Paper 518-2013

SAS® Metadata Reporting: Extracting Information from SAS Metadata

Jugdish Mistry, J2L Limited, Amersham, UK

ABSTRACT

Within the past few releases of SAS® Software; there has been a trend in fact a big emphasis to use and move to using more and more metadata. It is now, the one stop place for SAS configuration, SAS DI/BI, most SAS applications, and GRID developments. This wonderful method of storing data and managing SAS has no nice GUI for getting this information out. So if we wanted a user list, the name of the last person to update a DI flow, or list new jobs created in the past week, we have to use the appropriate GUI interface for interrogating and reporting on this information. This paper provides practical examples and discusses how using SAS one could extract and generate useful reports from metadata.

This paper provides practical examples and discusses various methods of extracting SAS metadata from the SAS Metadata Server to generate useful reports.

There are several options available for extracting SAS metadata, this paper will, via examples, show how the SAS IOMI interface can be used via BASE SAS PROCs such as PROC METADATA.

It should also be noted that other methods are also available. The DATA step interface and PROC METALIB are other ways to interact with the SAS Metadata Server; these are not in the scope of this presentation.

INTRODUCTION

The SAS Metadata Server stores SAS metadata objects that are predefined within the SAS Metadata model. Information such as server definitions and configuration, SAS DI/Studio jobs, SAS libraries, users and logins to SAS services and applications are all items of data that can be found stored within the SAS metadata server.

Knowing this, one can extract useful information that could solve problems and aid productivity. For instance it might be useful to know who last updated a job written using SAS DI Studio as it is now failing. A Test Manager might want a list of jobs that have been changed and are ready for testing. A SAS administrator might want to investigate why a SAS metadata server predefined library reference (libname) is not setup as defined. The sheer number of reports that could be generated is dependent on ones ability to develop them.

Before reports can be written however one must have an understanding of how the SAS Metadata Server stores metadata objects/types, and how these metadata types interact as associations between types. There is also a requirement to formulate requests in XML and to read the returned XML. The SAS XML Mapper tool can be used to generate the code to read, the XML output created.

SAS METADATA SERVER AND SAS METADATA

The SAS Metadata Server stores its data as metadata types as an Integrated Object Model (IOM) within a SAS repository. Clients interface to the SAS Metadata server via the SAS Open Metadata Interface using SAS Integration Technologies and an appropriate server. Typically a client will communicate with the IServer control interface within the SAS Metadata server to read and write metadata. There are other server interfaces available to administer and query metadata security.

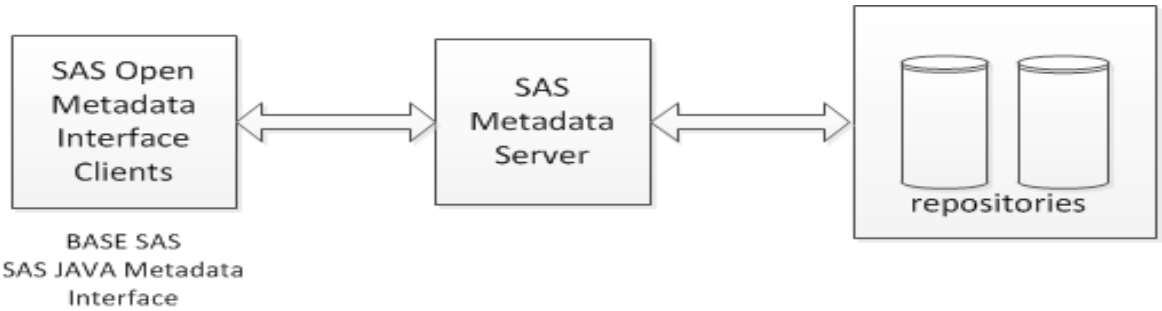

Figure 1. SAS Metadata Server

There are two SAS Metadata repositories created, during SAS installation, REPOS to manage SAS repositories and SAS to manage application data. Metadata types exist within two name spaces, SAS and REPOS. Metadata types are predefined types that have fixed logical definitions that applications then use.

