Paper 097-2009

Exploring the Metadata Family Tree

Elena Muriel, Amadeus Software, UK

ABSTRACT

An untapped resource of metadata functionality exists within the grasp of the SAS programmer. At first glance this rich mine of information is hidden from view, however metadata functions can extract this valuable information for a variety of different uses. All of which is available within the framework of the humble SAS data step.

Information extracted can vary from creating customised autoexec files based on a metadata group to the development of customised Enterprise Guide plug-ins that send automatic emails to groups defined in metadata. The metadata can even be queried to perform authentication and authorisation.

The paper will explain how to use the metadata functions available in the data step environment and highlight different uses of the information that can be extracted. It will also describe the basics of writing metadata queries via the Metadata Utility within the SAS Management Console.

This paper is for SAS programmers who wish to learn how the information is stored in the metadata server and how can this be extracted.

INTRODUCTION

The metadata server is a centralised storage centre for the SAS®9 world. Predominately, SAS administrators and developers enter information into the metadata server by registering tables, users and more, but they don't get anything back in return. This paper will show how the metadata server interaction can be a two way process.

The information that has been stored in metadata repositories can also be queried and retrieved when needed. Some of the basic methods for achieving this include:

- SAS Management Console Metadata Utility. This is an extremely useful facility to browse and control the structure of the metadata stored in the repositories.
- XML queries. Using the native language to communicate with the metadata server. XML is not intuitive in its construction but it is a very powerful tool.
- Data step functions. Allowing SAS programmers to exploit the data step environment to interact with the metadata server. This is the environment where most traditional SAS programmers will feel more comfortable.

The approach considered in this paper combines the use of the Metadata Utility and the data step functions to retrieve metadata. It will illustrate how to obtain the email addresses of a group of users and how to control the application server autoexec file based on the user launching the SAS session. The latter can be extremely useful when there are different user groups that need to automatically assign formats catalogs based on different departments.

METADATA STRUCTURE

Before a query can successfully be created, it is necessary to understand the basics of how the information is structured and stored in the metadata repositories.

All metadata is stored in a centralised repository that can be accessed by client applications using the SAS Open Metadata Interface. In terms of defining metadata structure, two metadata models (called Namespaces) can be used, REPOS and SAS. The SAS namespace is the most commonly used as it contains SAS application elements. The REPOS namespace relates to metadata specific to repositories structures.

Metadata objects will be added and stored using one of the available Namespaces. Each metadata object also has a set of properties (or attributes) and associations that can be accessed. A property describes a metadata object stored in the repository. For example, if the object is a library, properties include the name and description.

An association is the relation of that metadata object with other objects present in the repository. In the example of a library, an association can be the tables registered within the library.

One of the first valuable examples that can be created is one that obtains a list of all the types of objects that exist within a model. This gives us an idea of the different categories that are used to store information into the metadata server.

From a DMS session, a set of connection parameters must be provided to determine which metadata server and repository needs to be accessed. These can be set up using an **OPTIONS** statement.

```
options metaserver='localhost'
        metaport=8561
        metauser='elena.muriel'
        metapass='{sas001}c2FzYWRt'
        metaprotocol=bridge
        metarepository='Foundation';
```

The first metadata function to cover is **METADATA_GETNTYP**, which returns an object type from the server and stores the result under the Type variable. The following example illustrates how to read all possible object types:

metadata.sas

```
/*Lists all types of objects that are available*/
data _null_;
  length type $256;
  do until(rc<0);
    i+1;
    rc=METADATA_GETNTYP(i, type);
    if rc>0 then put type=;
  end;
run;
```

The extensive list obtained is included below.

