way.

New reports are created frequently, and most of the existing ones are constantly evolving as new information is incorporated from frequently used data sources or even new ones. This constant evolution requires the tables to be updated automatically, to ensure that all reports have accurate and timely data.

This challenge can be taken using SAS® Data Integration Studio, and SAS® Viya metadata. Invoking SAS® Viya REST API, we have the possibility to update the CAS tables that are related to one or more reports automatically.

The focus of the solution proposed in this paper will be on updating CAS tables automatically, for a given set of reports within a particular folder (and the folders that are under that), through the use of SAS® Data Integration Studio and SAS® Viya REST API. This integration is fully achievable by any other team, company or organization, using the same tools and with a few lines of SAS® code.

ARCHITECTURE OF THE SOLUTION

The data flow in our organization is automated by several ETL processes developed with SAS® Data Integration Studio that take information from various sources, transactional systems and other databases, and transform it so it can be used in SAS® Viya, mainly using SAS® Visual Analytics. This information is stored and updated daily in our data warehouse. Then, a connection to it from SAS® Viya is established and, in this way, the reports can take this information.

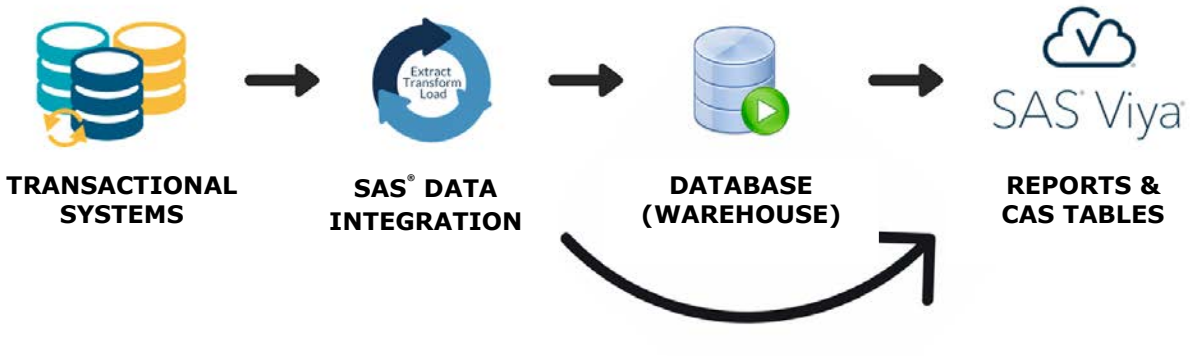


Figure 1. Basic representation of the architecture where the solution works.

In this paper, we will see how we can use SAS® Data Integration Studio to automatically and selectively update CAS Tables, with only a few lines of code in the ETL processes. We will use SAS® Viya metadata by invoking SAS® Viya REST API.

First, we will identify the SAS® Visual Analytics reports that must be updated automatically. In our case we do this by putting all production SAS® Visual Analytics reports below one specific branch of the SAS® Viya tree of folders.

Next, it is necessary to establish the relationship between each report and the CAS tables that feed those reports. This metadata is easily obtained with the use of the SAS® Viya REST API, which through certain queries allows us to know the list of reports and the list of its relationships with the tables.

Every day a large number of ETL processes run updating all the data warehouse tables. Some of them are the source to load the information in SAS® Viya CAS tables. Knowing which data warehouse tables must be loaded in SAS® Viya CAS Tables, allows the ETL process to launch the load process in SAS® Viya automatically after each table is updated.

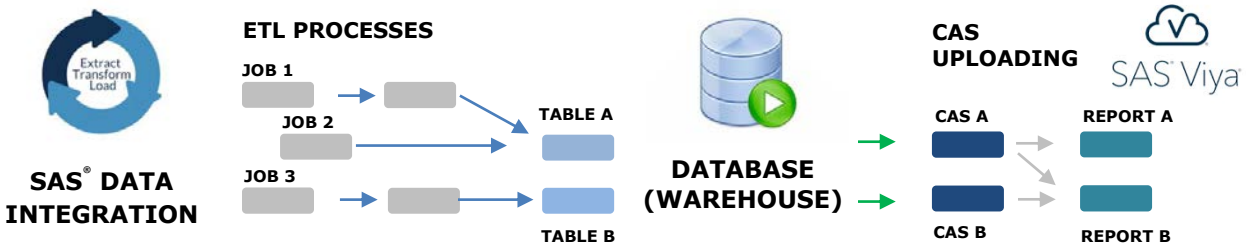


Figure 2. Representation of the linked process between jobs, table, CAS tables and reports.

