Open Metadata Interface

Clubs Canadiens automne 2013

Ordre du jour

- Introduction
- La SAS Management Console: outil de découverte des métadonnées
- Manipulation des métadonnées
  - Exemple: la définition des utilisateurs
- Open Metadata Interface, ou comment manipuler les métadonnées par programmation
- Conclusion
- Ressources

Introduction

La plate-forme SAS BI est architecturée autour des métadonnées

- D'où la nécessité d'administrer ces métadonnées:
  - Via la SAS Management Console
  - Par programmation
  - Dans DMS: commandes METABROWSE, METACON, et METAFIND
- Connaissance du modèle
- Puis manipulation des métadonnées via Open Metadata Interface: API utilisable dans un programme SAS (ou java).

Cette présentation va s'attacher, de manière très pratique, à vous montrer comment il est possible d'automatiser intégralement la gestion des métadonnées au sein de la plate-forme SAS9.

Le modèle des métadonnées

La documentation

Dans la SMC avec l'utilitaire de métadonnées

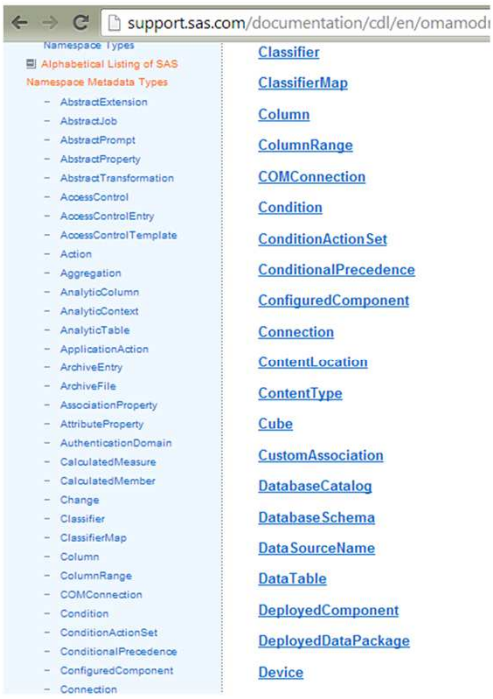
- Lister les objets d'un type particulier
- Obtenir l'ensemble des métadonnées associées à un objet
- Supprimer un objet

What is the SAS Metadata Model?
The SAS Metadata Model defines the metadata types that SAS metadata APIs use to create metadata objects. It is an object-oriented, hierarchical model.
It provides objects and classes that define metadata repositories, the SAS Repository Manager, and different types of application metadata.
It defines associations between related objects.
It uses inheritance of attributes and associations to affect common behaviors.
It uses subclassing to extend behaviors.
The SAS Metadata Model provides metadata types in the REPOS namespace and SAS namespace.
The REPOS namespace metadata types define metadata repositories and the SAS Repository Manager. The SAS Metadata Server uses information that clients store in repository objects to access metadata repositories. It uses the SAS Repository Manager to manage the metadata repositories.
The SAS namespace defines metadata types that describe application elements. Clients use the metadata types in various combinations to create metadata that describes application data and entities used by an application. Clients use this metadata to manage the entities that they describe. For example, a client might use the metadata that defines a stored process, cube, or job to manage the stored process, cube, or job. A client might export the metadata for a data store so that the data store can be accessed by more than one application.