Output - (Untitled)

Object Type	Object Type	Object Type	Object Type
AbstractExtension	EMModel	LogicalColumn	SASCatalogEntry
AbstractJob	EMRules	LogicalServer	SASClientConnection
AbstractProperty	Event	Login	SASFileRef
AbstractTransformation	Extension	Machine	SASLibrary
AccessControl	ExternalIdentity	Measure	SASLicense
AccessControlEntry	ExternalTable	Memory	SASPassword
AccessControlTemplate	Feature	MiningResult	SecurityRule
AggregateAssociation	FeatureMap	NamedService	SecurityRuleScheme
Aggregation	File	NumericExtension	SecurityTypeContainmentRu
AnalyticColumn	FitStatistic	OLAPProperty	Select
AnalyticTable	ForeignKey	OLAPSchema	ServerComponent
ArchiveEntry	Group	OnClause	ServerContext
ArchiveFile	GroupByClause	OpenClientConnection	ServiceComponent
AssociationProperty	HavingClause	OrderByClause	ServiceType
AttributeProperty	Hierarchy	Permission	SoftwareComponent
AuthenticationDomain	Identity	PermissionCondition	StepPrecedence
Change	IdentityGroup	Person	Stream
Classifier	Index	Phone	SummaryStats
ClassifierMap	ITChannel	PhysicalTable	SXLEMap
Column	ITContentSubscriber	Property	SyncStep
ColumnRange	ITEventSubscriber	PropertyGroup	TableCollection
COMConnection	ITFilter	PropertySet	Target
ConditionalPrecedence	ITMap	PropertyType	TCPIPConnection
ConfiguredComponent	ITModel	Prototype	Text
Connection	ITMsmqModel	PrototypeProperty	TextStore
ContentLocation	ITQueueAlias	PSColumnLayoutComponent	Timestamp
ContentType	ITRendModel	PSGridLayoutComponent	Transformation
Cube	ITSubscriber	PSLayoutComponent	TransformationActivity
DatabaseCatalog	ITTransportAlias	PSPortalPage	TransformationStep
DatabaseSchema	JFJob	PSPortalProfile	Tree
DataSourceName	Job	PSPortlet	UniqueKey
DataTable	Join	QueryClause	UnitofTime
DeployedComponent	JoinTable	QueryTable	Variable
DeployedDataPackage	Key	RelationalSchema	WhereClause
Device	KeyAssociation	RelationalTable	WorkTable
DeviceType	Keyword	Report	XPath
Dimension	Level	ResponsibleParty	
Directory	LocalizedResource	Role	
Document	LocalizedType	RowSelector	
Email	Location	SASCatalog	

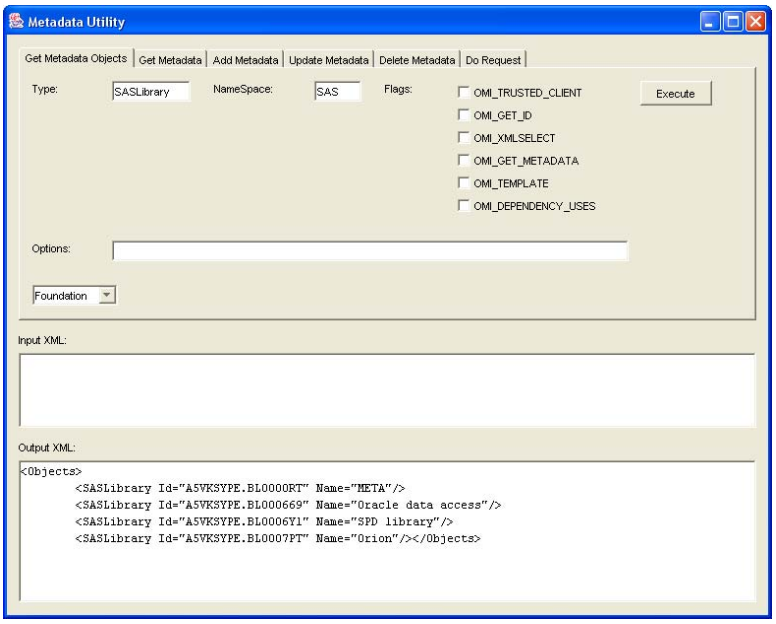
The starting point of any query is to know what information is required and under which metadata object type this is stored. This includes secure information that can only be accessed by users with appropriate credentials. For example, requests for user login details (stored under the Login type) can only be accessed by an unrestricted user or the user owning those login properties.