There are two metadata types namely Primary and Secondary. Primary types are objects that can be physically seen and listed in /Systems/Types folder within Management Console and the PublicType attribute is set. A secondary metadata type is not seen and is used to provide additional information for an object.

Examples of a Primary and Secondary types are Job and ForiegnKey respectively.

It is also important to understand that the Metadata model types are object orientated and exist within a hierarchy. All metadata types are subclasses of the Root metadata type. The Root metadata type (and hence all metadata types) attributes are listed below.

Name	Description	Type	Length
Id	Object's repository id.	String	17
Name	A logical identifier for the object. Used for, but not limited to, display.	String	60
Desc	More detailed documentation for this object.	String	200
MetadataCreated	Date and time metadata object was created.	Double	
MetadataUpdated	Date and time metadata object was last updated.	Double	
ChangeState	This attribute is used by Change management.	String	64
LockedBy	This attribute is used by Change management.	String	32
UsageVersion	Usage version is used by an application to indicate the usage pattern of the logical object.	double	

Table 1. Root Metadata type Attributes

Metadata types have associations that group together to define the item of information that needs to be stored and also the relationships between metadata types. For instance the metadata type Job has associations with the metadata types Trees, JFJobs and JobActivities. A Table metadata type would have associations with metadata types Columns.

All associations are two way and have cardinality defined between them. A table can have 0 or more columns. A column must belong to a table.

BROWSING METADATA

SAS Management Console provides an interface to view and manage metadata. Various plugins allow one to view and mange different metadata types. The underlying details of the metadata however, are not surfaced.

The Display Manager provided with BASE SAS can be used to browse metadata by using the METABROWSE command. An “explorer” type window will appear from which metadata types can be viewed. Connection information and metadata credentials will be required. The amount of metadata shown will depend on the userid used to connect to the SAS Metadata Server.

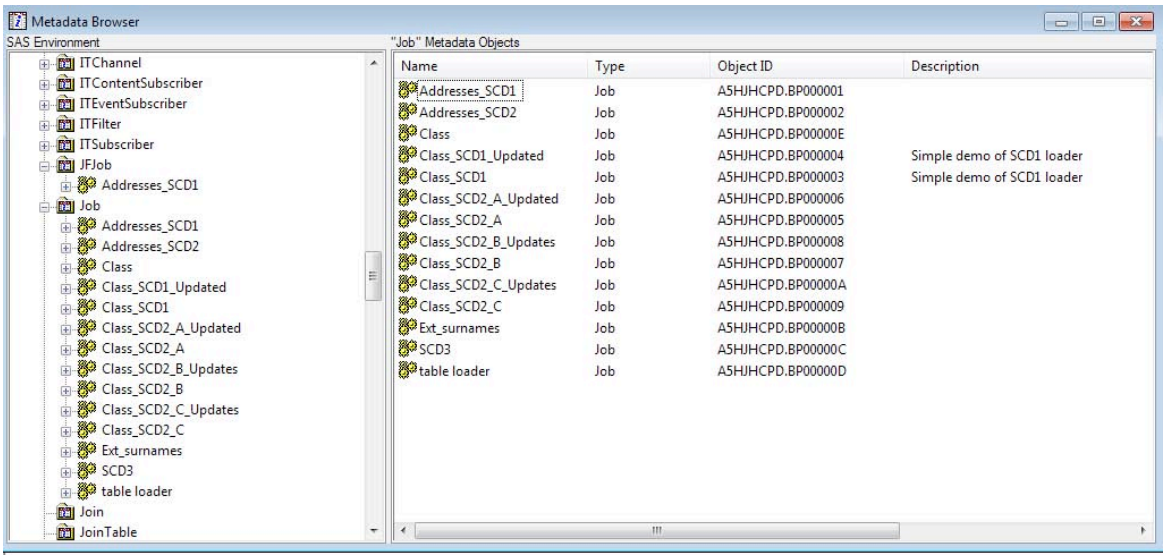

Figure 2. Metadata Browser showing Jobs metadata objects expanded

When a specific type is selected, the attributes and associations can be seen in the opposite pane.

