Paper 1517-2017

SAS® Data Integration Documentation Generator: a Capgemini Solution to Accelerate and Keeping it All "in Sync"

Richard Hogenberg, Capgemini

ABSTRACT

A common issue in data integration is that often the documentation and the SAS® data integration job source code start to diverge and eventually become out of sync. At Capgemini, working for a specific client, we developed a solution to rectify this challenge. We proposed moving all necessary documentation into the SAS® Data Integration Studio job itself. In this way, all documentation then becomes part of the metadata we have created, with the possibility of automatically generating Job and Release documentation from the metadata.

This presentation therefore focuses on the metadata documentation generator. Specifically, this presentation:

1) looks at how to use programming and documentation standards in SAS data integration jobs to enable the generation of documentation from the metadata

2) shows how the documentation is generated from the metadata, and the challenges that were encountered creating the code.

I draw on our hands-on experience; Capgemini has implemented this for a customer in the Netherlands, and we are rolling this out as an accelerator in other SAS data integration projects worldwide. I share examples of the generated documentation, which contains functional and technical designs, including a list with all source tables, a list with the target tables, all transformations with their own documentation, job dependencies, and more.

INTRODUCTION

Everyone has probably encountered the situation that the documentation and the corresponding source code were out of sync. This leads to developers often going directly to the source code instead of the documentation because that is the version of the truth and the documentation cannot be trusted. Often developers prefer to put extensive comment sections into the source code for documentation. While that is great it also has its drawbacks as it makes it harder to generate release notes or give someone a documentation set instead of all the source code. Therefore our solution contains the documentation within the SAS Data Integration Studio jobs.

In SAS Data Integration Studio the jobs that are built there are stored within the SAS metadata. There are possibilities to do some documentation within the SAS Data Integration job. If the possibilities are used smartly it is entirely possible to have the SAS Data Integration job fully self documented. We came up with some standards of doing the documentation within the Data Integration job. After that we developed the SAS code that generates a full documentation set from this by analyzing the SAS metadata tree and gathering all the metadata objects and information belonging to the SAS Data Integration jobs we want to documentation from.

DOCUMENTATION STANDARDS

Our first step was setting up some standards on how to do documentation inside a SAS Data Integration job. We concluded there are two levels were we can document on:

1) Job level
2) Transformation level

On a job level we document the general functional and technical details. While on a transformation level we document the more specific details for that specific transformation.

JOB LEVEL

As it is possible that the same SAS environment is used for different solutions it is important that we can differentiate between the metadata information of the different solutions. So it becomes essential that we can easily do a selection for which jobs we want to generate documentation. Job name giving standards must be introduced for this. We do this by having a prefix of two characters followed by an underscore as the first part of the name of the SAS Data Integration job. This way one can use this as the filter to generate documentation for the particular solution. Naming standards will not be further discussed in this paper, but it is essential in keeping a clear solution overview.

We looked at the possibilities of documenting on a job level and identified two options. One makes it hard to see as you need to go to the notes section of the properties of the job. The other springs into view as they are part of the layout of the job. These are the sticky notes. Therefore we choose sticky notes as part of the documentation framework.

Within a SAS DI job one can use sticky notes to write pieces of text and have them visible. It is possible to choose one of three colors for the sticky note in question: Blue, Pink and Yellow. Yellow is normally the default sticky note color. The color can be changed with the help of the menu hidden under the small downward pointing triangle in the left upper corner of the sticky note.

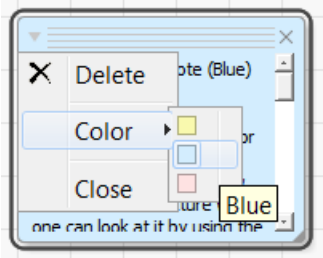

Figure 1 How to change the color of a sticky note

Having the possibility of adding multiple sticky notes to a job gave us the opportunity to differentiate between different types of notes. Too many sticky notes can make a job look cluttered. So use them smartly and do not put too many in one job.

Multiple sticky notes with the same color will be shown in the generated documentation in the order they are added to the job. This is because it is the order that they are stored within the metadata field that contains the sticky notes.