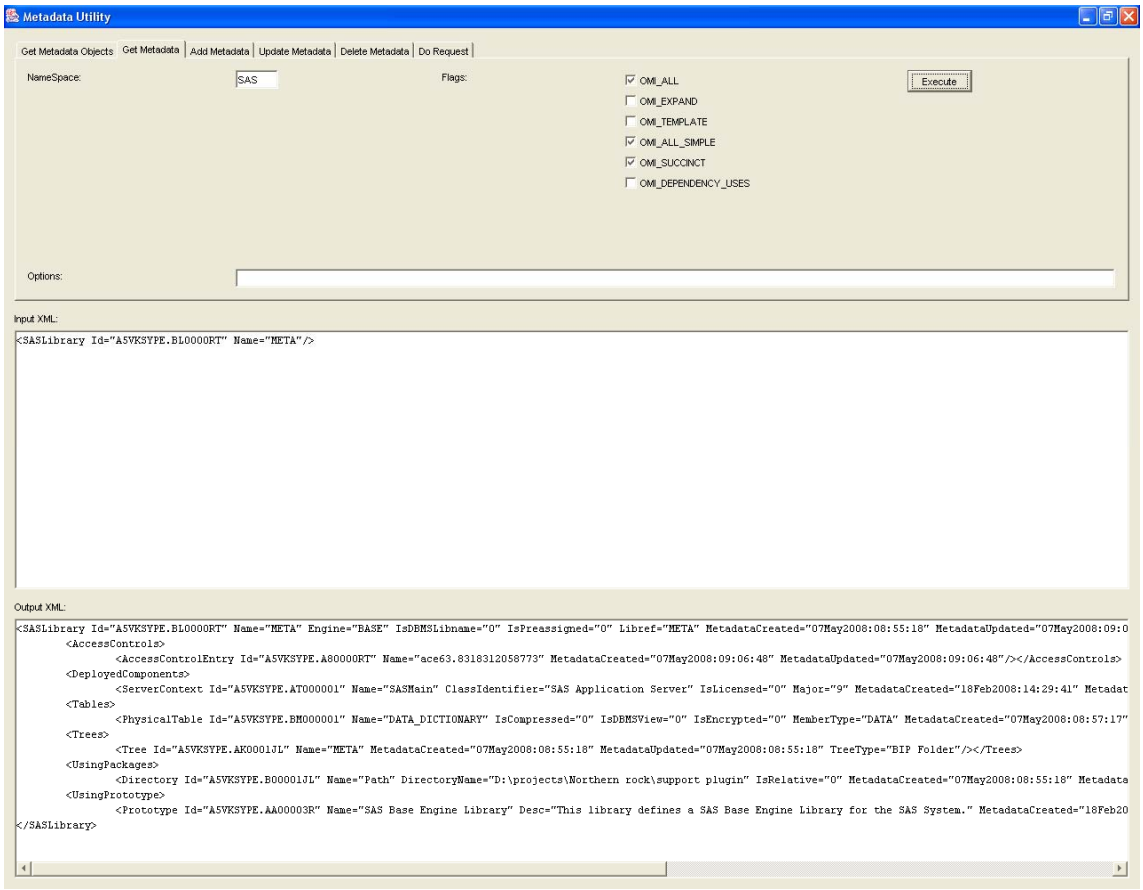
MAKING YOUR WAY AROUND METADATA

In most occasions, the challenge is to know where specific information has been stored in the metadata model. A good method of exploring the existing metadata is using the Metadata Utility tool found within SAS Management Console.

Basic questions such as how many libraries have been defined can easily be answered by the metadata server with this simple tool. Just add the Namespace to be used and the type of object to query for a given repository. Executing the query will return those results on screen.

More information on a specific object (such as properties or associations) is available through the "Get Metadata" tab.