Based on Figure 2, the process of loading CAS A and CAS B into the SAS® Viya environment will start only when TABLE A and TABLE B, respectively, have been updated. This means that we need to know first if the related ETL process with one or more of those tables have ended (in our diagram, JOB 1, JOB 2 and JOB 3). To achieve this, we used the internal table of objects and relationships from our data warehouse that allow us to know the dependencies between, for example, all views and the tables that they were built with. In addition to that, some specific metadata of ETL jobs is needed too, because we have to know when the information of every table or views (associated with each CAS tables) is effectively updated/finished. This point will be particular to each business or organization and the technologies that they are based on, so we don't go deeper in this part of the solution but will focus on the access and treatment of the information obtained from SAS® Viya. We mention this because it is important to take care about chaining every part of the process in such a way the information will be really updated.

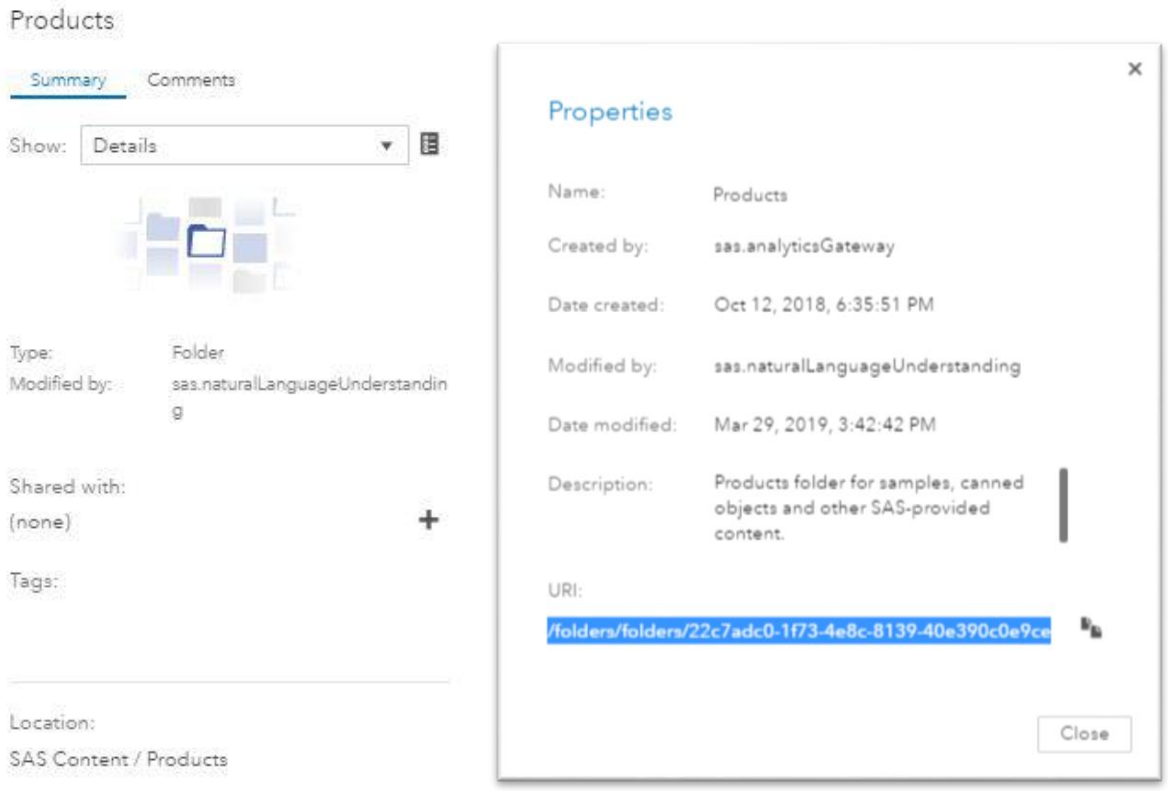
So in the next section, we will see how to obtain the metadata corresponding to reports and the relationships between those and the CAS tables, and how to load CAS tables in SAS® Viya invoking the REST API from SAS® Data Integration Studio.

