PharmaSUG 2022 - Paper AD-155

Connecting SAS® and Smartsheet® to Track Clinical Deliverables

Siddharth Kumar, Greg Weber, Navitas Data Sciences
Steve Hege, Alexion Pharmaceuticals

ABSTRACT

Efficient monitoring and management of the clinical programming development and validation lifecycle is vital and can be a challenge in clinical study reporting. There are various methods used by companies to track the progress of clinical deliverables. Our current solution is tracking progress using Microsoft Excel spreadsheets. In our Statistical Computing Environment (SCE), this is not ideal as it requires checking out, downloading, editing the spreadsheet, and then uploading and checking in the updated Excel file. This manual process is both time-consuming and prone to error. In addition, sharing and working collaboratively is problematic as only one user can update the file at a time, so a better solution is desired.

In this paper, we discuss connecting Smartsheet® with our SAS® LSAF® environment to provide a more collaborative Clinical Programming Deliverable Tracker requiring less intervention from managers and programmers. Smartsheet® is an online service for work management and collaboration that uses a tabular interface and provides workflow capabilities. We demonstrate techniques that utilize HTTP and REST to interact with and update our Smartsheet® Tracker from SAS®, using Proc HTTP along with the LSAF® macro and Smartsheet® APIs. Smartsheet® has become an important part of our SCE ecosystem and, using the processes and techniques developed for our Clinical Tracker, we plan to automate other SCE processes.

INTRODUCTION

When we introduced the Life Science Analytics Framework (LSAF), our department needed a way to manage user support requests. These included creating accounts, handling project access and creation requests, and responding to issues. It was suggested that we use Smartsheet as the organization licensed it. We were quickly and easily able to use the Smartsheet Dashboard capability to provide what we call the “LSAF Help Portal.” Since that time, we have been expanding our knowledge and using Smartsheet to manage and automate other processes. We concentrate here on just one of our automation efforts, which is to improve the tracking of clinical deliverables. The Clinical Programming Deliverable Tracker, referred to as “tracker” from now on, required using SAS to create and share Smartsheet objects, read sheets, update rows and columns in sheets, and so provided examples of all the techniques we used to connect SAS and Smartsheet. It should be noted that, while we use the SAS LSAF environment in our discussion, the techniques and methods demonstrated do not rely on LSAF and can be applied from any SAS environment.

In this paper, we discuss connecting Smartsheet® with our SAS® LSAF® environment to provide a more collaborative Clinical Programming Deliverable Tracker requiring less intervention from managers and programmers. Smartsheet® is an online service for work management and collaboration that uses a tabular interface and provides workflow capabilities. We demonstrate techniques that utilize HTTP and REST to interact with and update our Smartsheet® Tracker from SAS®, using Proc HTTP along with the LSAF® macro and Smartsheet® APIs. Smartsheet® has become an important part of our SCE ecosystem and, using the processes and techniques developed for our Clinical Tracker, we plan to automate other SCE processes.

PROGRAMMING TECHNOLOGIES

Besides SAS, these automations and processes rely on two technologies to move and update data between LSAF and Smartsheets. These technologies are the Smartsheet REST API 2.0 (Smartsheet, 2022) to access the web service and LSAF SAS Macro API 2.4 for accessing functions within LSAF.

REST AND SMARTSHEET API

An application programming interface (API) is a set of definitions and protocols used for building and integrating application software. It acts as a communicator between a program and a system so it can understand and respond as needed. REST (REpresentational State Transfer) is defined as a set of architectural constraints that can be implemented in various ways. A REST (sometimes RESTful API) is a combination of these concepts that allows interaction with web services from applications.

The Smartsheet API gives the ability to programmatically access, update and manage our Smartsheet data and accounts. With the Smartsheet API, we can build programs and processes that automatically read and update sheets; build folders and workspaces; and manage user account access to sheets and dashboards.