A formatted version of the Output XML panel is included below. Information included under the *SASLibraries* tag refers to library properties. Associations for the specific library are listed in separate tags, like *AccessControls* and *Tables*.


```
<SASLibrary Id="A5VKSYPE.BL0000RT" Name="META" Engine="BASE" IsDBMSLibname="0" IsPreassigned="0" Libref="META"
        MetadataCreated="07May2008:08:55:18" MetadataUpdated="07May2008:09:08:29">
        <AccessControls>
                <AccessControlEntry Id="A5VKSYPE.A80000RT" Name="ace63.8318312058773"
                MetadataCreated="07May2008:09:06:48" MetadataUpdated="07May2008:09:06:48"/></AccessControls>
        <DeployedComponents>
                <ServerContext Id="A5VKSYPE.AT000001" Name="SASMain" ClassIdentifier="SAS Application Server"
                IsLicensed="0" Major="9" MetadataCreated="18Feb2008:14:29:41" MetadataUpdated="18Feb2008:14:29:41"
                Minor="1" ProductName="SAS" SoftwareVersion="9.1" Vendor="SAS Institute"/></DeployedComponents>
        <Tables>
                <PhysicalTable Id="A5VKSYPE.BM000001" Name="DATA_DICTIONARY" IsCompressed="0" IsDBMSView="0"
                IsEncrypted="0" MemberType="DATA" MetadataCreated="07May2008:08:57:17"
                MetadataUpdated="07May2008:08:57:17" NumRows="-1" SASTableName="DATA_DICTIONARY"
                TableName="DATA_DICTIONARY"/></Tables>
        <Trees>
                <Tree Id="A5VKSYPE.AK0001JL" Name="META" MetadataCreated="07May2008:08:55:18"
                MetadataUpdated="07May2008:08:55:18" TreeType="BIP Folder"/></Trees>
        <UsingPackages>
                <Directory Id="A5VKSYPE.B00001JL" Name="Path" DirectoryName="D:\projects\Northern rock\support
                plugin" IsRelative="0" MetadataCreated="07May2008:08:55:18"
                MetadataUpdated="07May2008:08:55:18"/></UsingPackages>
        <UsingPrototype>
                <Prototype Id="A5VKSYPE.AA00003R" Name="SAS Base Engine Library" Desc="This library defines a SAS
                Base Engine Library for the SAS System." MetadataCreated="18Feb2008:14:29:00"
                MetadataType="SASLibrary" MetadataUpdated="18Feb2008:14:29:00"/></UsingPrototype>
</SASLibrary>
```

More information on how to use the Metadata Utility tool and the meaning of selected Flags can be found in Metadata for SAS®9 Programmers [Ref 1].

Properties and associations are easy to read from the Metadata Utility Output XML, but the next section will illustrate how this information can be captured programmatically using data step functions within a normal DMS session.

METADATA DATA STEP FUNCTIONS

Once the metadata information has been located using the Metadata Utility tool, the generated query in Management Console needs to be translated into the DMS environment. The data step environment contains a set of specific functions to query the metadata server. Some of the most useful ones are included below, but for a full list refer to the SAS documentation available.

Here are the most useful functions that can be found when querying the metadata server:

Function Syntax	Description
METADATA_RESOLVE(uri, type, id)	Resolves a metadata URI into a specific object type
METADATA_GETATTR(uri, attr, value)	Returns the named attribute for the object specified by the URI
METADATA_GETNASL(uri, n, asn)	Returns the nth named association for the object URI
METADATA_GETNASN(uri, asn, n, nuri)	Returns the nth associated object of the association specified
METADATA_GETNATR(uri, n, attr, value)	Returns the nth attribute on the object specified by the URI
METADATA_GETNOBJ(uri, n, nuri)	Returns the nth object matching the specified URI
METADATA_GETNPRP(uri, n, prop, value)	Returns the nth property on the object specified by the input URI
METADATA_GETNTYP(n, type)	Returns the nth object type on the server
METADATA_GETPROP(uri, prop, value)	Returns the named property for the object specified by the input URI

The URI or Universal Resource Identifier is the unique object number assigned to each metadata object.

In order to illustrate how these functions work an example has been prepared using the Metadata Utility tool.