DEVELOPMENT PROCESSS

OBTAINING REPORTS, CAS TABLES AND RELATIONSHIPS BETWEEN THEM FROM SAS® VIYA METADATA

The purpose of this part of the process is to know the relationship between each report, located in a particular folder (and subfolders of this one), and the CAS tables that provide the information for its graphics and other objects.

To do this, two queries (or formally known as requests) must be made, taking into account that the *uri* (the unique identifier) of the folder on which we will automate the reports is already known. The folder uri can be consulted in the properties of the element within SAS® Drive as you can see in Figure 1 (using screenshots of the Products folder as example). When you select a folder, by clicking in it, the right panel shows you by default the summary tab. This section has a drop down menu with two options: Details and Related Items. Keeping the first one, you must click on the side icon to find the corresponding uri.



Display 1. Screenshot of Products folder's sidepanel in SAS® Drive and its Details' information.

At the moment to make the request to the SAS® Viya REST API, other parameters to consider are the protocol, host, limit of items to be returned by the query (we use 50,000 since by default it returns 10 items) and of course the corresponding authorization to perform this task. The *authorization token* can be obtained using the corresponding user credentials (we use an admin user) and the following SAS® code (essentially a HTTP procedure) that returns the token stored in a global variable (*authtoken*) so you can use it later in the next requests.

```
  filename resp TEMP;
  %put Get authorization token...;
  %global authtoken;

  %let username = /*YOUR_USER_ID or &SYSUSERID*/
  %let password = /*YOUR_PASSWORD*/
  %let host = /*YOUR_HOSTNAME or &SYSHOSTNAME*/

  proc http
    method = "POST"
    url = "&host./SASLogon/oauth/token"
    in =
"grant_type=password%nrstr(&)username=&username.%nrstr(&)password=&password.
"
     out = resp;
     headers
        "Authorization"="Basic c2FzLmVjOg=="
```

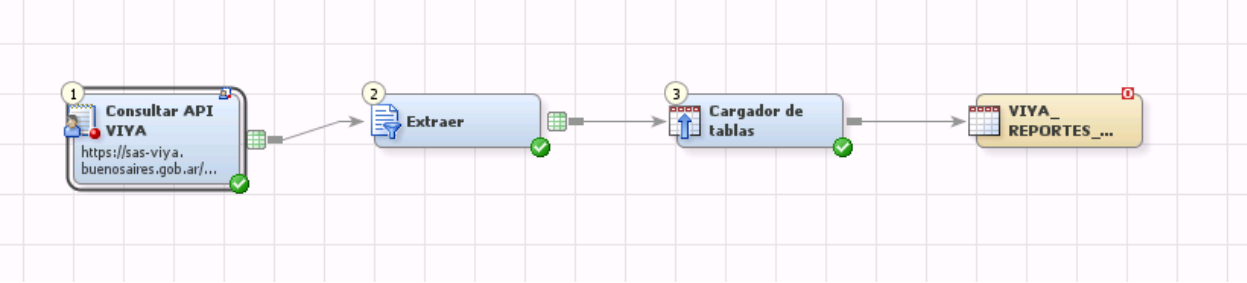
```
      "Accept"="application/json"
      "Content-Type"="application/x-www-form-urlencoded";
run;
%put Response status: &SYS_PROCHTTP_STATUS_CODE;

libname respjson JSON fileref=resp;
data _null_;
   set respjson.root;
   call symput('authtoken', access_token);
run;

%put authtoken = &authtoken;
```

Finally, with both the folder uri and token, you can do the request. This is made by writing custom code within a job node in SAS® Data Integration Studio and the information extraction ends in a table inside the data warehouse, as you can see in Figure 2. So the whole job consist in three steps or nodes:

1. Making the request.
2. Extracting the response information as a table.
3. Uploading the table in the data warehouse.



Display 2. Screenshot of the job that extracts information from the SAS® Viya REST API request.