While many different methods are available for using the Smartsheet API, we found delivering the information using four basic operations (GET, POST, PUT, and DELETE) from a PROC HTTP call as listed in Table 1 below.

Method	Description
GET	Gets the sheet specified in the URL. Returns the sheet, including rows, and optionally populated with discussion and attachment objects in XML formatted file.
POST	Creates a sheet from scratch in the user's Sheets folder
PUT	Updates the sheet specified in the URL.
DELETE	Deletes the sheet specified in the URL

Table 1. HTTP Methods

For cell or sheet updates, a SAS program builds a JSON format, input is built with the revising information, then transmitted to Smartsheets with a POST method. JSON is a lightweight, open file format used for storing and transporting data between web applications and servers (JSON.org, 2022). It uses human-readable text to store objects as attribute-value pairs, arrays, or serialized data values. An example of an attribute-value pair is {“name”:”John Doe”} (note the curly brackets and double quotes).

SAS AND LSAF MACRO API

The SAS Life Science Analytics Framework SAS Macro API allows us to access and act on objects in the LSAF repository and workspace. It follows the familiar SAS macro call syntax and the macros are automatically part of the SASAUTOS search path. While many API macros are available, in this paper, we are only discussing the ones relevant to our tracker shown in Table 2 below.

Macro Name	Information Retrieved
%LSAF_GETCHILDREN	All items that are within a repository folder and subfolders.
%LSAF_GETALLUSERS	All user accounts that are defined on the system.
%LSAF_GETGROUPS	All groups that are defined within the specified context.

Table 2. LASF API Macros Used in Our Project

Of course, LSAF is not a requirement to connect to Smartsheets, it’s just that this is what applies in our situation and environment.

CONNECTING SAS AND SMARTSHEET

GETTING STARTED WITH SMARTSHEET API

We did not have any experience with SAS Proc HTTP and the Smartsheet API documentation did not, of course, include any SAS code examples. There were examples for C#, Java Nodes.js, Python, Ruby, and

cURL. We found that we could quite easily take the cURL examples and translate them to the needed SAS proc http call. cURL is a command line tool that gives the capability to interact with websites.

Translating Smartsheet API cURL Example to SAS

For a simple example, we can look at the Smartsheet API documentation cURL code to create a new Smartsheet workspace. A Smartsheet workspace is similar in concept to an Excel workbook. The color-coding below indicates where the various cURL values are placed when translating to using Proc HTTP.

cURL

```
curl https://api.smartsheet.com/2.0/workspaces \
-H "Authorization: Bearer access token" \
-H "Content-Type: application/json" \
-X POST \
-d '{"name": "New workspace"}'
```

SAS, JSON and Proc HTTP

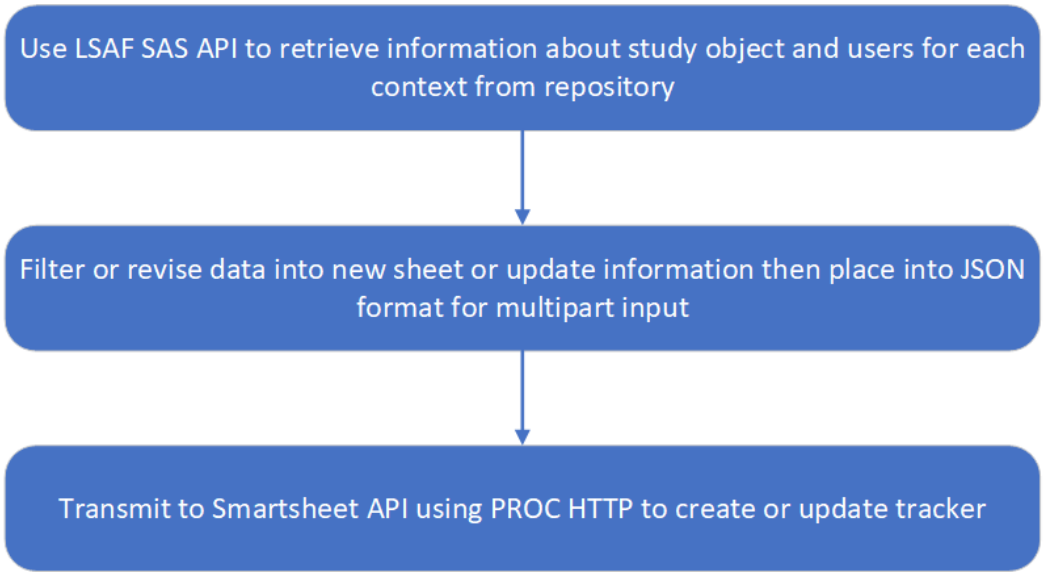
```
/* Build our JSON data */
filename wspace temp;

data _null_;
   file wspace;
     put "{";
     put '"name":"' "New workspace" '"';
     put "}";
run;

/* Send the JSON data to Smartsheet */
filename resp "c:\temp\resp.txt";
proc http
  proxyhost="proxy.com" /* LSAF requires a proxy */
  proxyport=3128
  url="https://api.smartsheet.com/2.0/workspaces"
  method="POST"
  in=wspace
  out=resp;
  headers
    "Accept"="application/json"
    "Content-Type"="application/json"
    "Authorization"="Bearer access token";
run;
```