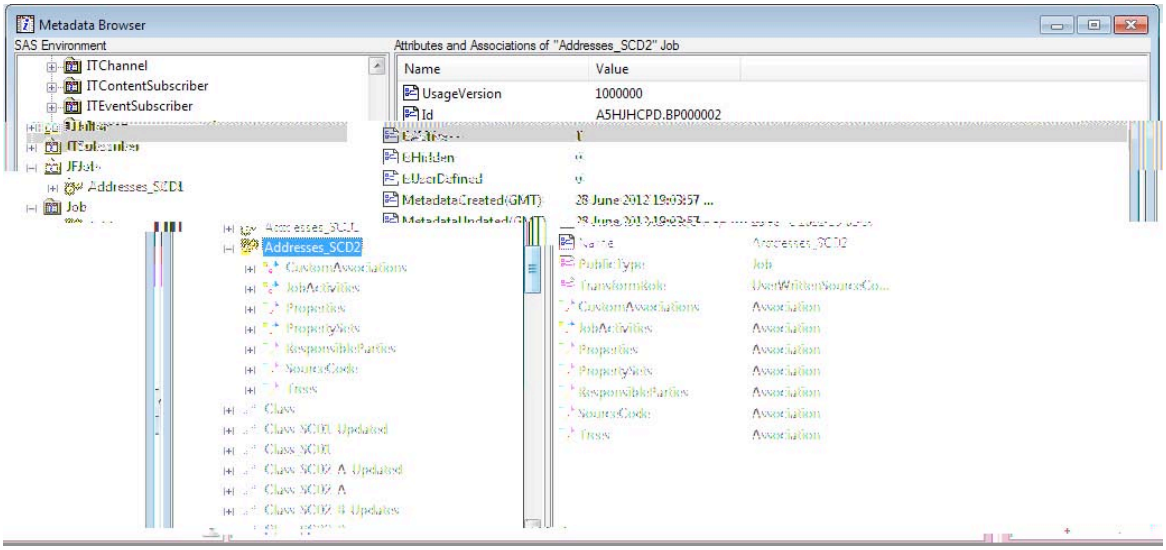

Figure 3. Metadata Browser showing a specific Job metadata object with attributes and associations

IDENTIFYING METADATA

Every item of metadata stored within the metadata server can be identified by its ID, a two-part name, separated by a ('.') dot. The first part of the ID identifies the id of the repository where the metadata resides and second part is the actual id of the object. The ID is very important and is the key attribute for identifying and locating metadata.

In the screen shot above (Figure 3) the metadata Job has id of A5HJHCPD.BP000002

Using metadata type/metadata identifier (type/Id) can also reference a metadata item. This is usually the most efficient method.

In the screen shot above (Figure 3) the metadata Job has id of Job/A5HJHCPD.BP000002

EXTRACTING METADATA

The information given above lays the foundation so metadata can be retrieved from the metadata server. There are several ways to interact with the Metadata Server; this paper will discuss the interaction using PROC METADATA.

The Metadata procedure interacts with the metadata server to read or write metadata. The interaction is via XML, if the correct XML is formed and then passed to the metadata server, the server will apply the request and return further XML containing the results.

To communicate with the Metadata Server, the following options must be set

SAS Option	Value
Metaport	Port id of the metadata server
Metaserver	Address of the metadata server (server name)
Metauser	Userid of the connector
Metapass	Password of the connector
Metarepository	Name of repository to query

Table 2. SAS Options for Metadata Server connection

Example:

With values set by Macro variables

```
options metaserver="&metaserver
        metarepository="Foundation"
        metaport=&metaport
        metauser="&usr"
        metapass="&pw";
```

PROC METADATA sends XML requests to the SAS Metadata Server and accepts the reply from the server. The main parameters are IN=XML string or fileref and OUT=fileref. The IN parameter must contain (or point to) the request XML.

The Metadata Server will parse the method and input parameters, the XML must use published parameters in the XML to represent method parameters. Typically the XML string will comprise of a request (<Get…>) method along with <NS>, <FLAGS> and <OPTIONS> input parameters.