At the first node, in the code section (inside node properties), we write the following SAS® code that performs the http request and then saves the information in a library. This library is then mapped and the part of the JSON that contains the item's details is placed in a table as we previously mentioned.

```
filename resp TEMP;
filename jmap TEMP;

%let bearerToken = "Bearer &authtoken."; /*the token obtained in previous
step*/
%let host = /*YOUR_HOSTNAME or &SYSHOSTNAME*/;
%let uri = /*YOUR_FOLDER_URI*/;
%let url = "&host.&uri./members?limit=50000&recursive=true"

proc http
  method = "GET"
  url = "&url."
```

```
  ct = "application/json"
  out = resp;
  headers
      "Authorization" = "&bearerToken."
      "Connection" = "keep-alive";
run;

libname items json fileref=resp map=jmap automap=create;
libname items json fileref=resp map=jmap;

proc datasets lib = items;
quit;
data &_output1.;
   set items.ITEMS;
run;
```

Please note that at the end of the code we make use of the variable *_output1* that exists in the environment of this specific node of SAS® Data Integration job. Other solutions could use different ways to store the information through the DATA step. Also, it is important to mention that the response information have between its elements not only reports but every subfolder or another type of elements (files, dataplans, etc.) contained by the one principal folder you select, so you need to filter the data to keep just the reports.

Ver datos: VIYA_REPORTES

Ir a la fila Ir

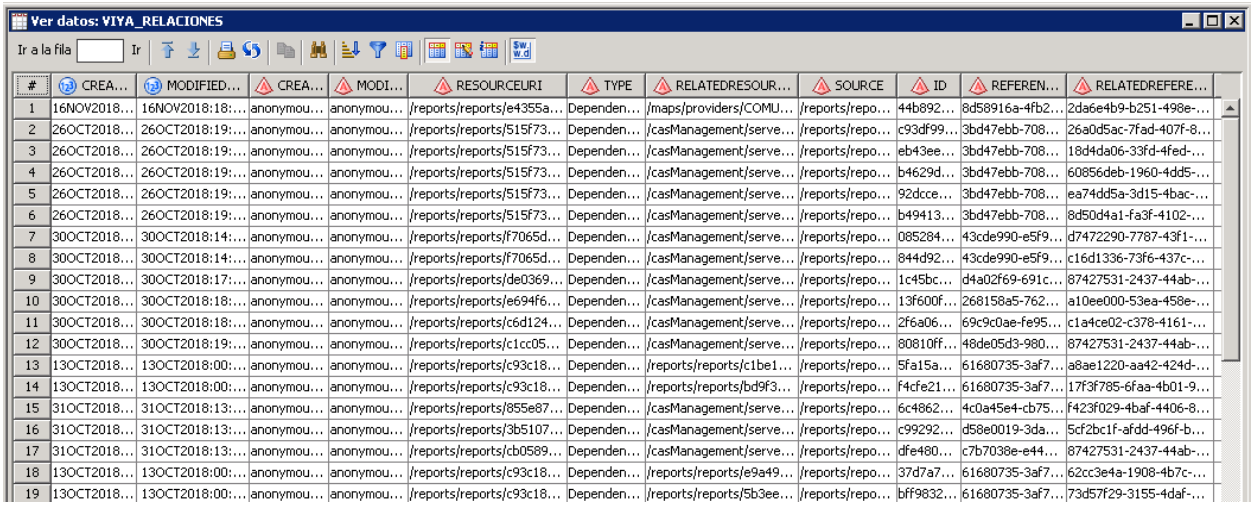
#	ID	NAME	CREATEDBY	FECHA_CREACION	MODIFIEDBY	FECHA_MODIFICADO	TYPE	VERSION	DESCRIPTION
1	d67dd76e-d15d-44a6-8...	Citas	20952878558	05NOV2018:19:29:02	20952878558	13DEC2018:15:50:02	report	1	
2	c1502bee-9a4d-48f6-a...	Citas (En Desa...	20952878558	02NOV2018:16:04:30	20952878558	11DEC2018:20:58:14	report	1	
3	ebdb4a9c-6831-4b3b-b...	Citasbackup	20952878558	06NOV2018:15:22:39	20952878558	06NOV2018:15:22:39	report	1	
4	e6b08c78-2e5e-4c03-9...	Cobertura - Co...	20952878558	20NOV2018:12:08:26	20952878558	19DEC2018:14:49:37	report	1	