PROCESS FLOW

The basic process flow for interacting between SAS LSAF and Smartsheet is shown below.



SMARTSHEET CLINICAL TRACKER

After some learning and experimentation, we were ready to put the Smartsheet API and Proc HTTP to use in developing our Smartsheet Clinical Tracker.

INITIAL TRACKER

For this discussion, we will use a tracker for SDTM deliverables. This Smartsheet grid has one row for each SDTM deliverable. The tracker has customary columns for tracking the progress of development and validation.

Domain	Specs Ready Y/N	Domain Label	Development Programmer	Development Status	Development Date	Validation Programmer	Validation Status
AE	Y	Adverse Events	S Hege	Ready for QC	10/20/21	G. Weber	QC Passed
CM	Y	Concomitant Medications	S Hege	Ready for QC	11/15/21	G. Weber	QC Passed
DM	Y	Demographics	S Hege	Ready for QC	10/19/21	G. Weber	QC Passed
DS	Y	Disposition	S Hege	Not-Started	10/19/21	G. Weber	Not-Started
EX	Y	Exposure	S Hege	In-Progress	10/19/21	G. Weber	In-Progress
LB	Y	Laboratory Test Results	S Hege	Ready for QC	01/27/22	G. Weber	Comment Sent
SE	Y	Subject Elements	S Kumar	Ready for QC	10/19/21	G. Weber	QC Passed
SV	Y	Subject Visits	S Kumar	Ready for QC	02/09/22	G. Weber	QC Passed
TA	Y	Trial Arms	S Kumar	Ready for QC	08/17/21	G. Weber	QC Passed
TE	Y	Trial Elements	S Kumar	Ready for QC	11/02/21	G. Weber	QC Passed

Figure 1. Initial Tracker Columns

Typically, these columns are updated by the programmers. To support, augment and validate these columns, we add information to the tracker which is automatically populated from information available in LSAF. The following columns are populated and updated periodically. The updated tracker is shown at the end of this section.

1. Last modification date of the programs (both development and validation programs).
2. The user who last modified the programs.
3. List of WARNINGs or ERRORs from the SAS logs, if any.
4. Results of comparison of the development and validation data.
5. An overall development status.

