Metadata for SAS® 9 Programmers

Elena Muriel, Amadeus Software Ltd, UK

ABSTRACT

Metadata is a fundamental part of SAS9. Each component from data through to delivery is registered in a SAS9 repository and forms a rich source of information just waiting to be exploited. The good news is that this resource is available to any avid SAS programmer.

This paper introduces the fundamental concepts of metadata. These concepts include details on metadata location, structure and how to query and exploit it. The investigation will deliver an overview of key concepts and also provide a basis for further exploration of this important SAS9 resource.

Three different methods will be considered to illustrate how to query the metadata server. Information will be obtained regarding SAS tables registered in a library, column names and primary keys defined within a SAS data set. The first method will use the front end interface provided by SAS Management Console, which provides an interactive entry point to our metadata. The second one will consider SAS data step functions which allow us to query metadata from the standard DMS environment. The third and final method will explore the creation of more complex queries using XML.

The conclusion will analyse the different approaches for interrogating metadata to deliver a code of best practice for metadata queries.

This paper is intended for an advanced SAS user using the SAS9 environment. Examples included in this paper use the SAS Management Console, DI studio and SAS DMS although not all of these products are required to exploit your metadata.

INTRODUCTION

As you are probably aware, absolutely everything that you can see in the SAS9 environment is held in the metadata, from each server defined in the environment, to every column in every table. It works in a similar fashion to the DICTIONARY tables in traditional SAS, but is far more wide reaching. The good news is that you can get to it all programmatically; it just takes a little practice. You can query, add, update & delete pretty much anything you like, but please be careful. These methods are very useful, extremely powerful, and potentially very dangerous.

Remember: BACK UP YOUR METADATA before starting any work!

The actual metadata is held in a series of SAS data sets under the ***MetadataServer\MetadataRepositories\[RepositoryName]*** file system or directory. You can easily open them in SAS, but you won't be able to make any sense of them as they are encrypted. The key to accessing this information is to query your metadata server.

The SAS9 toolset provides three ways to achieve this:

- The SAS Metadata Utility (MDU). A slightly crude but effective front end which can be accessed via the Tools menu in SAS Management console. With it, you are able to query, add, update and delete metadata. This is not a coding tool, but it's great for querying, update metadata with one off fixes and any other ad-hoc queries you might have;

- Data Step functions. SAS9 includes new Data Step functions which have been written specifically to query and update your metadata. Providing an identifier or searching your metadata for specific items, these functions allow you to obtain properties of an object using the SAS Data Step;

- PROC METADATA & XML Maps. This is a more powerful approach to query and update metadata programmatically. This method is based on XML queries, which are passed to the metadata server via the METADATA procedure. XML maps are then used to view the results as SAS data sets via the XML libname engine.

Before we go any further, let's introduce a few situations in which you might want to use metadata manipulation to aid your development:

Part way through a project someone decides to change the format of all the date columns from YYMMDD10 into DATE9. Instead of having to manually edit multiple DI studio jobs and registered tables, the metadata utility allows you to search and extract all the columns which use YYMMDD10. After replacing the format with the new one (using any text processor), the metadata can be updated and all formats changed in one step.

A similar scenario, in which a decision has been made to make sure all column labels are to be 'proper case'. This time we can extract all columns via an XML query, apply the PROPCASE function to each label, and update the registered columns with PROC METADATA.

Or simply use some of the Data Step functions to create data sets than can later be used in a Loop transformation in DI studio or help you when writing macro code.

With metadata manipulation you can control access control entries, server definitions and user accounts without the need of using the SAS Management Console. Also when you get more familiarised with the basics you can even create entire jobs that can be deployed via DI Studio allowing fast and efficient coding.

In the next few sections we would like to cover some of the basics behind the metadata structure and start the journey into metadata by expanding on the different querying methods available. Updates and deletes to metadata can be achieved using similar approaches but they will not be cover in this paper.

METADATA STRUCTURE

Before we can use some of the tools available to extract metadata it is important to understand how this metadata is structured. The SAS Open Metadata Interface is shipped with two namespaces. A namespace is just a metadata model that can be accesses by the Metadata Server. These are:

- REPOS: Defines metadata types specific for repositories;
- SAS: Whose purpose is to define metadata types for the most commonly used SAS application elements. This allows the sharing of metadata between SAS applications. This is the namespace most commonly used by the client applications.

Both namespaces refer to metadata types. Each available metadata type models the metadata for a particular object, where an object is a more familiar thing, like tables, columns, indexes, etc. A metadata type is also structured into hierarchies. So we can find subtypes and subsubtypes defined that extend the behavior of objects.

Each metadata object is described by:

- Attributes or Properties: These are specific characteristics of each object e.g. the format name used by a column in a data set;
- Associations: Describes the relationships that one object can have with others e.g. the relationship between a table and its columns. For example, if you are accessing the **PhysicalTable** metadata type you can use an association called **Columns** to retrieve their variable names. And if you are in the **Column** metadata type you can use the **Table** association to retrieve the name of the table.

Both the SAS and REPOS metadata namespaces define multiple metadata types (over 150). For documentation purposes these types can be grouped in submodels. Submodels try to help in the navigation of the different properties. For example there is a Relational submodel which comprises types that describe relational tables and other objects that can be found in relational database systems such as indexes, columns, keys and schemas.