5	48a0c8f2-adb0-4724-b...	Cobertura - Co...	20952878558	19DEC2018:13:50:00	20952878558	19DEC2018:14:48:02	report	1	
6	9b28c678-ca08-4fed-8...	Combos y Ban...	20952878558	23OCT2018:17:13:26	20952878558	19DEC2018:13:06:06	report	1	
7	84c097b7-57de-450e-b...	Consulta Certif...	20952878558	12NOV2018:19:12:46	20952878558	12NOV2018:19:12:47	report	1	
8	82882a85-4827-488e-b...	Consulta Certif...	20208911504	27SEP2018:17:11:25	20208911504	16AUG2019:20:14:20	report	1	
9	d7422476-dbfd-4e34-a...	Consulta ID de...	20336912319	17DEC2019:15:31:37	20336912319	17DEC2019:15:56:03	report	1	
10	a62723ef-c09d-4671-b...	Consulta QA	27226503876	05APR2019:14:01:24	27226503876	29APR2019:15:01:51	report	1	
11	54b50f52-aa88-484b-9...	Consulta de Fir...	20952878558	12NOV2018:19:20:46	20952878558	12NOV2018:19:20:46	report	1	
12	855e87e9-75d7-41dd-b...	Consulta de Fir...	20208911504	30OCT2018:13:00:30	20208911504	31OCT2018:13:01:27	report	1	
13	136bff32-10f7-4c1e-87...	Consulta ticket...	27226503876	05APR2019:13:56:26	27226503876	10OCT2019:12:47:13	report	1	
14	ec273658-2af5-45d2-a...	Consultas RIB ...	27280621981	16DEC2019:15:12:03	20261945046	19DEC2019:19:09:55	report	1	
15	b45998be-0131-4945-8...	Consultas RIB-...	27280621981	10DEC2019:15:22:30	27280621981	12DEC2019:20:18:40	report	1	
16	ed7b4bed-959f-483f-9f...	Consultas RIB-...	27280621981	17OCT2019:19:09:30	27280621981	09DEC2019:19:40:07	report	1	
17	ffae5b5a-db37-4388-a...	Consultas RIB-...	27280621981	01OCT2019:19:43:50	27280621981	21OCT2019:14:18:59	report	1	
18	9644ebdd-7a76-497a-9...	Consultas RIBs...	27280621981	08OCT2019:17:55:41	27280621981	08OCT2019:17:55:41	report	1	
19	6c75d87a-f46b-4230-8...	ContenidoSAS ...	27147633020	29NOV2019:18:31:15	27147633020	29NOV2019:19:59:34	report	1	
20	16625e15-5a56-4b9a-8...	Control SSL	20316530800	04OCT2019:14:31:11	20316530800	06FEB2020:15:16:03	report	1	
21	d18fc0ac-38ef-4725-9e...	Control de Red...	27280621981	01FEB2019:19:35:02	27280621981	01FEB2019:20:11:34	report	1	
22	473033e4-6c7f-41f2-8...	Control de Red...	20952878558	26AUG2018:22:20:56	sasboot	26OCT2018:19:41:48	report	1	
23	225b88a8-ec05-4bc5-9...	Control de Red...	20952878558	12NOV2018:20:00:07	20952878558	12NOV2018:20:00:10	report	1	

