PhUSE EU Connect 2018

Paper SM01

iGET It! Downloading Data from a Centralised Repository using PROC HTTP

Nev Cope, Early Clinical Development, IMED Biotech Unit, AstraZeneca, Cambridge, UK

ABSTRACT

AstraZeneca released two new platforms in 2017. The first was entimICE-AZ (by Entimo AG [Berlin, Germany]), a single clinical data repository analysis and reporting environment, replacing many legacy platforms for patient-level clinical trial data and aiming to be our "single source of truth". The second was a high-performance statistical analysis computing platform, The SAS® Grid.

Early clinical development (ECD) programming work is exploratory in nature to support decision making in early-phase (phase 1 to 2) clinical development, so interactive code development is key. SAS Grid enables interactivity, but company data is stored in entimICE-AZ, whose SAS environment is batch focused. The two systems are hosted in different data centres with no direct link, except for a Web Services Server attached to entimICE-AZ. This paper introduces web services and PROC HTTP and outlines the development of a set of SAS Macros called 'iGET' that authenticate, query hierarchies and temporarily download data from entimICE-AZ to the SAS Grid, allowing programmers in ECD to utilise the power of both environments.

INTRODUCTION

Moving to new computer systems can be a difficult and stressful time, especially when legacy systems are slowly being retired and the day-to-day work still needs to be done. AstraZeneca went live with two new computing platforms in 2017, a SAS® Grid and the entimo® "entimICE" product.

The entimICE-AZ system is a customised version of the EntimICE product offered by Entimo, and described on their website as [1]:

> *a metadata-driven and configurable product suite with seamlessly integrated, model-independent metadata repository (MDR) for standards governance, high-performing clinical data repository (CDR) for data management, multi-language statistical computing environment (SCE) for analysis as well as numerous productive tools along the entire clinical study lifecycle*

The AstraZeneca (AZ) implementation is designed to be the main storage system for all clinical trial data from all phases of clinical development. Snapshots of study data from multiple vendors and databases are loaded into the system, and analysis performed using built-in SAS controllers (R and Spotfire instances are also available). The SAS programs (including macros and other tools), log and output files are also stored within the system, in separate Development and Production areas, along with version control and a full audit trail.

The SAS Grid is a system where SAS computing tasks are distributed among multiple computers on a network, all under the control of a SAS Grid Manager. In this environment, workloads are distributed across a grid cluster of computers.

Both systems went live at AstraZeneca in 2017. The SAS Grid replaced virtual desktops with PC SAS installations, and entimICE-AZ replaced numerous different file shares and data repositories. Around the same time, the ECD group created a new internal team, labelled "Data Operations", whose role was to focus on bringing efficiencies to the wider group, through evaluation, development, design, and maintenance of tools, utilities and efficiencies in data collection and analysis.

This paper will discuss the journey the author took, as part of this new Data Operations team, to try and connect the systems and bring coding flexibility to his programming colleagues.

SEPARATE SYSTEMS

Common tasks that ECD programmers and statisticians perform are running statistical simulations, analysing clinical data for regulatory reporting (IB, DSUR, etc.), and performing exploratory analyses. This latter task has caused some problems to SAS users within the new systems. entimICE-AZ is capable of running pre-existing code and creating outputs in batch, but exploratory analyses are very much an interactive process. Code is written and executed, results are examined, code is tweaked and rerun, results are examined, and so on.

While interactive code development is possible within entimICE-AZ, the interface is a simple text editor and not an IDE. There is no code completion or any task builders. Any created output (logs, lst, rtf, data sets, etc.) are saved to other areas of the system that can take additional clicks to retrieve and review, further slowing down the user experience. This type of work could be greatly improved if it could be performed on the SAS Grid instead, using SAS Studio or SAS Enterprise Guide, but entimICE-AZ and the SAS Grid are physically located in different data centres with no hard connection between the two.