USING METADATA

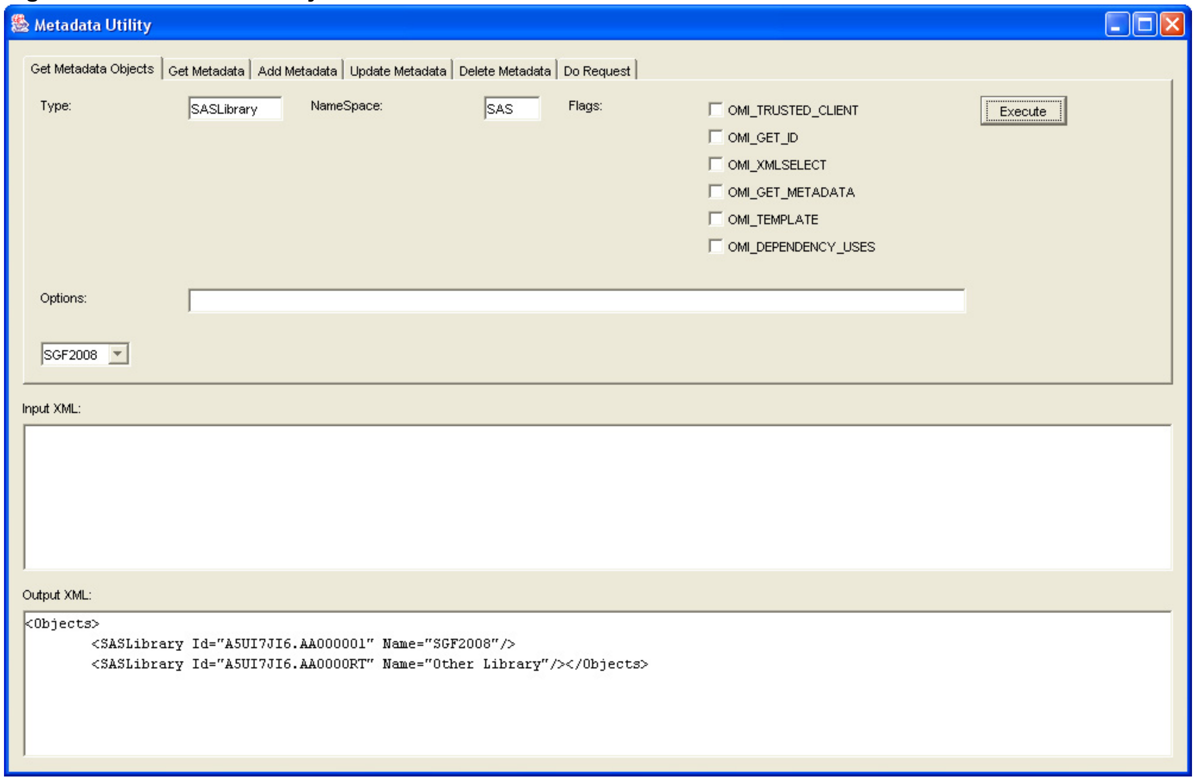
To introduce the three methods for querying your metadata server we are first going to retrieve a list of all data sets registered under a SAS library, also a list of columns contained inside a data set and finally we ask the metadata server for primary keys defined in a SAS data set.

METADATA UTILITY

This invaluable tool can be found under the Tools menu in SAS Management Console. It's not the most intuitive of tools, and there's no help button either, so let's take a quick guided tour.

Upon entry (and a reminder of how powerful this tool can be), we can see several tabs across the top of the application, each relating to a specific request which can be passed to the metadata server. We'll concentrate to the first couple which relate to metadata queries. The others relate to edits which are advanced features you can explore once you are more familiar with how it all fits together.

Figure 1 - Get Metadata Objects



Type: SASLibrary

NameSpace: SAS

Options:

SGF2008

```
<Objects>
        <SASLibrary Id="A5UI7JI6.AA000001" Name="SGF2008"/>
        <SASLibrary Id="A5UI7JI6.AA0000RT" Name="Other Library"/></Objects>
```

GET METADATA OBJECTS

On opening the application, you will always be presented with the **Get Metadata Objects** tab. Think of this as the gateway to the metadata as it allows you to start querying your metadata without an appropriate id.

First let's just query the metadata server for all data sets registered for a given repository. The metadata type used to obtain this information will be the **SASLibrary** for the **SAS** namespace. Select the appropriate metadata repository from the drop down list (Note: The default repository is the one which you connected to via management console) and select Execute. The results will be displayed on the Output XML pane, containing a list of all the libraries with their correspondent ID and Names.

It is also possible to specify filtering criteria with an XMLSelect statement. First you need to inform the metadata server that you would like to filter the results by simply checking the **OMI_XMLSelect** flag. Now you need to enter the search criteria in the **Options** field. To do this you must specify an XMLSelect statement like:

<XMLSelect Search="@Name='SGF2008'"**/>**

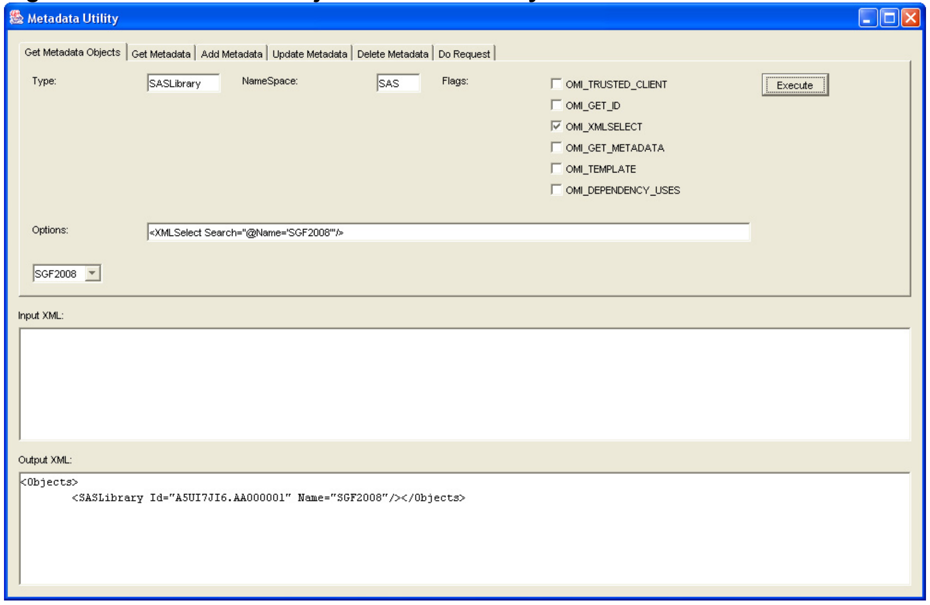
In which you only want to return those libraries which name is SGF2008. You could also search for a library reference starting with the letters SGF. In those cases the statement will change into:

<XMLSelect Search="@Libref=:'SGF'"/>

This is a standard XML tag with a single keyword. Inside the double quotes you can find the statement that's doing all of the work by requesting specific object names. It works just like any other conditional statements, but the operand to the left of the operator is an object property.

You'll also need to be aware that there are only three valid operators that you can use when creating select statements, which are '=', '=:', and '?' (equals, begins with, and contains). There is no concept of 'NOT', so you must always ask for what you want.

Figure 2 – Get Metadata Objects search Library



Type: SASLibrary
NameSpace: SAS
Flags:
- [ ] OMI_TRUSTED_CLIENT
- [ ] OMI_GET_ID
- [✓] OMI_XMLSELECT
- [ ] OMI_GET_METADATA
- [ ] OMI_TEMPLATE
- [ ] OMI_DEPENDENCY_USES

Options: <XMLSelect Search="@Name='SGF2008'"/>

SGF2008

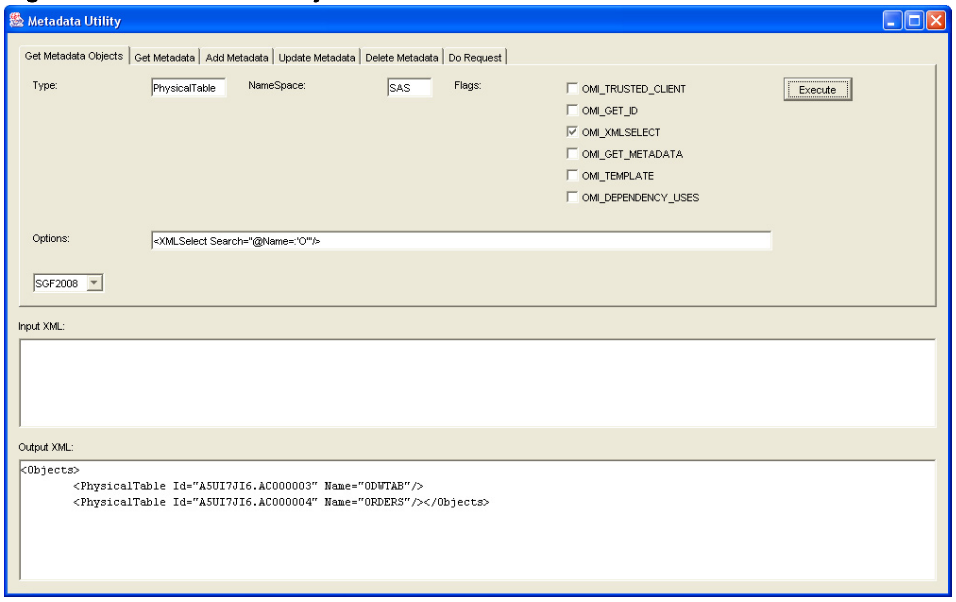
Input XML:

Output XML:

```
<Objects>
       <SASLibrary Id="A5UI7JI6.AA000001" Name="SGF2008"/></Objects>
```

One of the most useful metadata types is the **PhysicalTable** as it allows you to retrieve information on registered tables. Here is another example in which you can request all tables that begin with the letter 'O'.

Figure 3 – Get Metadata Objects search tables



Type: PhysicalTable
NameSpace: SAS
Flags:
- [ ] OMI_TRUSTED_CLIENT
- [ ] OMI_GET_ID
- [✓] OMI_XMLSELECT
- [ ] OMI_GET_METADATA
- [ ] OMI_TEMPLATE
- [ ] OMI_DEPENDENCY_USES

Options: <XMLSelect Search="@Name=:'O'"/>

SGF2008

Input XML:

Output XML:

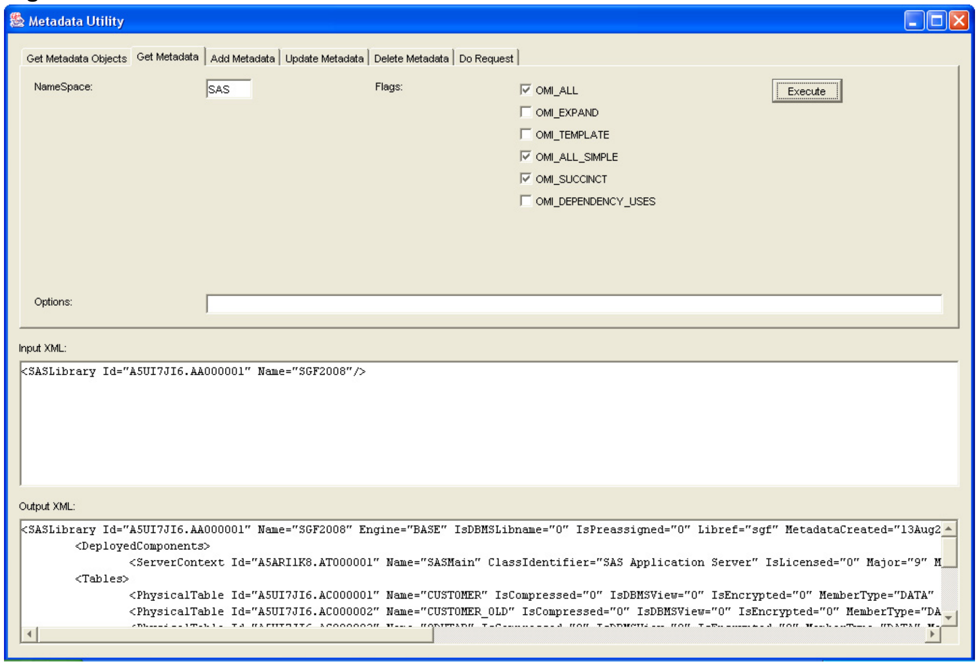
```
<Objects>
       <PhysicalTable Id="A5UI7JI6.AC000003" Name="ODWTAB"/>
       <PhysicalTable Id="A5UI7JI6.AC000004" Name="ORDERS"/></Objects>
```

GET METADATA

Once you have obtained a list of metadata Id's you can now request more details about a specific item. For example let's consider the **SGF2008** library.

To obtain metadata information for a specific object you must specify the unique **Metadata Id**. This can be achieved by simply copying a statement from the output XML pane into the input XML pane (here we are using the statement for the SGF2008 library which we obtained earlier). We may specify other properties in this string (for example 'name') but these are simply used to return the properties in a slightly different order and have no effect on the information returned.

Figure 4 – Get Metadata



NameSpace: SAS

Flags:
[✓] OMI_ALL
[ ] OMI_EXPAND
[ ] OMI_TEMPLATE
[✓] OMI_ALL_SIMPLE
[✓] OMI_SUCCINCT
[ ] OMI_DEPENDENCY_USES

Options:

Input XML:

```
<SASLibrary Id="A5UI7JI6.AA000001" Name="SGF2008"/>
```

Output XML:

```
<SASLibrary Id="A5UI7JI6.AA000001" Name="SGF2008" Engine="BASE" IsDBMSLibname="0" IsPreassigned="0" Libref="sgf" MetadataCreated="13Aug2
        <DeployedComponents>
                <ServerContext Id="A5ARI1K8.AT000001" Name="SASMain" ClassIdentifier="SAS Application Server" IsLicensed="0" Major="9" M
        <Tables>
                <PhysicalTable Id="A5UI7JI6.AC000001" Name="CUSTOMER" IsCompressed="0" IsDBMSView="0" IsEncrypted="0" MemberType="DATA"
                <PhysicalTable Id="A5UI7JI6.AC000002" Name="CUSTOMER_OLD" IsCompressed="0" IsDBMSView="0" IsEncrypted="0" MemberType="DA
```

This tells the metadata server that you would like information about this particular object, but it doesn't specify how much you'd like to bring back. With a **Get Metadata** request you must use the flags to define what level of information you'd like to see.

When extracting metadata there are only three flags that are particularly relevant, and these are **OMI_ALL**, **OMI_ALL_SIMPLE** and **OMI_SUCCINCT**. Here's a quick overview of what information each of the individual flag settings will give us and some sample XML.

Table 1 - Flags

Flag	Metadata Server Response
None	Returns your XML string exactly as you specified it
OMI_ALL_SIMPLE	Returns all of the possible properties for an object
OMI_ALL	Returns all of the possible associations for an object
OMI_SUCCINCT	Suppresses properties and/or associations which are not in use for a particular object.

This XML has been created using a combination of the OMI_ALL and OMI_ALL_SIMPLE flags. The color coding represents the effect of these flags in the results displayed on the Output XML pane.

<SASLibrary Id="A5UI7JI6.AA000001" Name="SGF2008" Engine="BASE" IsDBMSLibname="0" IsPreassigned="0" Libref="sgf">

<AccessControls/>

```
 <Tables>
   <PhysicalTable Id="A5UI7JI6.AC000001" Name="CUSTOMER" Desc="" IsCompressed="0" IsDBMSView="0"
          IsEncrypted="0" MemberType="DATA" NumRows="-1" SASTableName="CUSTOMER"
          TableName="CUSTOMER"/>
  </Tables>

 <Trees>
   <Tree Id="A5UI7JI6.A6000003" Name="SGF2008" TreeType="BIP Folder"/>
 </Trees>

 <UsingPackages>
   <Directory Id="A5UI7JI6.AB000002" Name="Path" DirectoryName="D:\SGF2008\data" IsRelative="0"/>
 </UsingPackages>

</SASLibrary>
```