Display 3. Screenshot of the resulting table from members' folder request.

The same code and logic explain above can be reused to obtain the relationships. In that case, you just need to change the declaration of the *url* variable as follows (just replace this one in the previous code).

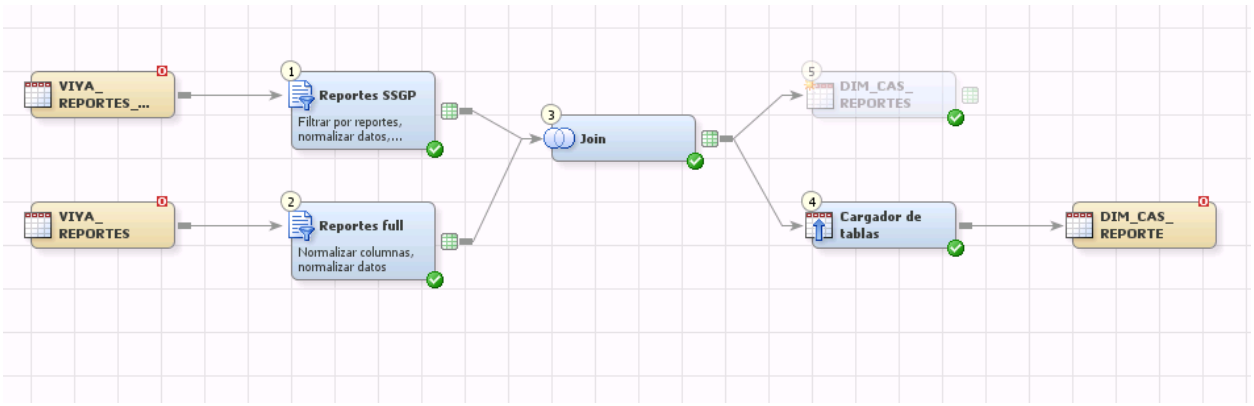
```
  %let url =
"&host./relationships/relationships?limit=20000&filter=eq('type','Dependent')
  "
```