Accessing the *Person* object type for a user defined metadata, the following properties and associations are currently defined in the repository:

Output XML:

```
<Person Id="A5VKSYPE.AR0000RT" Name="Elena Muriel" Desc="Elena Muriels user account" MetadataCreated="25Jul2008:10:59:18" MetadataUpdated="18Sep2008:10:14:45">
        <EmailAddresses>
                <Email Id="A5VKSYPE.BQ000001" Name="Work" Address="elena.muriel@amadeus.co.uk" EmailType="Work" MetadataCreated="22Aug2008:16:02:24"
                MetadataUpdated="05Dec2008:12:19:40"/>
        </EmailAddresses>
        <IdentityGroups>
                <IdentityGroup Id="A5VKSYPE.A30001JL" Name="Dept A" MetadataCreated="25Jul2008:11:00:49" MetadataUpdated="25Jul2008:11:00:49"/>
                <IdentityGroup Id="A5VKSYPE.A30004MP" Name="Admins" MetadataCreated="08Oct2008:10:30:42" MetadataUpdated="08Oct2008:10:35:40"/>
                <IdentityGroup Id="A5VKSYPE.A30000RT" Name="Dept B" MetadataCreated="23Jul2008:12:30:33" MetadataUpdated="04Dec2008:15:48:14"/>
        </IdentityGroups>
        <Logins>
                <Login Id="A5VKSYPE.AS0000RT" Name="Login.Elena Muriel.92" ChangeState="ELENALAPTOP\ADMINISTRATOR" MetadataCreated="25Jul2008:10:59:18"
                MetadataUpdated="27Nov2008:14:33:21" Password="********" UserID="elenalaptop\administrator"/>
        </Logins>
</Person>
```

Notice the *EmailAddresses* and *IdentityGroup* associations, which give access to the personal email addresses and the metadata groups assigned to that user.

```
<EmailAddresses>
        <Email Id="A5VKSYPE.BQ000001" Name="Work" Address="elena.muriel@amadeus.co.uk"
        mailType="Work" MetadataCreated="22Aug2008:16:02:24"
        MetadataUpdated="05Dec2008:12:19:40"/>
</EmailAddresses>
```

The information obtained above can also be accessed by using different metadata data step functions.

a) **Reading the description (*Desc*) property from an object.** This can also be used to retrieve any other information stored under the *Person* tag

Metadata Function	Result

`data _null_;`
`  length perDesc $256;`
`  rc=METADATA_GETATTR("omsobj:Person?@Name='Elena Muriel'",`
`  'Desc',perDesc);`
`  put perDesc=;`
`run;`
	`perDesc=Elena Muriels user account`

b**) Finding metadata objects under an association related to the *Person* type.** Once an ID is obtained, information such as email addresses can also be read

Metadata Function	Result

`data _null_;`
`  length nuri $256;`
`  rc=METADATA_GETNASN("omsobj:Person?@Name='Elena Muriel'",`
`  'EmailAddresses',1,nuri);`
`  put nuri=;`
`run;`
	`nuri=OMSOBJ:Email\A5VKSYPE.BQ000001`

c) **Obtaining the ID for a given metadata object**

Metadata Function	Result

`data _null_;`
`  length nobjuri $256;`
`  rc=METADATA_GETNOBJ("omsobj:Person?@Name='Elena Muriel'",`
`  1,nobjuri);`
`  put nobjuri=;`
`run;`
	`nobjuri=OMSOBJ:Person\A5VKSYPE.AR0000RT`

d) **Obtaining a list of all the possible properties for the given *Person* object and values stored**

Metadata Function

```
data _null_;
  length attr metanvalue $256;
  i=0;
  do until(rc<0);
    i+1;
    rc=METADATA_GETNATR("omsobj:Person?@Name='Elena Muriel'",
                         I,attr,metanvalue);
    if rc>0 then put attr= metanvalue=;
  end;
run;
```

Result

attr	metanvalue
Title	
Name	Elena Muriel
MetadataUpdated	18Sep2008:10:14:45
MetadataCreated	25Jul2008:10:59:18
LockedBy	
Desc	Elena Muriels user account
ChangeState	
Id	A5VKSYPE.AROOOORT

e) **Obtaining a list of all the possible associations that the *Person* object can have.**