If only there was a way to get data from entimICE-AZ into SAS Grid…

RESTFUL WEB SERVICES

Shortly after the two systems entered production, I received some R code from a colleague that interfaced with entimICE-AZ using Web Services. This took the form of some POST and GET calls using Representational State Transfer (REST), that logged into entimICE-AZ and requested a SAS dataset be downloaded across an internet connection.
According to Wikipedia [2] REST is:

> *Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services.*

It goes on to explain that:

> *In a RESTful web service, requests made to a resource's URI will elicit a response with a payload formatted in either HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the stored resource, and the response can provide hypertext links to other related resources or collections of resources. When HTTP is used, as is most common, the operations available are GET, POST, PUT, DELETE, and other predefined CRUD [Create, Read, Update, Delete] HTTP methods.*

At that time, documentation and guidance around the entimICE-AZ Web Services had not been released, and so I began to research RESTful services and find out if it was possible to access them from SAS.

PROC HTTP

Fairly quickly in my search I discovered PROC HTTP, which allows SAS to send and receive HTTP requests.

For each request sent with PROC HTTP there is a Method that can be specified that informs the Web Services Server what you are sending over, so it can be processed accordingly.
The REST methods I could use with entimICE-AZ were as follows:

Method	CRUD Function	Use case
POST	Create	Log in to the system and return a “session ID” for use in future transactions with the server
GET	Read	Download a file, query the contents of a directory or check a session ID is valid/active
DELETE	Delete	Removes Session ID from server

An example of the code required to log in to entimICE-AZ is shown below:

```
*--- Set FILENAMEs up to receive the PROC HTTP output ---;
filename resp TEMP;
filename headout TEMP;

*--- Request login ---;
proc http
   url = "https://entimicews.servername.net/entimicews/rest/sessions"
   out = resp
   method = "POST"
   headerout = headout headerout_overwrite
   ;
   headers "username"="ABCD123" "password"="PASSWORD";
run;

*--- Read responses ---;
data _resp;
   infile resp length=len;
   length text $1000.;
   input text $varying1000. len;
run;
```

```
data _headout;
   infile headout  length=len;
   length text $1000.;
   input text $varying1000. len;
run;
```

The resulting data sets are:

_HEADOUT data set

	text
1	HTTP/1.1 200 OK
2	Server: Apache-Coyote/1.1
3	Content-Type: text/plain
4	Content-Length: 36
5	Date: Wed, 01 Aug 2018 13:59:07 GMT
6	

_RESP data set

	text
1	a05e4989-61ba-406c-a823-ba63e2a0ab09

The **_HEADOUT** data set shows that the request was successful, and the **_RESP** data set contains the Session ID that is associated with the newly logged-in session.
This session ID will then be fed into any subsequent PROC HTTP calls using other methods in order to verify your credentials for each transaction.

Note that, for subsequent examples, this session ID has been assigned to a macro variable, `&session`.

DOWNLOADING DATA

Now that a temporary connection to entimICE-AZ exists, we can request a file for download – in this case a SAS data set. This requires the GET method as follows:

```
*--- Set FILENAMEs up to receive the PROC HTTP output ---;
filename _headout TEMP;
filename _outfile "$files/sasuser.v94/test.sas7bdat";

*--- Request SAS dataset ---;
proc http
   url       = "https://entimicews.servername.net/entimicews/rest/system?path
root/cdar/common/ecd/test.sas7bdat"
   method    = "GET"
   headerout = _headout
   out       = _outfile;
   headers "Session-ID"="&session." "Accept"="application/octet-stream";
run;
```

This downloaded the SAS data set **TEST** and placed it in the user area.

TIMEOUT!

The entimICE-AZ system does have a 120-minute timeout on connections, but keeping connections open when no longer needed ties up resources on the entimICE-AZ servers, which only have a limited number of slots for such connections.
To log out requires the third of the methods discussed earlier – DELETE:

```
*--- Set FILENAMEs up to receive the PROC HTTP output ---;
filename resp TEMP;
filename headout TEMP;

*--- Issue the WebService DELETE command ---;
proc http
  url=" https://entimicews.servername.net/entimicews/rest/sessions/&session."
  method = "DELETE"
  headerout = headout headerout_overwrite
  out=resp;
  headers "Session-ID"="&session." "Accept"="application/json";
run;
```

This removes the session token, effectively logging the user out of entimICE-AZ.

DATA SECURITY

At this point the idea of using PROC HTTP to interface with entimICE-AZ appeared to be a good one. However, it did raise one immediate and very important issue, that of data security.

One of the main rationales for implementing the entimICE-AZ system was to consolidate multiple data storage systems and provide a regulated way to access that data. By downloading data to my user area, I had immediately violated several internal data security policies. Clearly, turning this code into a utility macro and releasing it to the wider department needed more consideration and planning.