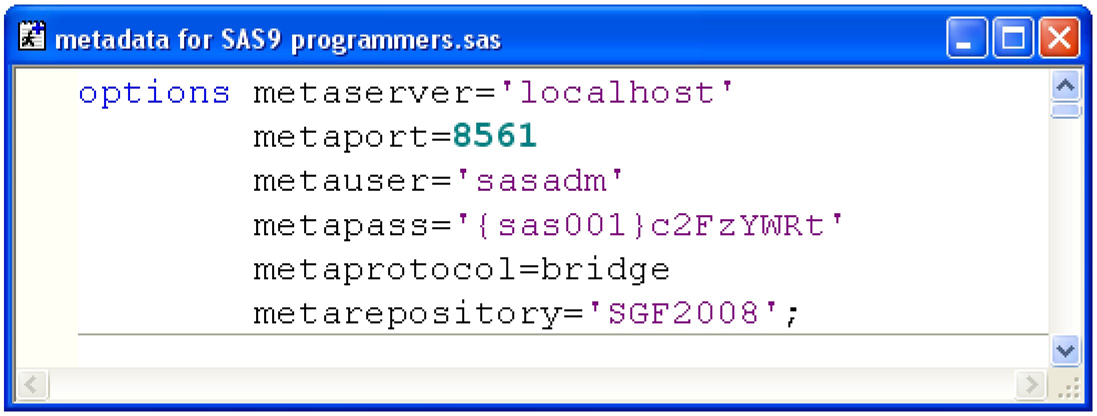
This is a direct way of querying metadata, but sometimes you might need to create a program that can be reused to obtain metadata definitions.

CONNECTING TO METADATA SERVER

When using metadata Data Step functions and XML queries you might need to connect to the metadata server. In those situations you have to define a few server connection parameters using an options statement. Normally, you need to provide the following information:

- Metaserver: Name of the server where the metadata server is running;
- Metaport: Port number in which the metadata server is running;
- Metauser: User name to use in the connection;
- Metapass: User password. Remember that you can always encode your password by using the PROC PWENCODE procedure;
- Metaprotocol: Protocol to use in the connection;
- Metarepository: Name of the metadata repository to connect to.

Figure 5 – Connection parameters



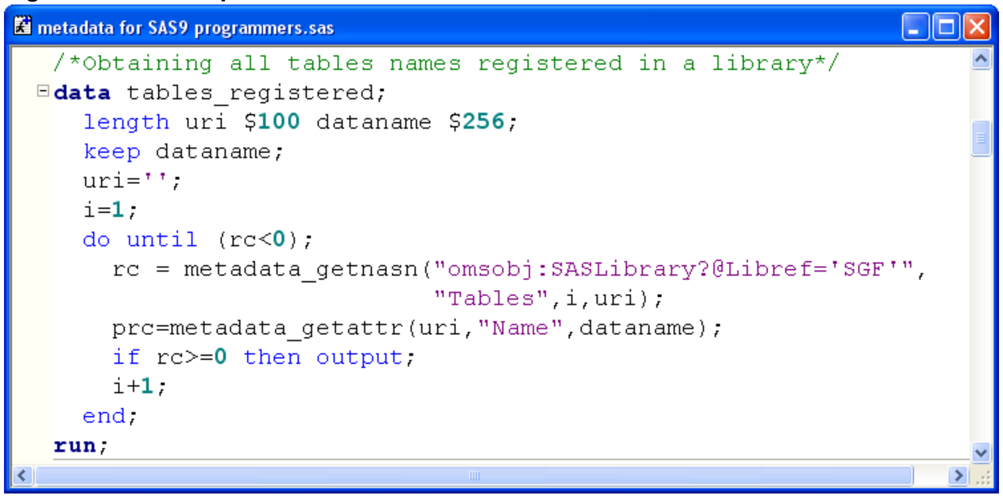
```
options metaserver='localhost'
        metaport=8561
        metauser='sasadm'
        metapass='{sas001}c2FzYWRt'
        metaprotocol=bridge
        metarepository='SGF2008';
```

METADATA DATA STEP FUNCTIONS

There is a second method for exploiting metadata that uses the normal Data Step environment in a regular SAS session. Data Step functions can be useful when defining your own transformations or custom code in DI studio, or simply creating macro code. Remember to connect to your metadata server first.

Let's obtain information about all the tables registered inside a SAS library. Consider the following code:

Figure 6 – Data Step all tables



```
/*Obtaining all tables names registered in a library*/
data tables_registered;
  length uri $100 dataname $256;
  keep dataname;
  uri='';
  i=1;
  do until (rc<0);
    rc = metadata_getnasn("omsobj:SASLibrary?@Libref='SGF'",
                          "Tables",i,uri);
    prc=metadata_getattr(uri,"Name",dataname);
    if rc>=0 then output;
    i+1;
  end;
run;
```

In here there are two specific metadata functions:

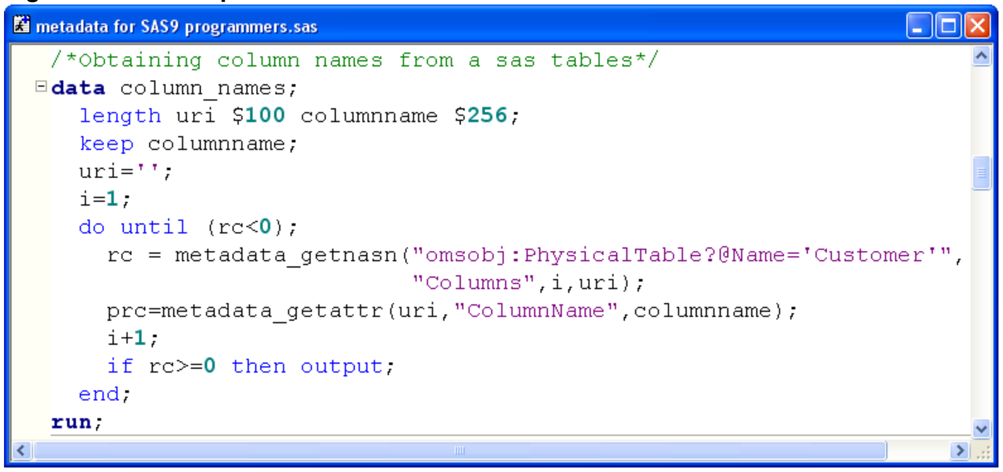
- Metadata_getnasn: Returns the identifiers of all the Tables registered under the SGF library reference name. The first parameter specifies the metadata type you are after (**SASLibrary**) and searches for a specific library (using the libref). **Tables** is the name of the association you are trying to retrieve. The output variable URI (Uniform Resource Identifier) contains the identifier value;
- Metadata_getattr: Once you have obtained the identifier you can interrogate that object for an attribute, in this case the **Name**.

By using some loop processing you can populate the data set with all the table names registered under a library.

The same information is available in the Metadata Utility tool. Just request to see the **SASLibrary** metadata type in the **Get Metadata Objects** tab and then include the results obtained for the SGF library in the **Get Metadata** tab.

If you want to obtain information about the column names stored in a given data set then you will need to change the name of the metadata type to interrogate. In this case from **SASLibrary** into **PhysicalTable**, obtain the ID of the columns and request the **ColumnName** attribute.

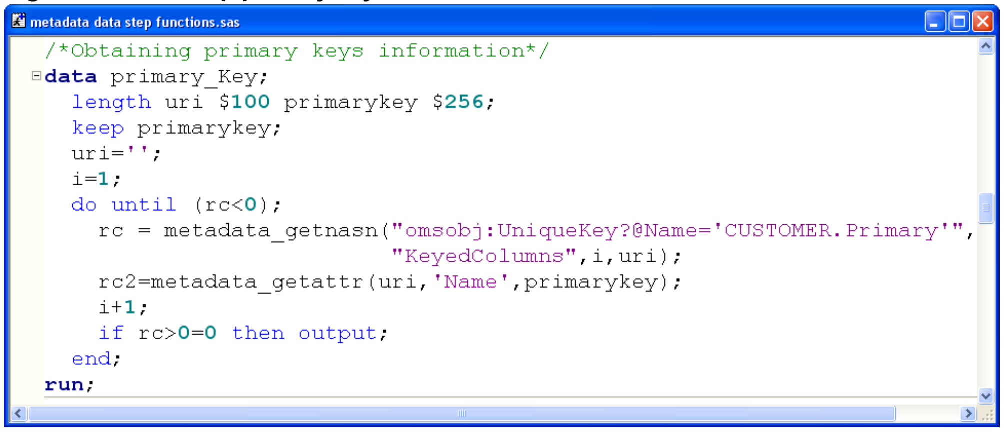
Figure 7 – Data Step columns



```
/*Obtaining column names from a sas tables*/
data column_names;
  length uri $100 columnname $256;
  keep columnname;
  uri='';
  i=1;
  do until (rc<0);
    rc = metadata_getnasn("omsobj:PhysicalTable?@Name='Customer'",
                          "Columns",i,uri);
    prc=metadata_getattr(uri,"ColumnName",columnname);
    i+1;
    if rc>=0 then output;
  end;
run;
```

And finally, if you are trying to get information about the primary keys included in a data set then use the following example:

Figure 8 – Data Step primary keys


metadata data step functions.sas

```
/*Obtaining primary keys information*/
data primary_Key;
  length uri $100 primarykey $256;
  keep primarykey;
  uri='';
  i=1;
  do until (rc<0);
    rc = metadata_getnasn("omsobj:UniqueKey?@Name='CUSTOMER.Primary'",
                          "KeyedColumns",i,uri);
    rc2=metadata_getattr(uri,'Name',primarykey);
    i+1;
    if rc>0=0 then output;
  end;
run;
```

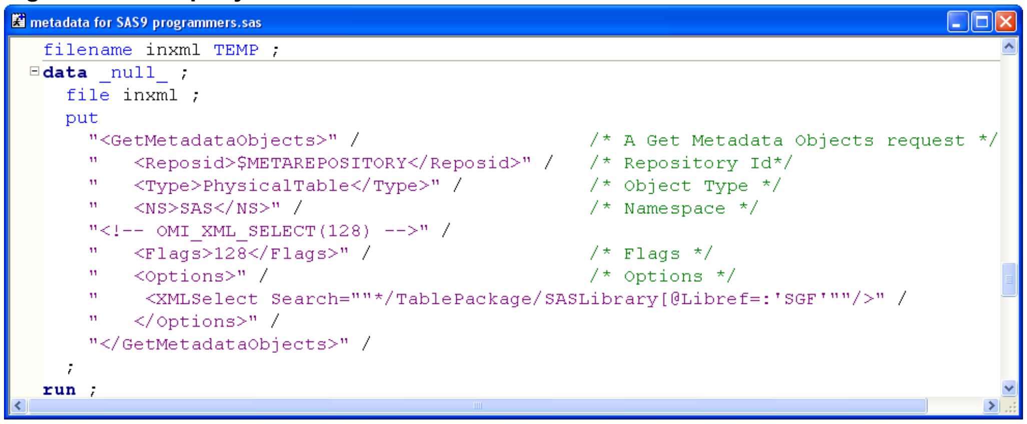
Which uses the **UniqueKey** metadata type for the CUSTOMER data set and interrogates for the Primary key and the **KeyedColumns**

XML & PROC METADATA

This third method is the most advanced and flexible when it comes to query metadata. This is because it is based around XML and the SAS Open Metadata Interface uses this language as its means of transport. You can therefore perform XML queries to the metadata server and the results will also be given back in an XML format.