Metadata Function

```
data _null_;
  length metaasn $256;
  i=0;
  do until(rc<0);
    i+1;
    rc=METADATA_GETNASL("omsobj:Person?@Name='Elena Muriel'",
                         i,metaasn);
    if rc >0 then put metaasn=;
  end;
run;
```

Result

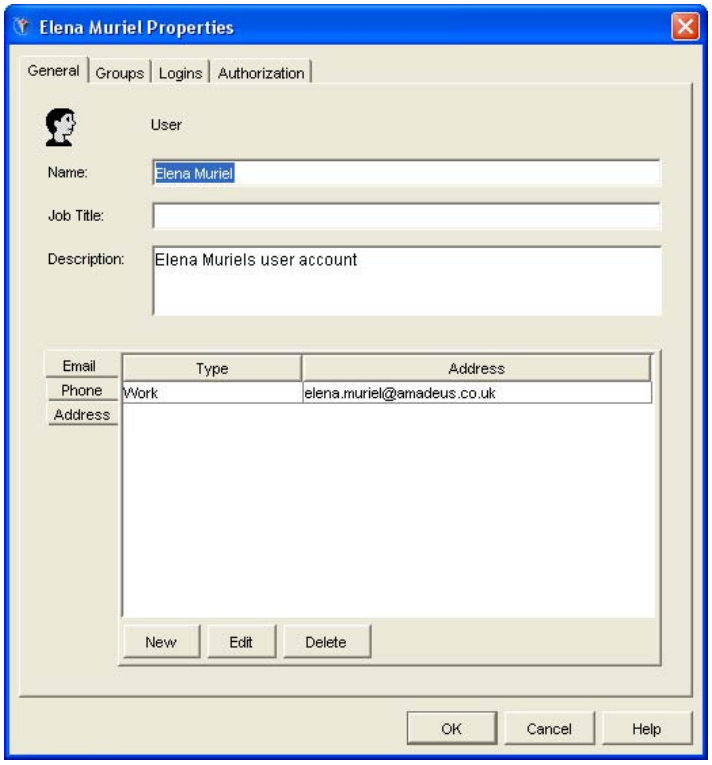
Metaasn
AccessControlEntries
AccessControls
Changes
Documents
EmailAddresses
Extensions
ExternalIdentities
Groups
IdentityChanges
IdentityGroups
Implementors
Keywords
LocalizedAttributes
Locations
Logins
Notes
PhoneNumbers
PrimaryPropertyGroup
Properties
PropertySets
Responsibilities
ResponsibleParties
SourceTransformations
SpecSourceTransformations
SpecTargetTransformations
SubscriberIdentities
TargetTransformations
Timestamps
Trees
UsedByPrototypes
UsingPrototype
Variables
Properties

Once it is understood how to retrieve this basic information, let's have a look at some utility programs.

OBTAINING EMAIL ADDRESSES

A very useful query is to notify the SAS Administrators when there are any errors or problems with the system. A metadata query can be created to dynamically obtain the email addresses of those users which are included in the SAS Administrator metadata group. This way if new administrators are added or removed the notification email programs does not need to be updated.

In order to retrieve email addresses these must have been previously specified in Management Console using the User Manager plug in.



The following program creates a data set containing all the email addresses of the users contained in the Admins metadata user group. This is achievable by using a combination of the METADATA_GETNASN and METADATA_ATTR functions.

metadata.sas

```
*Obtain email address;
%let group=Admins;

data work._emails(keep=email);
  length  uri nameid emailuri email $256;
  i=1;
  group=symget('group');
  do until(rc<0);
    /*Initialise variables*/
    uri='';nameid='';emailuri='';email='';
    /*Obtain object information for given group*/
①   rc=metadata_getnasn("omsobj:IdentityGroup?@Name='"!!group!!"'",
                         "MemberIdentities",i,uri);
②   rc2=metadata_getattr(uri,"Id",nameid);

    /*Obtain email address for users*/
③   rc3=metadata_getnasn("omsobj:Person?@Id='"!! nameid !!"'",
                          "EmailAddresses",1,emailuri);
④   rc4=metadata_getattr(emailuri,"Address",email);
    i+1;
    if rc>=0 and email ne '' then output;
    put _all_;
  end;
run;
```

The first part of the program obtains identifiers for each person included on the Admins metadata group, whilst the second part retrieves the email addresses.