The main reason for wanting to download information from entimICE-AZ to the SAS Grid was to perform exploratory analyses on study data. Any regulated output, such as periodic safety reporting (IB or DSUR) or data analysis specified in the relevant study's Statistical Analysis Plan, would need to be done within entimICE-AZ in order to create and maintain the audit trail required for such events. It's worth noting that as we are operating within the early-phase clinical development space, the data sets we use tend to be very small, containing data for only a few dozen patients.

All this pointed towards using the SASWORK area for downloading the data. SASWORK is only accessible to the single user whose SAS session created it and is deleted once the SAS session is closed. Downloading data anywhere else would mean putting in place some sort of access control framework and governance structure, and would run the risk of fragmenting the available data sources and undermining the point of a central data repository.
So, it was decided that data should only be allowed to be downloaded to a user's current SASWORK, and therefore the data would be only temporary.

MACROTISATION

We had some basic SAS code in place to log in to entimICE-AZ, download data sets, and log out (as shown above). It seemed that with some additional work, these could be turned into utility macros that could be released to the wider group.

After careful planning, a lot of coding and several rounds of review we had created the following utility macros[a]:

Descriptor	Macro Name	Comment
chk_session	ecb_entimice_chk_session_001	Check if a given session ID is active
download	ecb_entimice_download_001	Downloads all datasets from a specified entimICE-AZ folder
getdir	ecb_entimice_getdir_001	Return the contents of a specified entimICE-AZ directory
getfile	ecb_entimice_getfile_001	Download a single file
login	ecb_entimice_login_001	Log into entimICE-AZ
logout	ecb_entimice_logout_001	Log out
servers	ecb_entimice_servers_001	Return the server name for a given instance (dev, prod, train)

These formed the building blocks of larger tools that could be released to end-users:

Descriptor	Macro Name	Comment
dir_list	ecb_entimice_dir_list_001	Return a dataset containing a list of directory contents, with the option to do so recursively for a specified number of levels
download_ds	ecb_entimice_download_ds_001	Log in, create a subfolder in SASWORK, download selected data sets from entimICE-AZ, assign a libname, log out
libname	ecb_entimice_libname_001	Log in, create a subfolder in SASWORK, download ALL data sets from a specified folder in entimICE-AZ, assign a libname, log out

PULLING IT ALL TOGETHER

We now had in place a mechanism for connecting to entimICE-AZ from the SAS Grid and downloading data to a secure, temporary area. Initial comments from users were positive, and they found it simple to retrieve their data and do whatever exploratory analysis was required. Inevitably, as users were now able to use SAS data stored within entimICE-AZ for exploratory work, we were asked if it were possible to work cross-platform for regulatory work as well.
The quick answer to that was "no", due to the audit trails and controls deliberately implemented within entimICE-AZ, but this led to discussion around whether it could be possible to design a workflow and some sort of guidance on using the two platforms together.

[a] Note that all macros in Early Clinical Biometrics begin with "ecb_", followed by a second categorical label (in this case "entimice") and suffixed with the macro version number ("001" in these initial versions)

Some examples of where this would be useful would be making complex updates to existing code, or writing a program from scratch – both situations where a programmer would typically write a data step, execute it, view the log and results, go back and make a minor adjustment and repeat. Also, it could be a contingency plan should the SAS controllers in entimICE-AZ go down (a common occurrence during the first few months after release).

Would it be possible to have the same piece of code run on both systems without any additional changes?

WORKING CROSS-PLATFORM

After more investigation, there were two main stumbling blocks to have code that worked cross-platform. The first problem was that of the SAS data itself. In entimICE-AZ, the data locations and libnames are set up automatically using metadata and parameters in the system for each deliverable as you execute code. The SAS Grid is, by comparison, a blank canvas when code is run for the first time. We had managed to get around this, conceptually at least, by using the newly constructed macros to download data from entimICE-AZ to the SAS Grid and assign libnames.

The second problem was that of metadata. entimICE-AZ is a metadata driven system, with much of this metadata used in the execution of SAS code by way of global macro variables. Examples of their use include the location of where outputs should be stored, where logs should be stored, the study number and drug code, the type of deliverable being run, the user name, and so on. What if we could set this up manually in a program when running on the SAS Grid? Maybe some sort of setup or initialisation program that could be run before anything else to simulate the entimICE-AZ environment?