The url doesn't make use of the folder uri because it is not necessary, the response obtained by the request has all the relationships between reports and CAS tables (the filter applied indicates that only this type of relationship is returned). Then both tables (tables and relationships) can be joined to generate a third table where the link between the reports and the CAS tables is finally obtained. In the following scheme it can be seen that the query to the previous URL returns a list of relationships where we can to take *resourceUri* as a foreign key for the union with the table of reports that we previously obtained (using their IDs) and know the associated CAS tables through the *relatedResourceUri* field.



#	CREA...	MODIFIED...	CREA...	MODI...	RESOURCEURI	TYPE	RELATEDRESOUR...	SOURCE	ID	REFEREN...	RELATEDREFERE...
1	16NOV2018...	16NOV2018:18:...	anonymou...	anonymou...	/reports/reports/e4355a...	Dependen...	/maps/providers/COMU...	/reports/repo...	44b892...	8d58916a-4fb2...	2da6e4b9-b251-498e-...
2	26OCT2018...	26OCT2018:19:...	anonymou...	anonymou...	/reports/reports/515f73...	Dependen...	/casManagement/serve...	/reports/repo...	c93df99...	3bd47ebb-708...	26a0d5ac-7fad-407f-8...
3	26OCT2018...	26OCT2018:19:...	anonymou...	anonymou...	/reports/reports/515f73...	Dependen...	/casManagement/serve...	/reports/repo...	eb43ee...	3bd47ebb-708...	18d4da06-33fd-4fed-...
4	26OCT2018...	26OCT2018:19:...	anonymou...	anonymou...	/reports/reports/515f73...	Dependen...	/casManagement/serve...	/reports/repo...	b4629d...	3bd47ebb-708...	60856deb-1960-4dd5-...
5	26OCT2018...	26OCT2018:19:...	anonymou...	anonymou...	/reports/reports/515f73...	Dependen...	/casManagement/serve...	/reports/repo...	92dcce...	3bd47ebb-708...	ea74dd5a-3d15-4bac-...
6	26OCT2018...	26OCT2018:19:...	anonymou...	anonymou...	/reports/reports/515f73...	Dependen...	/casManagement/serve...	/reports/repo...	b49413...	3bd47ebb-708...	8d50d4a1-fa3f-4102-...
7	30OCT2018...	30OCT2018:14:...	anonymou...	anonymou...	/reports/reports/f7065d...	Dependen...	/casManagement/serve...	/reports/repo...	085284...	43cde990-e5f9...	d7472290-7787-43f1-...
8	30OCT2018...	30OCT2018:14:...	anonymou...	anonymou...	/reports/reports/f7065d...	Dependen...	/casManagement/serve...	/reports/repo...	844d92...	43cde990-e5f9...	c16d1336-73f6-437c-...
9	30OCT2018...	30OCT2018:17:...	anonymou...	anonymou...	/reports/reports/de0369...	Dependen...	/casManagement/serve...	/reports/repo...	1c45bc...	d4a02f69-691c...	87427531-2437-44ab-...
10	30OCT2018...	30OCT2018:18:...	anonymou...	anonymou...	/reports/reports/e694f6...	Dependen...	/casManagement/serve...	/reports/repo...	13f600f...	268158a5-762...	a10ee000-53ea-458e-...
11	30OCT2018...	30OCT2018:18:...	anonymou...	anonymou...	/reports/reports/c6d124...	Dependen...	/casManagement/serve...	/reports/repo...	2f6a06...	69c9c0ae-fe95...	c1a4ce02-c378-4161-...
12	30OCT2018...	30OCT2018:19:...	anonymou...	anonymou...	/reports/reports/c1cc05...	Dependen...	/casManagement/serve...	/reports/repo...	80810ff...	48de05d3-980...	87427531-2437-44ab-...
13	13OCT2018...	13OCT2018:00:...	anonymou...	anonymou...	/reports/reports/c93c18...	Dependen...	/reports/reports/c1be1...	/reports/repo...	5fa15a...	61680735-3af7...	a8ae1220-aa42-424d-...
14	13OCT2018...	13OCT2018:00:...	anonymou...	anonymou...	/reports/reports/c93c18...	Dependen...	/reports/reports/bd9f3...	/reports/repo...	f4cfe21...	61680735-3af7...	17f3f785-6faa-4b01-9...
15	31OCT2018...	31OCT2018:13:...	anonymou...	anonymou...	/reports/reports/855e87...	Dependen...	/casManagement/serve...	/reports/repo...	6c4862...	4c0a45e4-cb75...	f423f029-4baf-4406-8...
16	31OCT2018...	31OCT2018:13:...	anonymou...	anonymou...	/reports/reports/3b5107...	Dependen...	/casManagement/serve...	/reports/repo...	c99292...	d58e0019-3da...	5cf2bc1f-afdd-496f-b...
17	31OCT2018...	31OCT2018:13:...	anonymou...	anonymou...	/reports/reports/cb0589...	Dependen...	/casManagement/serve...	/reports/repo...	dfe480...	c7b7038e-e44...	87427531-2437-44ab-...
18	13OCT2018...	13OCT2018:00:...	anonymou...	anonymou...	/reports/reports/c93c18...	Dependen...	/reports/reports/e9a49...	/reports/repo...	37d7a7...	61680735-3af7...	62cc3e4a-1908-4b7c-...
19	13OCT2018...	13OCT2018:00:...	anonymou...	anonymou...	/reports/reports/c93c18...	Dependen...	/reports/reports/5b3ee...	/reports/repo...	bff9832...	61680735-3af7...	73d57f29-3155-4daf-...