The first thing you need to define is the XML string which contains the query to ask. As shown on previous examples let's extract the ID and Name of all data sets registered under the **SGF** library.

Figure 9 – XML query


metadata for SAS9 programmers.sas

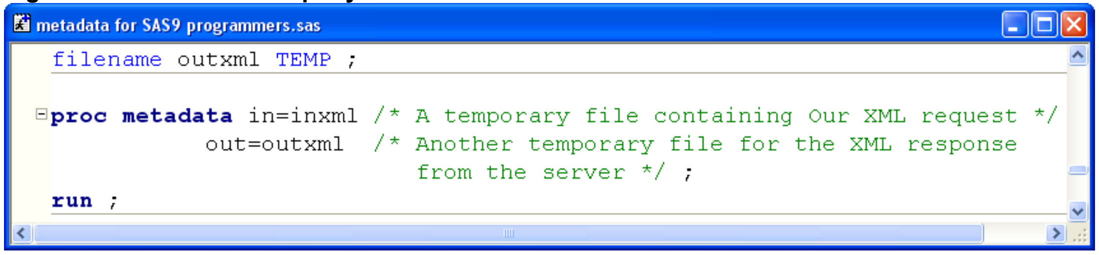
```
filename inxml TEMP ;
data _null_ ;
  file inxml ;
  put
    "<GetMetadataObjects>" /                    /* A Get Metadata Objects request */
    "   <Reposid>$METAREPOSITORY</Reposid>" /   /* Repository Id*/
    "   <Type>PhysicalTable</Type>" /           /* Object Type */
    "   <NS>SAS</NS>" /                         /* Namespace */
    "<!-- OMI_XML_SELECT(128) -->" /
    "   <Flags>128</Flags>" /                   /* Flags */
    "   <Options>" /                            /* Options */
    "    <XMLSelect Search=""*/TablePackage/SASLibrary[@Libref=:'SGF']""/>" /
    "   </Options>" /
    "</GetMetadataObjects>" /
  ;
run ;
```

Most of the tags included in the XML file have already been covered in the Metadata Utility section; however there are a couple things that need a little more information:

- GetMedataObjects: This is the same as performing a query in the **Get Metadata Objects** tab of the metadata utility;
- $METAREPOSITORY: This parameter is evaluated by the metadata server to the Id of the currently active repository (i.e. the one you're connected to). In 99% of cases this is the exactly what you want, though you may also specify the physical Id of a repository if required;
- PhysicalTable: This is the metadata type you want to access;
- SAS. Type of Namespace to be used;
- 128. This number represents the OMI_XML_SELECT flag. Behaves like if you had selected this flag on the metadata utility. Other examples of flags numbers are OMI_TEMPLATE (4) and OMI_GET_METADATA (256). If you need to set multiple flags then just add all of the relevant values;
- XMLSelect: Selection criteria for picking up tables. The search path needs to use the association paths between a library and a physical table to be able to relate the two of them.

The next step is to pass this request to the metadata server, which you must do with PROC METADATA.

Figure 10 – Execute XML query



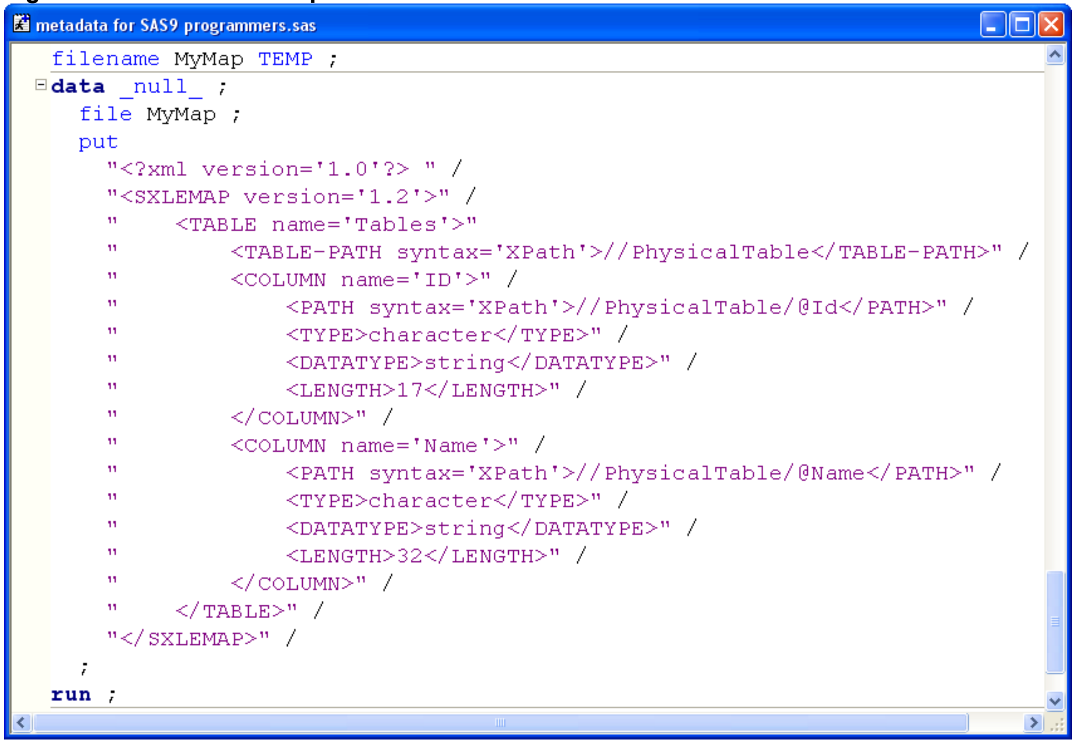
metadata for SAS9 programmers.sas

```
filename outxml TEMP ;

proc metadata in=inxml /* A temporary file containing Our XML request */
              out=outxml  /* Another temporary file for the XML response
                             from the server */ ;
run ;
```