① Queries the given group for all the users that are included. The result is the URI for the persons included on the Admin group, which is obtained by querying the *IdentityGroup* object type and accessing the *MemberIdentities* association.

```
OMSOBJ:Person\A5VKSYPE.AR0000RT
```

② Obtains the identifier part for the given URI.

```
A5VKSYPE.AR0000RT
```

③ Using the *Person* property for the given ID, the program can access the first *EmailAddress* association

```
OMSOBJ:Email\A5VKSYPE.BQ000001
```

If multiple email addresses have been specified in metadata then a decision needs to be made on which one to keep

④ Reads the email address property for the given address identifier.

```
elena.muriel@amadeus.co.uk
```

The following shows the equivalent results obtained from the Metadata Utility tool when querying the *IdentityGroup* object type for the Admins group.

Output XML:

```
<IdentityGroup Id="A5VKSYPE.A30004MP" Name="Admins" MetadataCreated="08Oct2008:10:30:42" MetadataUpdated="08Oct2008:10:35:40">
        <AccessControlEntries>
                <AccessControlEntry Id="A5VKSYPE.A8000ASX" Name="GRANT Admins access" MetadataCreated="14Nov2008:11:56:17" MetadataUpdated="14Nov2008:11:56:17"/></Access
        <MemberIdentities>
                <Person Id="A5VKSYPE.AR0000RT" Name="Elena Muriel" Desc="Elena Muriels user account" MetadataCreated="25Jul2008:10:59:18" MetadataUpdated="18Sep2008:10:1
                <Person Id="A5VKSYPE.AR000001" Name="SAS Administrator" MetadataCreated="18Feb2008:14:29:19" MetadataUpdated="18Feb2008:14:29:19"/></MemberIdentities>
        <UsedByPrototypes>
                <Tree Id="A5VKSYPE.AK00099D" Name="Admins Permissions Tree" MetadataCreated="14Nov2008:11:56:17" MetadataUpdated="14Nov2008:11:56:17" TreeType=" Permissi
</IdentityGroup>
```

And the *Person* object type to obtain the email address

Output XML:

```
<Person Id="A5VKSYPE.AR0000RT" Name="Elena Muriel" Desc="Elena Muriels user account" MetadataCreated="25Jul2008:10:59:18" MetadataUpdated="1
        <AccessControlEntries>
                <AccessControlEntry Id="A5VKSYPE.A8000CC0" Name="GRANT Elena Muriel access" MetadataCreated="18Nov2008:11:39:53" MetadataUpd
        <EmailAddresses>
                <Email Id="A5VKSYPE.BQ000001" Name="Work" Address="elena.muriel@amadeus.co.uk" EmailType="Work" MetadataCreated="22Aug2008:1
        <IdentityGroups>
                <IdentityGroup Id="A5VKSYPE.A30001JL" Name="Dept A" MetadataCreated="25Jul2008:11:00:49" MetadataUpdated="25Jul2008:11:00:49
                <IdentityGroup Id="A5VKSYPE.A3000336" Name="Media Manager Administrators" MetadataCreated="05Aug2008:13:07:40" MetadataUpdat
                <IdentityGroup Id="A5VKSYPE.A30003UX" Name="Media Manager Financial Users" MetadataCreated="27Aug2008:14:56:00" MetadataUpda
```

VERIFYING GROUP MEMBERSHIP

The second utility program included shows how to use metadata functions to automatically assign autoexec files when users from different departments start SAS sessions. This is extremely useful when using Enterprise Guide in an environment where each department needs to access different formats catalogs. It can also be used to perform specific departmental processing when using the same logical workspace server.