Display 4. Screenshot of the resulting table from relationships request.



Display 5. Screenshot of the job that joins both relationships and reports tables.

Here it is important to mention that the CAS table/tables associated with each report are in the *relatedResourceUri* field which you need to split up by the slash character in order to extract the information about name of the table itself and the caslib where they are located.

Finally, with this information we can go to the next step where we will see how to load each CAS table once the part of the ETL process that populates that table finishes, making sure that the data that users see in the reports is up to date.

It should be noted that the development of the solution could end here, because depending on the infrastructure of the organization, CAS tables may be loaded without checking if the

source data is updated or not. The metadata collected so far is enough to update the CAS tables.

PREPARING METADATA FROM ETL JOBS AND LINKING THIS ONE WITH EACH CAS TABLE

As it was explained before, our solution needs that every CAS table that we will upload to SAS® Viya has its data properly updated. To achieve this, you need to prepare, through another ETL process, a table similar to the next one where it can be seen the final relationship between, mainly, each CAS table, tables which they depend on and jobs that load those tables. For more information about the SAS® code that can return this information, please see the References section at the end of this paper.

Ver datos: ETL_CONTROL_CAS

Ir a la fila Ir

#	CAS_NAME	DEP_NAME	DEP_OWNER	OWNER	TIPO	JOB_FLOW	LVL1	LVL2	LVL3	LVL4	LVL5	FECHA_CARGA_TABLA	ID_CARGA
1	LSR_PSOC_REC...	STG_PSOC_LOT...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:12:22	1201094
2	LSR_PSOC_REC...	STG_PSOC_PAG...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:11:26	1201094
3	LSR_PSOC_REC...	STG_PSOC_PLA...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:10:31	1201094
4	LSR_PSOC_REC...	STG_RU_DOMIC...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:11	1201094
5	LSR_PSOC_REC...	STG_RU_IDENTI...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:56	1201094
6	LSR_PSOC_REC...	STG_RU_REGIS...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:46	1201094
7	LSR_PSOC_REC...	STG_SIS_BARRI...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:01:46	1201094
8	LSR_PSOC_REC...	STG_SIS_COMU...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:01:36	1201094
9	LSR_PSOC_REC...	STG_PSOC_BEN...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:37	1201094
10	LSR_PSOC_REC...	STG_PSOC_BEN...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:15:32	1201094
11	LSR_PSOC_REC...	STG_PSOC_CUE...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:07	1201094
12	LSR_PSOC_REC...	STG_PSOC_DET...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:17	1201094
13	LSR_PSOC_REC...	STG_PSOC_DET...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:17:12	1201094
14	LSR_PSOC_REC...	STG_PSOC_EXP...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:11:47	1201094
15	LSR_PSOC_REC...	STG_PSOC_LOT...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:12:22	1201094
16	LSR_PSOC_REC...	STG_PSOC_PAG...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:11:26	1201094
17	LSR_PSOC_REC...	STG_PSOC_PLA...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:10:31	1201094
18	LSR_PSOC_REC...	STG_RU_DOMIC...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:11	1201094
19	LSR_PSOC_REC...	STG_RU_IDENTI...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:56	1201094
20	LSR_PSOC_REC...	STG_RU_REGIS...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:04:46	1201094
21	LSR_PSOC_REC...	STG_SIS_BARRI...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:01:46	1201094
22	LSR_PSOC_REC...	STG_SIS_COMU...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:01:36	1201094
23	LSR_PSOC_REC...	STG_PSOC_BEN...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:37	1201094
24	LSR_PSOC_REC...	STG_PSOC_BEN...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:15:32	1201094
25	LSR_PSOC_REC...	STG_PSOC_CUE...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:07	1201094
26	LSR_PSOC_REC...	STG_PSOC_DET...	SAS_STG ...	DMPRD ...	VIEW ...	JOB_PMDS_000...	CADENA_...	SSGPMBI_...	JOB_PMD...	...	...	27FEB2020:02:13:17	1201094

Display 6. Screenshot of the table that relate CAS tables with its corresponding ETL jobs.

The fields that are really necessary in this case are:

- Name of the CAS – CAS_NAME (or other property that works as an identifier)
- Tables associated with CAS – DEP_NAME (obtained from internal table of objects and relationships from our data warehouse because each CAS could be equivalent to a view and so depend on more than one table).
- Jobs associated to tables – JOB_FLOW(mentioned on previous paragraph)

So if you then join this table the information of whether a job has ended or not, you can obtain a flag for each CAS indicating that it is certainly ready to upload to SAS® Viya. Taking this flag as trigger, in the next section we will see how to update all the CAS tables using a final ETL process in SAS® Data Integration Studio, though this also can do with, for example, SAS® Studio.

LOADING INFORMATION AS SAS® VIYA CAS TABLES FROM SAS® DATA INTEGRATION

With the information of tables and relationships, and having executed all the corresponding ETL processes, it is easy to iterate, taking the field that is identified with the name of the CAS table to load (CAS_NAME in our example) and performing a CASUTIL procedure, which has the caslib "IN" where the table is located (connection to the database) and the caslib "OUT" where we want to upload the information, both as parameters.

```
proc casutil incaslib = "IN" outcaslib = "OUT" sesref = &SessionName.;
   load casdata = "&CAS_NAME."
   casout = "&CAS_NAME. "
   importoptions = (filetype = "BASESAS", datatransfermode = "parallel")
   promote;
quit;
```

CONCLUSION

The techniques explained in this document provide several utilities: the first is to ensure that SAS® Data Integration Studio ETL processes end their execution by loading CAS tables automatically, guaranteeing that the reports are updated. The second is the possibility to automatically identify and load the CAS tables that should be updated, consulting the SAS® Viya metadata and invoking functions of the SAS® Viya REST API. Finally and most importantly, it demonstrates the ability to integrate SAS® Viya with SAS® 9.4.

REFERENCES

SAS® Communities. PROC Metadata to get Dependent Objects (https://communities.sas.com/t5/SAS-Data-Management/Proc-Metadata-to-get-Dependent-Objects/td-p/275297)

RECOMMENDED READING

- *Getting started with REST APIs for SAS® Viya*

CONTACT INFORMATION <HEADING 1>

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

Oscar Simionati
oscar.simionati@buenosaires.gob.ar