You can open the results generated in the OUTXML file by navigating through the File Shortcuts in the SAS explorer window and opening the file. The information displayed is a single XML statement containing all the results. As this is not very easy to read you can define an XML map and an XML library to read this file as a SAS data set. The map will basically define what the SAS data set will look like, and where it will get its data from (although it will not actually create a SAS table).

Figure 11 – Define XML map



metadata for SAS9 programmers.sas

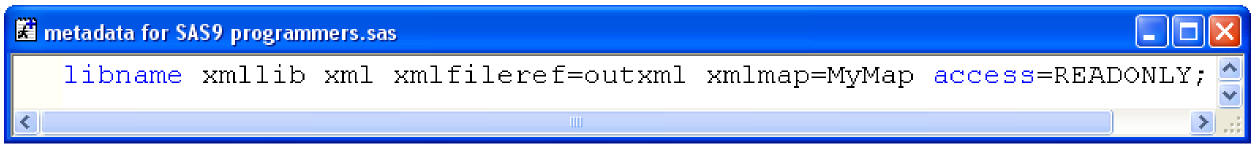
```
filename MyMap TEMP ;
data _null_ ;
  file MyMap ;
  put
    "<?xml version='1.0'?> " /
    "<SXLEMAP version='1.2'>" /
    "     <TABLE name='Tables'>"
    "         <TABLE-PATH syntax='XPath'>//PhysicalTable</TABLE-PATH>" /
    "         <COLUMN name='ID'>" /
    "             <PATH syntax='XPath'>//PhysicalTable/@Id</PATH>" /
    "             <TYPE>character</TYPE>" /
    "             <DATATYPE>string</DATATYPE>" /
    "             <LENGTH>17</LENGTH>" /
    "         </COLUMN>" /
    "         <COLUMN name='Name'>" /
    "             <PATH syntax='XPath'>//PhysicalTable/@Name</PATH>" /
    "             <TYPE>character</TYPE>" /
    "             <DATATYPE>string</DATATYPE>" /
    "             <LENGTH>32</LENGTH>" /
    "         </COLUMN>" /
    "     </TABLE>" /
    "</SXLEMAP>" /
  ;
run ;
```

The main statements needed for the creation of this XML map are:

- TABLE-PATH: Defines the observation boundary. This means that every time a close **PhysicalTable** tag is found in the OUTXML file, a new record will be created in the SAS data set;
- PATH: The path statements included under the COLUMN tags defines which object, and more importantly which property of this object contains the data you want to use for the current column. For this map only the ID and the Name of the table are considered;
- The other statements relate the properties of the SAS data set to create such as table name, column name, type of variable and length.

This XML map creates a file containing XML tags that tells the SAS XML libname engine how to interpret an XML document. To associate the results obtained from the PROC METADATA procedure and the map simply assign a library using the XML engine. This allows you to view the information as a SAS data set.

Figure 12 – Assign XML libname engine



```
libname xmllib xml xmlfileref=outxml xmlmap=MyMap access=READONLY;
```

CONCLUSION

Successful exploitation of metadata can be an extremely useful skill. A first step towards this is the ability of querying your metadata server.
Ad-hoc queries and basic navigation can be achieved by using the Metadata Utility tool included in the SAS Management Console client.
Basic information can be obtained programmatically using the new Data Step functions. Whilst more advanced and complex queries can be performed using the native metadata language of XML and the PROC METADATA procedure.

As confidence grows when using these new techniques more complex task such as editing, adding and deleting metadata can be attempted.

REFERENCES

SAS 9.1 Open Metadata Interface Reference
SAS 9.1 Open Metadata Interface: User's Guide
SAS 9.1 Help

ACKNOWLEDGMENTS

The authors would like to thank Hadley Christoffels, who introduced us to the intricate art of metadata manipulation, and also Iain Knowles with whom we shared the learning experience.

Thanks also to David Shannon for his critical review of the paper.

CONTACT INFORMATION

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

Author Name:	Elena Muriel
Company:	Amadeus Software Limited
Address:	Mulberry House, 9 Church Green, Witney, Oxon OX28 4AZ
Work Phone:	+44 (0) 1993 848010
Email:	Elena.muriel@amadeus.co.uk
Web:	www.amadeus.co.uk