Next example checks if the current user belongs to Dept B defined in metadata.

metadata.sas

```
   *Check group in just one step;
   %let group1=Dept B;

 data _Verify(keep=rc);
    length  loginuri dataname type $256 name $30 id $20;
    loginuri="";dataname="";type="";id="";
    getpersonid=symget('sysuserid');
① rc=metadata_getnobj("omsobj:Login?@UserId contains '"||getpersonid||"'",
                        1,loginuri);
② rc2=metadata_getattr(loginuri,"Name",dataname);
③ name=scan(dataname,2,'.');
   /* Build URI for the group with a name of &group1 and person name*/
④ obj="omsobj:IdentityGroup?IdentityGroup[@Name='"||"&group1"||
         "'][MemberIdentities/Person[@Name='"||name||"']]";
⑤ rc=metadata_resolve(obj,type,id);
   put rc= type= id=;
   label rc="&group1";
  run;
```

① The Login object type is used to obtain the URI of the person launching the session

② From the URI the metadata login name is obtained

③ The login credentials stored have the format of

```
<Login Id="A5VKSYPE.AS0000RT" Name="Login.Elena Muriel.92"/>
```

Thereafter the SCAN function has been used to retrieve the person user name, which is then used by the second data step.

④ Build the query search by concatenating two search criteria. This allows navigation through the *IdentityGroup* and *MemberIdentities* associations.

⑤ The METADATA_RESOLVE function is used to check if a person with those properties is listed under the Dept B metadata group. If the function is successful, the return code RC will be set to 1 and the person belongs to the group.

Below are the returns of the query using the Metadata Utility when displaying the associations and properties of the Dept B metadata group.

Output XML:

```
<IdentityGroup Id="A5VKSYPE.A30000RT" Name="Dept B" MetadataCreated="23Jul2008:12:30:33" MetadataUpdated="04Dec2008:15:48:14">
        <AccessControlEntries>
                <AccessControlEntry Id="A5VKSYPE.A80002BD" Name="GRANT Dept B access" MetadataCreated="24Jul2008:14:47:25" MetadataUpdated="24Jul2008:14:47:25"/></AccessCont
        <IdentityGroups>
                <IdentityGroup Id="A5VKSYPE.A30005EH" Name="conneciton to oracle" MetadataCreated="17Nov2008:13:23:53" MetadataUpdated="20Nov2008:15:13:47"/></IdentityGroups
        <MemberIdentities>
                <Person Id="A5VKSYPE.AR0000RT" Name="Elena Muriel" Desc="Elena Muriels user account" MetadataCreated="25Jul2008:10:59:18" MetadataUpdated="18Sep2008:10:14:45
        <UsedByProcesses>
                <Tree Id="A5VKSYPE.AK000335" Name="Dept B Permissions Tree" MetadataCreated="24Jul2008:14:47:25" MetadataUpdated="24Jul2008:14:47:25" TreeType=" Permissions
</IdentityGroup>
```

This program can easily be converted into a macro that gets executed by the generic ***Appserver_autoexec.sas*** program. Once identified the group a user belongs to, **%INCLUDE** statements can be used to add any department custom specific code.

CONCLUSION

Although initially it can be a little daunting, the use of metadata data step functions allows developers to take the driving seat when interacting with the metadata server. The use of these functions gives greater flexibility when it comes to interacting and understanding the information stored in repositories.

REFERENCES

[1] Metadata for SAS®9 Programmers, Elena Muriel & Paul Simkin
[2] SAS 9.1 Open Metadata Interface Reference
[3] SAS 9.1 Open Metadata Interface: User's Guide
[4] SAS 9.1 Help

CONTACT INFORMATION

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

Author Names	Elena Muriel
Company:	Amadeus Software Limited
Address:	Orchard Farm, Witney Lane, Leafield, Oxon OX29 9PG
Phone:	01993 878287
Fax:	01993 878042
Email:	elena.muriel@amadeus.co.uk
Web:	www.amadeus.co.uk