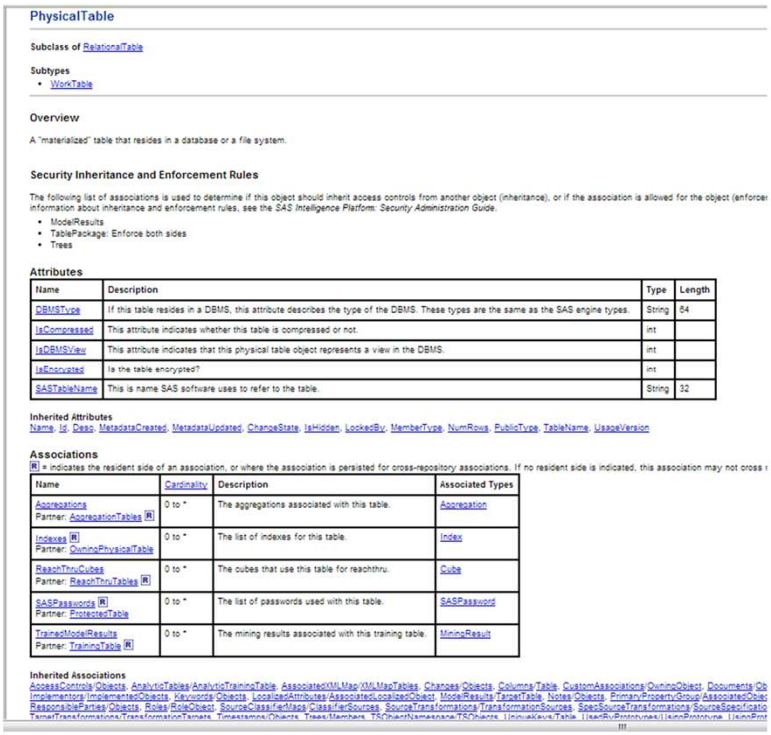
Documentation

- Liste alphabétique des différents Types d'objets

SAS® 9.3 Metadata Model: Reference



Documentation



PhysicalTable

Subclass of RelationalTable

Subtypes
- WorkTable

Overview

A "materialized" table that resides in a database or a file system.

Security Inheritance and Enforcement Rules

The following list of associations is used to determine if this object should inherit access controls from another object (inheritance), or if the association is allowed for the object (enforce
information about inheritance and enforcement rules, see the *SAS Intelligence Platform: Security Administration Guide*.

- ModelResults
- TablePackage: Enforce both sides
- Trees

Attributes

Name	Description	Type	Length
DBMSType	If this table resides in a DBMS, this attribute describes the type of the DBMS. These types are the same as the SAS engine types.	String	64
IsCompressed	This attribute indicates whether this table is compressed or not.	int	
IsDBMSView	This attribute indicates that this physical table object represents a view in the DBMS.	int	
IsEncrypted	Is the table encrypted?	int	
SASTableName	This is name SAS software uses to refer to the table.	String	32

Inherited Attributes
Name, Id, Desc, MetadataCreated, MetadataUpdated, ChangeState, IsHidden, LockedBy, MemberType, NumRows, PublicType, TableName, UsageVersion

Associations

R = indicates the resident side of an association, or where the association is persisted for cross-repository associations. If no resident side is indicated, this association may not cross

Name	Cardinality	Description	Associated Types
Aggregations Partner: AggregationTables R	0 to *	The aggregations associated with this table.	Aggregation
Indexes R Partner: OwningPhysicalTable	0 to *	The list of indexes for this table.	Index
ReachThruCubes Partner: ReachThruTables R	0 to *	The cubes that use this table for reachthru.	Cube
SASPasswords R Partner: ProtectedTable	0 to *	The list of passwords used with this table.	SASPassword
TrainedModelResults Partner: TrainingTable R	0 to *	The mining results associated with this training table.	MiningResult

Inherited Associations
AccessControls/Objects, AnalyticTables/AnalyticTrainingTable, AssociatedXMLMap/XMLMapTables, Changes/Objects, Columns/Table, CustomAssociations/OwningObject, Documents/Ob
Implementors/ImplementedObjects, Keywords/Objects, LocalizedAttributes/AssociatedLocalizedObject, ModelResults/TargetTable, Notes/Objects, PrimaryPropertyGroup/AssociatedObjec
ResponsibleParties/Objects, Roles/RoleObject, SourceClassifierMaps/ClassifierSources, SourceTransformations/TransformationSources, SpecSourceTransformations/SourceSpecificatio

Ordre du jour

- Introduction
- La SAS Management Console: outil de découverte des métadonnées
- Manipulation des métadonnées
  - Exemple: la définition des utilisateurs
- Open Metadata Interface, ou comment manipuler les métadonnées par programmation
- Conclusion
- Ressources