RETRIEVING THE DATA

The first step is to use the %lsaf_getchildren macro to gather the list of items. In this case, SDTM datasets. The data is shown in Figure 2 below.

```
%lsaf_getchildren(lsaf_path=C:\project\analysis\sdtm\output, lsaf_recursive=1);
```

name	itemType	lastModified	isVersioned	version
ae.sas	sas:sasprogram	Thu Feb 10 15:23:23 GM...	1	1.21
cm.sas	sas:sasprogram	Thu Jan 06 15:20:06 GM...	1	1.42
dm.sas	sas:sasprogram	Mon Dec 06 20:23:55 GM...	1	1.7
ds.sas	sas:sasprogram	Tue Feb 08 19:18:01 GM...	1	1.18
ec.sas	sas:sasprogram	Tue Nov 09 15:31:19 GM...	1	1.0
eg.sas	sas:sasprogram	Mon Nov 08 09:48:28 GM...	1	1.9
ex.sas	sas:sasprogram	Tue Oct 05 19:26:01 GM...	1	1.1
fa.sas	sas:sasprogram	Fri Feb 18 08:01:05 GMT ...	1	1.22
ho.sas	sas:sasprogram	Thu Jan 20 11:53:37 GM...	1	1.5
ie.sas	sas:sasprogram	Mon Nov 22 05:31:28 GM...	1	1.5

Figure 2. The Output of %lsaf_getchildren

Using the list of existing SDTM deliverables, we look up the data we want to populate into the additional columns. We manipulate this dataset to get it into the format required. Following is an example of the input data that would be used to create the JSON file to be passed with Proc HTTP. It contains the Smartsheet rowid, DomainID (the column ID), and the values as shown in Figure 3 below. These values

are extracted using the Smartsheet API and are required to insert the values into the corresponding cell in the Smartsheet grid.

rowid	DomainID	domain	logmessage	comparison_report_value
1295066...	7603780312164228	DM	No findings found from log check	No unequal values were found. All valu...
5728297...	7603780312164228	IE	No findings found from log check	No unequal values were found. All valu...
6854197...	7603780312164228	QS	No findings found from log check	No unequal values were found. All valu...
		PC	No findings found from log check	No unequal values were found. All valu...
7842277...	7603780312164228	HO	No findings found from log check	No unequal values were found. All valu...
		EC	No findings found from log check	No unequal values were found. All valu...
		IS	No findings found from log check	No unequal values were found. All valu...
7575070...	7603780312164228	PE	No findings found from log check	No unequal values were found. All valu...
3476497...	7603780312164228	MH	No findings found from log check	No unequal values were found. All valu...
5235715...	7603780312164228	EX	No findings found from log check	No unequal values were found. All valu...

Figure 3. Smartsheet Row and Column ID Information

BUILDING THE JSON PACKAGE

We take the input data and put it into the JSON format we devised from looking at the cURL example in the Smartsheet API documentation. We usually call this the JSON package, which will be sent to Smartsheet using Proc HTTP. The example section of code below updates the log message column.

```
data _null_;
  file json_in;
  set input_data;
  put "{" ;
  put '"id":' '"' rowid '"' ',' ;
  put '"cells":'
  put '{' '"columnId":' DomainId ',' "value": '"' logmessage '"' '}';
run;
```

SENDING THE JSON PACKAGE TO SMARTSHEET TO UPDATE THE TRACKER

The JSON file is processed through Proc HTTP to update Smartsheet using the Smartsheet API.

```
filename resp temp; /* to capture the return response */
filename head temp;  /* to capture the return header */

proc http
  proxyhost="proxy.com"    /* If your env. requires a proxy */
  proxyport=3128
   url=https://api.smartsheet.com/sheets/sheet_id/rows
   method="POST"           /* PUT, DELETE, GET              */
   in=json_in
   headerout=head
   out=resp;
   headers
        "Accept"="application/json"
        "Content-Type"="application/json"
        "Authorization"="Bearer Bearer Code";
   run;
```

UPDATED TRACKER

Following is an example of an updated tracker in Figure 4. The columns have been updated with the latest information.