We use the colors of the sticky notes in order to differentiate between the type of information in the sticky note. We used the three available colors in the following way:

- Blue sticky notes contain functional information / decisions

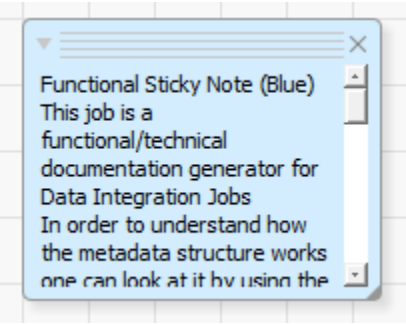

Figure 2 Functional information in a blue sticky note

- Pink sticky notes contain technical information / decisions

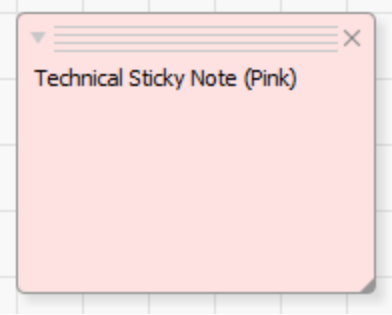

Figure 3 The Pink colored technical information sticky note

- Yellow sticky notes will be ignored for the documentation. These can be used for any information one feels useful to have but which is unnecessary to be in the generated documentation. For instance some notes for future improvements.

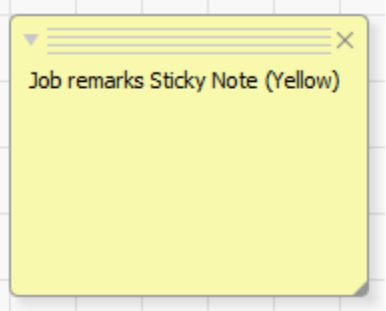

Figure 4 The yellow general remarks sticky note

TRANSFORMATION LEVEL

Changing the name from the default transformation name to a meaningful name will greatly enhance the understanding of a Data Integration job as people can easily see what is supposed to happen in each of the transformations. It also helps in the documentation that we generate to identify the transformation the information is coming from. Having multiple instances of the same type of transformation in a job with the default name will make it hard to link the generated document back to the Data Integration job.

The description of a transformation can only contain a limited amount of characters. This means we can only put a very short description of what the intention of the transformation is in the description of the transformation. We use the notes tab of the transformation as the place to put a full description of the technical / functional details of the transformation.

All user written code of transformations will be fully shown in the generated documentation. For user written code it is useful to ensure that it is properly commented by itself and use the notes section of the transformation for more details.

It is possible to add comments to an expression that is part of a transformation. Expressions are fully shown in the generated documentation. Large comment sections will make it hard to read though.

DOCUMENTATION GENERATOR

The next step was creating a SAS Data Integration job that we could use interactively to do the different tasks we envisioned:

- Generate documentation of Data Integration jobs
- Generate data dictionary
- Generate Job dependency overview

CHALLENGES ENCOUNTERED

After we had set the documentation structure within a SAS Data Integration Job we had the challenge how to get this information out of the metadata. The first issue we encountered is that there is little information available on how SAS Data Integration jobs are actually stored within the SAS metadata.

After some research we found the SAS Metadata Browser. The SAS Metadata Browser can be found within SAS base (which is bundled with SAS Data Integration Studio) by going to the menu Solutions then to Accessoires where the menu item ‘Metadata Browser’ can be found.

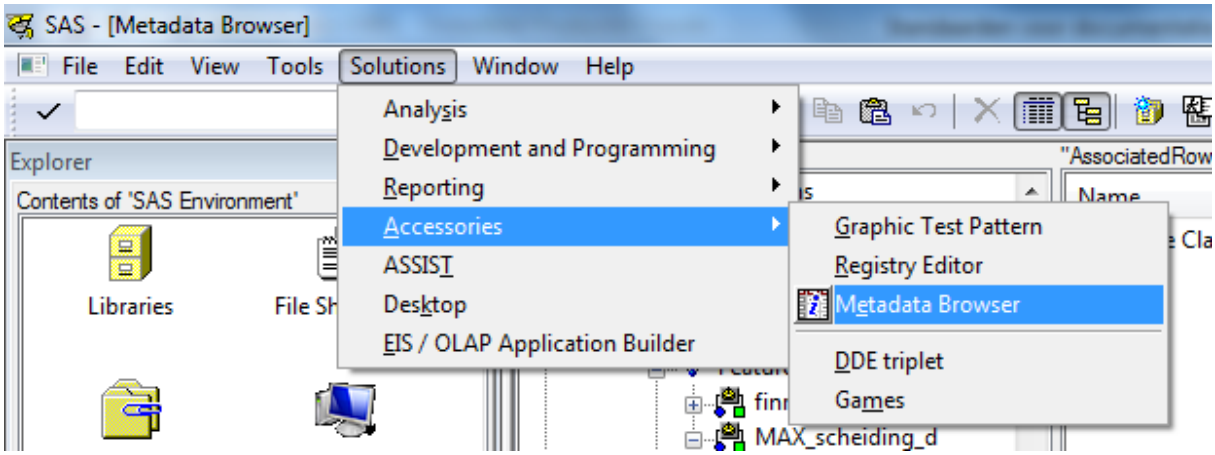

Figure 5 Where to find the SAS Metadata Browser

With the help of the Metadata Browser we got a first insight in how the SAS metadata is structured. Initially we had some issues finding the location of the sticky notes. A tip led us to another useful way to get insight in how SAS metadata is stored.