Example: simple call to metadata and partial output to the SAS Log

```
proc metadata
   in='<GetTypes>
       <Types/>
       <Ns>SAS</Ns>
       <Flags>0</Flags>
       <Options/>
       </GetTypes>'
   ;
run;
```

```
<GetTypes><Types><Type
Id="AbstractExtension" Desc="Abstract
extension" HasSubtypes="1"/><Type
Id="AbstractJob" Desc="Abstract job"
HasSubtypes="1"/><Type
Id="AbstractPrompt" Desc="Abstract
prompt" HasSubtypes="1"/><Type
Id="AbstractProperty" Desc="Abstract
property"
HasSubtypes="1"/>
```

If the same output is sent to a file using the OUT=fileref option, the output can be viewed using a browser.

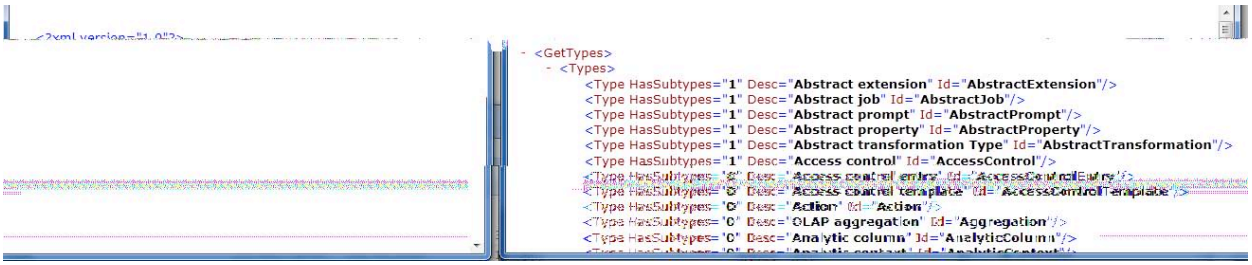

Figure 4. Internet Browser - output from the same GetTypes method.

The above request is sent to the SAS Metadata Server to extract a list of metadata types from the SAS name space. The SAS name space can be changed to REPOS to list metadata types specific to the REPOS name space.

To extract information for a specific metadata object the <GetMetadata> method can be used.

```
proc metadata
   in='<GetMetadata>
       <Metadata>
        <Job Id="A5HJHCPD.BP000001"
Name="" >
        </Job>
        </Metadata>
        <NS>SAS</NS>
        <Flags>0</Flags>
        <Options/>
        </GetMetadata>'
   out=mdoutput
   ;
run;
```

```
<?xml version="1.0"?>
- <GetMetadata>
  - <Metadata>
      <Job Name="Addresses_SCD1" Id="A5HJHCPD.BP000001"/>
    </Metadata>
    <NS>SAS</NS>
    <Flags>0</Flags>
    <Options/>
</GetMetadata>
```

Only attributes listed in the Job element are extracted to the report. To get all attributes for the requested metadata object use OMI_ALL (1) flag.

```
<?xml version="1.0"?>
- <GetMetadata>
  - <Metadata>
    - <Job UsageVersion="1000000" TransformRole="UserWrittenSourceCode" PublicType="Job" MetadataUpdated="28Jun2012:19:03:57"
      MetadataCreated="28Jun2012:19:03:57" LockedBy="" IsUserDefined="0" IsHidden="0" IsActive="1" Desc="" ChangeState=""
      Name="Addresses_SCD1" Id="A5HJHCPD.BP000001">
        <AccessControls/>
        <AssociatedPrompt/>
        <Changes/>
        <ComputeLocations/>
        <ConditionActionSets/>
      - <CustomAssociations>
          <CustomAssociation Desc="" Name="ControlOrder" Id="A5HJHCPD.BQ000001"/>
          <CustomAssociation Desc="" Name="STATELIBRARY" Id="A5HJHCPD.BQ00000N"/>
          <CustomAssociation Desc="" Name="ALT_TEMP_LIBRARY" Id="A5HJHCPD.BQ00000O"/>
        </CustomAssociations>
        <Customizers/>
        <DeployedComponents/>
        <Documents/>
        <Extensions/>
        <ExternalIdentities/>
        <FavoritesContainers/>
        <Groups/>
        <Implementors/>
      - <JFJobs>
          <JFJob Desc="" Name="Addresses_SCD1" Id="A5HJHCPD.C0000001"/>
        </JFJobs>
      - <JobActivities>
          <TransformationActivity Desc="" Name="New Transformation Activity" Id="A5HJHCPD.BT000001"/>
        </JobActivities>
```

Figure 5. Internet Browser output of GetMetadata with Flags OMI_ALL (1) set - partial output only.