BREAKING IT DOWN

At its most basic level, a typical SAS program we would run consists of 5 steps/concepts:

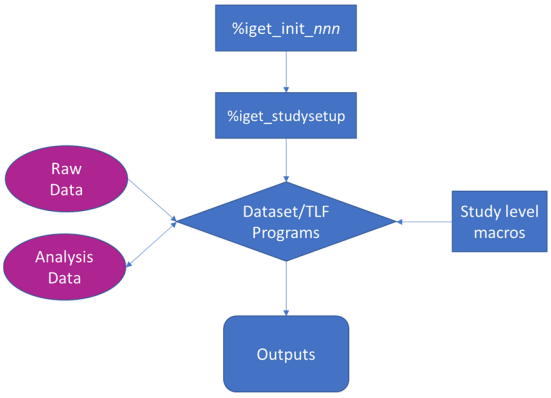
1. Initialisation
2. Setting up utility macros
3. Data acquisition
4. Analysis/manipulation
5. Output (either a derived data set or some sort of report/graphic)

Points 2, 3 and 5, if set up internally in entimICE-AZ, and can be simulated when working on the SAS Grid by way of some sort of setup program. These steps could be bypassed when running in entimICE-AZ as the system will take care of them automatically. Point 4 is the main body of the code, and as both systems are running the same SAS version no changes are needed[b]. This leaves the key ingredient, point 1 – the initialisation file.

We ran through several prototypes before settling on the workflow below. Once we had something that worked, we invited a couple of study teams to try it out and get some of their current code to run on both platforms. The feedback we gathered here was tremendously useful, highlighting many things that we as purely technical programmers hadn't considered when the code was being run in a live study situation.

I GET IT! THE iGET[c] WORKFLOW

The general structure of the iGET programs is shown below. Each program that is to be compatible with both platforms will follow the same layout.



[b] Except for any file paths, but these will be built up with macro variables

[c] As a side note at this point, I wanted to mention the project name – iGET. Early on we needed some sort of name to refer to the project. We looked at several acronyms, but nothing really stuck. There had been a rash of acronyms starting with "i" within the company around that time (the entimICE-AZ project itself was called "iCARE" internally), and as the original macros were there to "get" the data, the new name was born. The umbrella label for the whole project became "iGET", which stands for **I**ntegration of **G**rid and **E**ntimICE **T**ool.

%IGET_INIT

Programs begin with a call to **%iget_init**.
This macro sits at a global level, along with all the other production utility macros and so cannot be edited.
It performs 4 main functions:

1. It determines which platform you are running on, to allow code branching later
2. It updates SASAUTOS so that all macros and tools are available to both platforms[d]
3. It makes the same global system macro variables available to both platforms (for example, entimICE-AZ has a macro variable called &_exec_userid containing the login name of the user. In SAS Grid this is &_clientuserid)
4. It makes any study-level macros available to both systems via an autocall library

After execution of **%iget_init**, the SAS environment on both systems will be virtually identical from a programmer's perspective.

%IGET_STUDYSETUP

The next step, the **%iget_studysetup** macro adds more study-specific information. This includes:

- General SAS options
- Output font sizes (for ODS RTF)
- Setup of format catalogs
- Reset titles/footnotes
- Data reporting period or cutoff dates
- Global macro variables (treatment names, selection criteria, etc.)
- Macro variable specifying output location if running on SAS Grid