Archive a SAS Data Integration job by creating an ‘archive as a SAS package’. Then unzip the package. Within the unzipped package there are multiple XML files. One of these XML files contains the complete structure of the Data Integration job. With the help of this we found our missing sticky notes and how we could find the particular object within the SAS meta data.

The sticky notes we found in the DiagramXML part of the job that describes the layout of the job. We discovered that it is show in the Metadata Browser but the browser only shows a limited piece of this object.

After we finally had a good understanding on how the SAS Data Integration job is stored within the SAS metadata we moved on the interesting part of writing a program to extract this information.

The trick to understand what the most efficient of analyzing the tree of information that is available in the SAS metadata is checking what the URI (Universal Resource Identifier) of the object is that you are looking at in the Metadata Browser.

A metadata object consists of properties and associations with other metadata objects. Associations can be used to retrieve the associated objects by using the name of the association and the URI. This is done by looping over the associated objects one by one till no new ones are found.

SAS FUNCTIONS USED

As a short summary the SAS metadata functions that we ended up using and a few code samples.

The functions we used are:

- metadata_getnobj
- metadata_getnasn
- metadata_getnatr
- metadata_getattr

Below is an example how to obtain the uri's of the Data Integration jobs that contain DG_:

```
i=0;
do until(rc1<0);
    i+1;
    rc1=metadata_getnobj("omsobj:Job?@Name contains 'DG_'",i,uri);
    if rc1>0 then do;
          …
    end;
end;
```

The code below shows how to analyze the metadata tree over all the transformations within a job:

```
i=0;
do until(rc1<0);
    i+1;
    rc1=metadata_getnasn("&JobID.","TransformationSources",i,nuri);
    if rc1>0 then do;
          …
    end;
end;
```

THE GENERATED DOCUMENTATION

Each SAS Data Integration job gets its own document generated. While some improvements can be made on the layout of the document it contains the information we wanted it to have. Below is a small sample of what the generated documentation looks like:

Job name
DG_DataIntegrationMetaData_report
Functional Requirements
Functional Sticky Note (Blue) This job is a functional/technical documentation generator for Data Integration Jobs In order to understand how the metadata structure works one can look at it by using the SAS base tool and then in the menu select Solutions --> Accessories --> Metadata Browser In the metadata browser open Foundation and then Job. The entire list with all Data Integration jobs can be seen The functional sticky notes in the job color code Blue, technical sticky notes have the color pink, general job remarks the color yellow. Sticky notes can be found by walking the following associations path: PropertySets --> USERPROPERTIES --> SetProperties --> DiagramXML The attribute DefaultValue contains the entire layout diagram of the Data Integration Job as such it also contains all

of the Sticky Notes information like color and text. The text is HTML formatted

Technical Requirements

Technical Sticky Note (Pink)
A filter is set in the user written code for which jobs to generate documentation. Currently job names that contain 'DS_'

Transformation

ControlOrder=1 Transformation Name=Technical Documentation Generator

TransformRole	SASUserExit
Name	Technical Documentation Generator

…

CONCLUSION

It was challenging to create the documentation generator. But it gave a lot of insight in how SAS Data Integration Studio uses the metadata to store SAS Data Integration jobs. The generation of documentation from the SAS Data Integration jobs makes the lives of the developers a lot easier as they only have to work inside SAS Data Integration Studio and can document while they are creating the job.

REFERENCES

Muriel, Elena. 2009. “Exploring the Metadata Family Tree”. Proceedings of the SAS Global 2009 Conference. Available at http://support.sas.com/resources/papers/proceedings09/097-2009.pdf

RECOMMENDED READING

- *SAS® 9.4 Metadata Model: Reference*
- *SAS® 9.4 Language Interfaces to Metadata, Third Edition*

CONTACT INFORMATION

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

Richard Hogenberg
Capgemini
mailto:richard.hogenberg@capgemini.com
www.capgemini.com