Flag	Action
OMI_ALL (1)	Get all attributes and associations that are documented for the metadata type
OMI_FULL_OBJECT (2)	Only valid for Primary Types – expand all associations for the given metadata type *
OMI_TEMPLATE (4)	Signifies that OPTIONS element contains a TEMPLATE definition
OMI_SIMPLE (8)	Get all attributes for metadata type + association if returned.
OMI_XMLSELECT (128)	Request contains a search criteria that needs to be used.
OMI_GET_METADATA (256)	For GetMetadataObjects method, expand each metadata type
OMI_SUCCINCT (2048)	Don't return attributes that have no or null value.

Table 3. A sample selection of useful flags.

Flags can be combined and augmented together, so 2049 would return all attributes and associations but only where there is data to return.

<GetMetadata> method gets metadata for only one specific metadata type, the prerequisite being that the ID of the object is known.

To create a report of all instances of a metadata type the <GetMetadataObjects> method must be used. One also needs to understand the relationship between metadata types to construct the correct XML request. Lets say for example we have a requirement to list the steps along with the table names of all inputs and outputs for every DI Studio Job. This information could be very useful for documentation purposes, understanding the complexity of a job, and hence estimating the amount time and resources needed to test the job.

The metadata browser in BASE SAS comes into it own here; one can find the metadata type and then look at all the associations and sub associations and construct the XML accordingly. This is also an example where the OPTIONS parameter is used along side the TEMPLATES parameter.