Development Program (Last Modified By)	Development Program (Last Modified Date)	Development Log Check	Validation Program (Last Modified By)	Validation Program (Last Modified Date)	Validation Log Check	Validation Comparison Report	Validation Run after Development Date (Yes/No)	Overall Status
S Kumar	Thu Jan 13 17:11:16 G	No findings found from log check	G Weber	Wed Feb 02 18:20:35 GM	No findings found from log check	!Unequal values were found.	Yes	!Issue
S Kumar	Wed Feb 02 20:02:50 (	No findings found from log check	G Weber	Wed Feb 02 21:44:57 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Thu Jan 27 12:26:46 G	No findings found from log check	G Weber	Wed Feb 02 15:28:24 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Fri Jan 28 06:56:46 GM	No findings found from log check	G Weber	Thu Jan 13 20:39:00 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Fri Feb 04 21:04:11 GM	No findings found from log check	G Weber	Mon Feb 07 13:27:30 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Tue Jan 18 07:52:51 G	No findings found from log check	G Weber	Wed Feb 02 12:08:22 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Fri Feb 04 14:39:46 GM	No findings found from log check	G Weber	Mon Nov 29 05:51:31 GM	WARNING: The requested format could not be applied successfully because the Value Comparison	!Unequal values were found.	No	!Issue
S Kumar	Tue Dec 07 12:27:00 G	No findings found from log check	G Weber	Tue Dec 07 12:37:46 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Mon Feb 07 10:10:07 (	No findings found from log check	G Weber	Fri Jan 28 10:06:54 GMT	No findings found from log check	No unequal values were found. All values compared are exactly equal.	No	!Issue
S Kumar	Tue Feb 01 18:28:11 G	No findings found from log check	G Weber	Thu Feb 03 09:46:14 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Tue Feb 01 18:40:41 G	No findings found from log check	G Weber	Wed Feb 02 11:32:12 GM	No findings found from log check	No unequal values were found. All values compared are exactly equal.	Yes	OK
S Kumar	Tue Feb 01 21:01:13 G	No findings found from log check	G Weber	Fri Feb 04 08:56:16 GMT	chklog dataset !Not Available	!Not Available	No	!Issue

Figure 4. Tracker with Updated Information

The LSAF Job feature allows project leads to schedule the job run to update the tracker in regular intervals or the job can be run at any time. We continue to add functionality. The entire process of managing the tracking of clinical deliverables can be automated. This includes initiating a new workspace, creating and sharing the tracker with team members, updating the column drop-down lists, adding rows as items are found by DPP (Data Presentation Plan), and updating the LSAF folder-level metadata columns.

CONCLUSION

In this paper, we presented just one of the ways we have found to take advantage of connecting SAS with Smartsheet. Using a combination of the LSAF API, Smartsheet API and just plain old SAS we were able to provide Project Leads with up-to-date information regarding the status of their project deliverables. This is in addition to the improvement of just using Smartsheet over the existing cumbersome process of tracking using an Excel spreadsheet, which required a download, edit, and upload process. Also, we have

been able to automate time-consuming LSAF administrative tasks such as the creation of new user accounts, project access requests and password resets. RAVE and other data transfers are also benefitting from these techniques. The possibilities seem endless.

REFERENCES

SAS Blog Chris Hemedinger, 23Jan2018 **How to test PROC HTTP and the JSON library engine**

SAS Blog Chris Hemedinger, 16Jan2018 **How to secure your REST API credentials in SAS programs**

JSON.org, **Introducing JSON**, https://www.json.org/json-en.html, Retrieved 07Apr2022

Smartsheet API 2.0 Documentation, Smartsheet Inc., 2022-03-07, https://smartsheet.redoc.ly/, Retrieved 07Apr2022

RECOMMENDED READING

- The references listed above
- Smartsheet API documentation
- SAS® Life Science Analytics Framework: SAS Macro API 2.4 User's Guide

CONTACT INFORMATION

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

Steven Hege
Alexion Pharmaceuticals
Steven.Hege@alexion.com

Siddharth Kumar
Navitas Data Sciences
Siddharth.Kumar@navitaslifesciences.com

Greg Weber
Navitas Data Sciences
Greg.Weber@navitaslifesciences.com