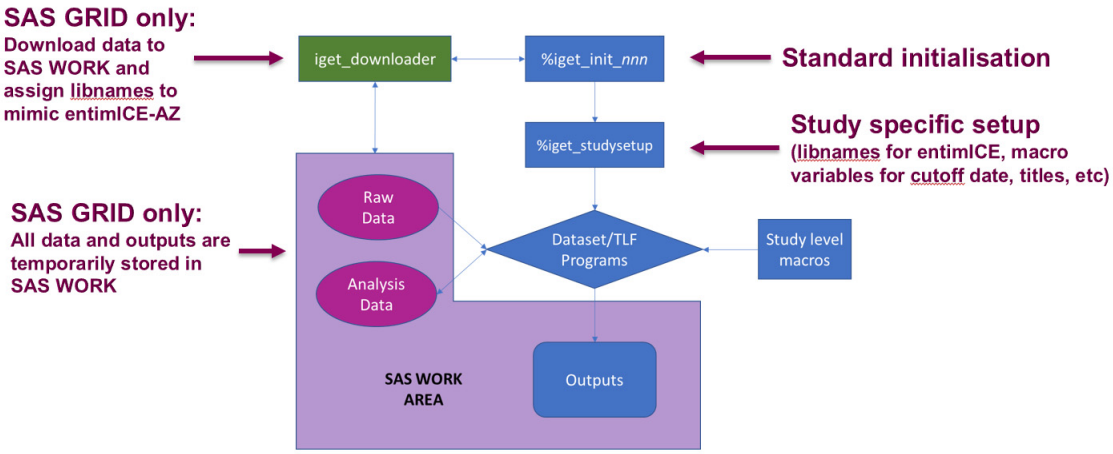
There can be multiple **%iget_studysetup** macros for a particular project, but typically one for each deliverable.

iGET_DOWNLOADER

The next step relates to the SAS Grid only and consists of a SAS program that contains the entimICE-AZ data download macro calls. This program is called FIRST in a SAS Grid programming session, and will download the required data and set up libnames to them.

It is worth noting at this stage that it is expected there would be multiple **iget_downloader** programs written for a particular project. There could be one for a specific deliverable (IB, DSUR, etc.) that only downloads the data that is required for that deliverable. There could be another that downloads all data, or indeed data from multiple sources, in order to perform some exploratory analysis. There could even be **iget_downloader** programs specific to each user, where they can control exactly what data is to be made available to allow them to perform their chosen task quickly.

In the diagram below, you can see that the **iget_downloader** downloads the data and sets up libnames. In the entimICE-AZ system, data is already present and libnames are set up by metadata and so this step is not required.



[d] EntimICE-AZ is our "single source of truth" and holds the master copies of all macros. These are mirrored on the SAS® Grid servers in a dedicated area

THE FINISHED PRODUCT

So now all the pieces are in place, and a template for all SAS programs has been established.
The only thing that differs between the two systems is what a user must do when they begin a SAS session.

In entimICE-AZ this is easy – they do no additional steps, just open their code and start running it as normal, the **%iget_init** and **%iget_studysetup** programs get everything ready for the analysis to be executed.

A SAS Grid user has to perform an extra initial step as soon as they start their SAS session - and that is to run the **iget_downloader** program. This makes SAS data available to the session, and then the **%iget_init** and **%iget_studysetup** program can pick these up and continue with the other setup tasks.

Any subsequent SAS code can be run on both systems unchanged, with identical results.
Note that every SAS program a user creates needs to contains the **%iget_init** and **%iget_studysetup** calls at the very top, but the **iget_downloader** is a separate program that a user executes at the start of a SAS Grid session.

Once developed, code can be easily moved between the systems. As part of the user guide we had to reiterate that entimICE-AZ should house the master copy of any program (in keeping with our new “single source of truth” mentality), and stress to users that code developed on the SAS Grid should be uploaded to entimICE-AZ frequently, and ideally deleted off the Grid servers once tasks had been completed to avoid project fragmentation.

CONCLUSION

It was my intention in writing this paper to show the journey I experienced, starting with an example of a RESTful call to a web services server and learning how to use these with SAS, through developing many utility macros and releasing a handful of user-facing macro tools, that ultimately led to a complete workflow for users wishing to structure their code to make it executable on two independent platforms across different operating systems. Along the way there were a few dead-ends and several technological obstacles, but these were overcome and a fairly robust procedure was established.

ACKNOWLEDGMENTS

I would like to thank my Data Operations colleagues, John McDade and Paul Melon for their assistance in the development and testing of the iGET suite.

REFERENCES

[1] "entimICE," Entimo AG, [Online]. Available: https://www.entimo.com/products/entimice/143-entimice. [Accessed 17 09 2018].

[2] "Representational State Transfer," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed 17 09 2018].

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Nev Cope
AstraZeneca
Da Vinci Building, Melbourn Science Park
Royston
SG8 6HB
United Kingdom
Email: neville.cope@astrazeneca.com