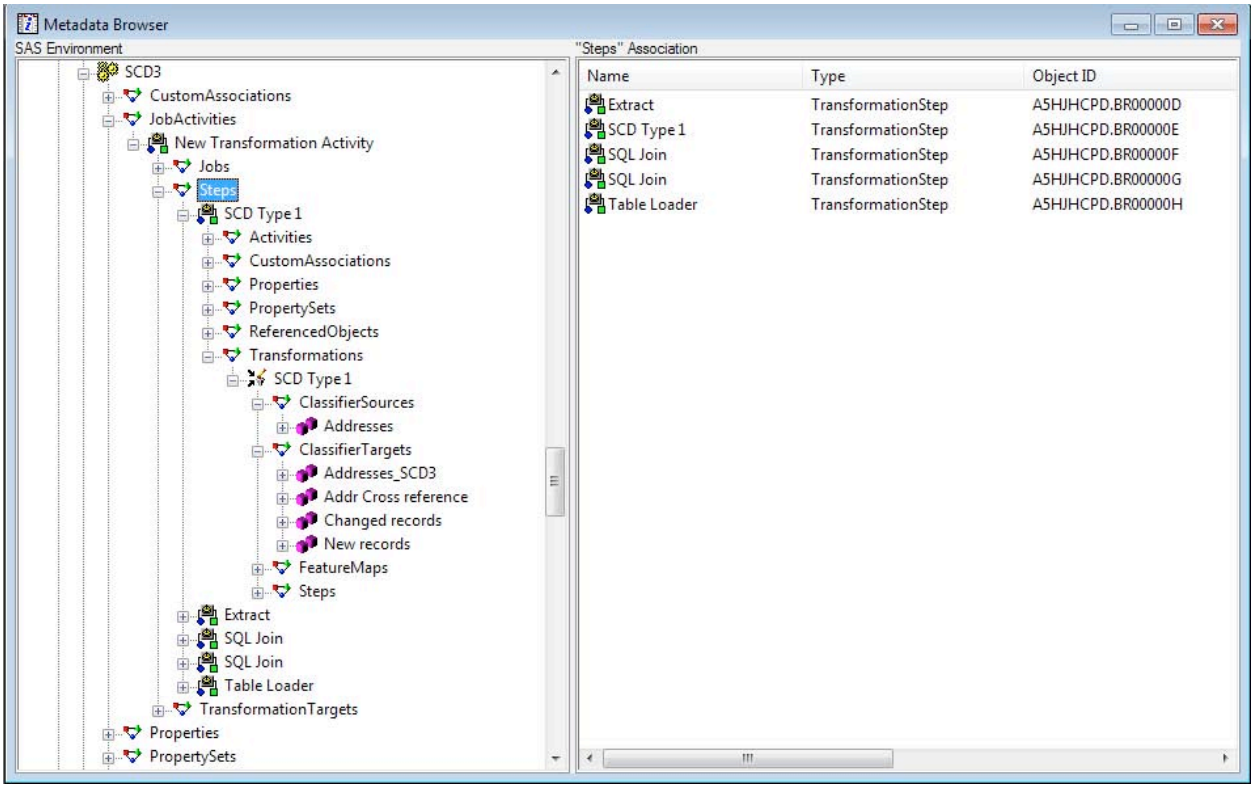

Figure 6. Metadata Browser window – showing associations of a DI Studio Job.

From the example given in Figure 6 above, we can see that a Job has an association with the metadata type JobActivities. The JobActivities metadata type has associations with the TransformActivity, which in turn has an association with Steps and so on. Ultimately we want to traverse down to the ClassifierMap's association with ClassifierTargets and ClassifierSources. The XML required as input for such a report is shown below, each level of association has been indicated by indentations used. The call to PROC Metadata has been omitted.

NOTES:

1. For each association we can list the attributes required in the TEMPLATES element; this way only attributes listed are extracted. The TEMPLATES parameter is used within OPTIONS parameter.
2. Also the FLAGS has been set to 388 OMI_TEMPLATES(4) + OMI_GETMETADATA(256) + OMI_XMLSELECT(128).
3. As the OMI_XMLSELECT flag has been provided, we could optionally request output for a specific job.

```
<GetMetadataObjects>
 <Reposid>$METAREPOSITORY</Reposid>
 <Type>Job</Type>
 <Objects/>
 <NS>SAS</NS>
<Flags>388</Flags>
<Options>
  <Templates>
    <Job Id="" Name="" Desc=""
                  MetadataCreated="" MetadataUpdated="">
      <JobActivities/>
    </Job>


    <JobActivities>
      <TransformationActivity/>
    </JobActivities>

      <TransformationActivity  Name="">
        <Steps/>
      </TransformationActivity>

          <Steps Name="" Id="">
             <TransformationStep/>
          </Steps>

              <TransformationStep Name="" Id="">
                <Transformations/>
              </TransformationStep>

                  <Transformations Name="" Id="">
                    <ClassifierMap/>
                  </Transformations>

                      <ClassifierMap Name="" Id="">
                            <ClassifierTargets/>
                            <ClassifierSources/>
                      </ClassifierMap>

                          <ClassifierTargets Name="" Id="">
                            <PhysicalTable/>
                            <WorkTable/>
                          </ClassifierTargets>

                          <ClassifierSources Name="" Id="">
                            <PhysicalTable/>
                            <WorkTable/>
                          </ClassifierSources>

                              <PhysicalTable Name="" Id="">
                              </PhysicalTable>
```

```
                                        <WorkTable Name="" Id="">
                                        </WorkTable>

  </Templates>
 </Options>
</GetMetadataObjects>
```

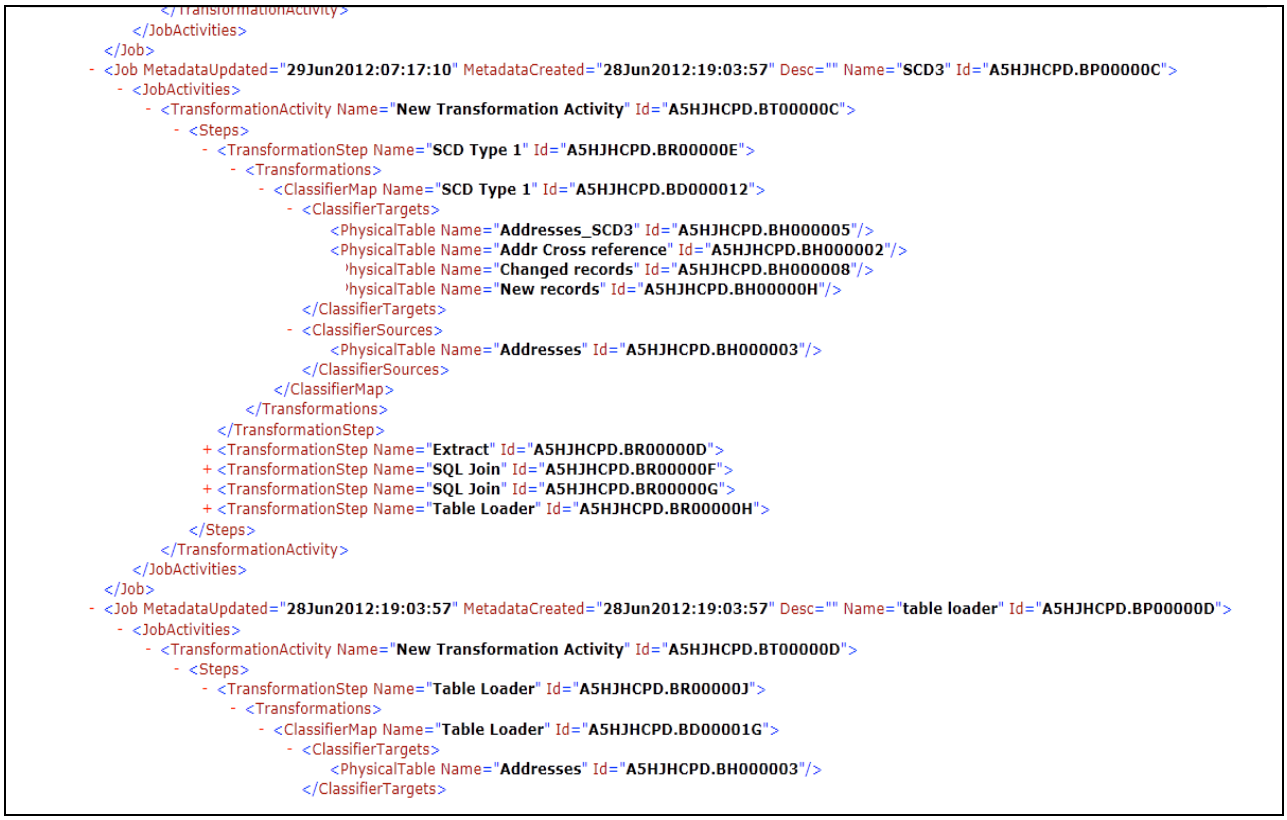

Figure 7. Internet Browser window – showing partial of output for the above example.

CONCLUSION

The topics covered within this paper are just the tip of the iceberg. One could go much further and generate reports for other metadata types, Stored Processes or lists of directories defined; the uses are endless. There is so much information contained in the Metadata Server that it is a crime not to utilize it.

There is potential to create user frontends that surface documentation for business analysts, other developers or displays a data dictionary of all the data being processed by SAS.

There are other techniques available to extract information from the SAS Metadata Server such as using Data Step functions or the SAS Java Interface to metadata and additional methods (GetRepositories, GetSubTypes) for PROC Metadata. It is also possible to create new metadata and change existing metadata. A topic for another time…

ACKNOWLEDGMENTS

To all the people who kept requesting data out of the SAS Metadata Sever, list is too many. Keep asking is what I say…

RECOMMENDED READING

- SAS® 9.3 Metadata Model: Reference
- SAS® Open Metadata Interface: Reference and Usage
- SAS® 9.3Language Interfaces to Metadata

CONTACT INFORMATION

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

Jugdish Mistry
J2L Limited
11 Ash Grove,
Amersham, Buckinghamshire, HP6 5QU

+44 777 567 3000
Jugdish@J2L.CO.UK