La SAS Management Console

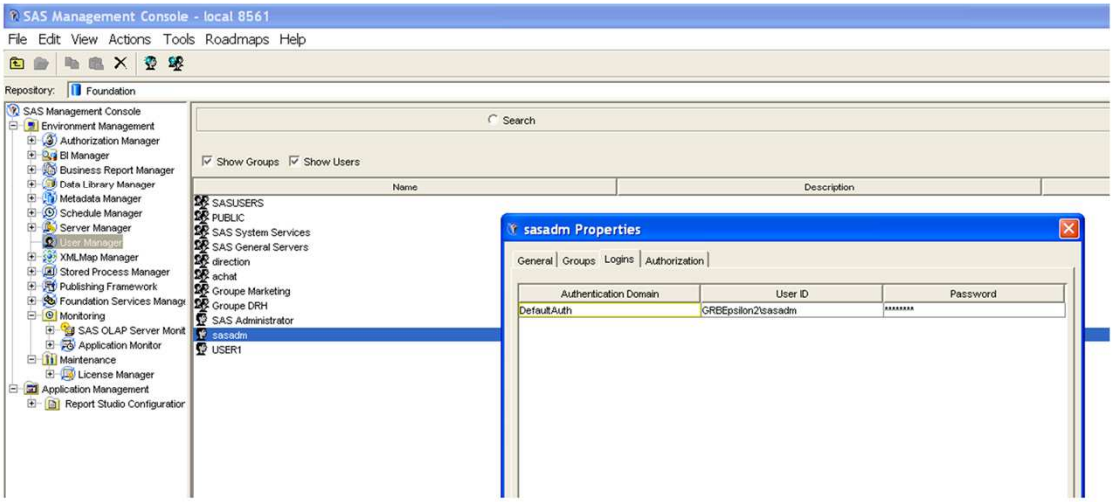
Affiche des vues des objets stockés dans les métadonnées

- Et de leurs attributs
- Et de leurs relations avec d'autres objets…

L'utilitaire de métadonnées permet de récupérer l'XML descriptifs de ces objets, attributs et relations.

Dans notre **S**AS **M**anagement **C**onsole vous voyez des utilisateurs : En fait un utilisateur est représenté dans les métadonnées par un **objet** de type « **PERSON** ».

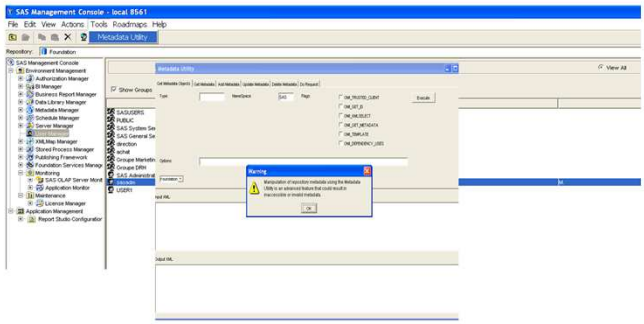
La SAS Management Console

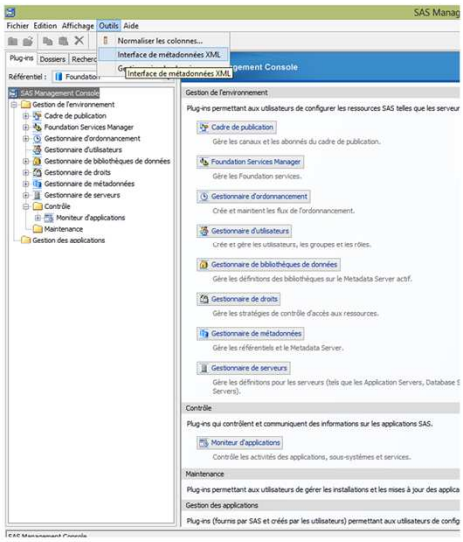


Toujours concernant un utilisateur, et pour illustrer notre propos, cet utilisateur possède un ou plusieurs **login** de connexion qui sont, eux aussi, représentés dans les métadonnées par des **objets** (de type « **login** »).

Enfin un dernier exemple concernant cette fois les données SAS. Une bibliothèque (**librairie**) SAS vue depuis **l'Add-in Excel** est elle aussi un objet (type :**SASLibrary**). Mais cela ne s'arrête pas là, chaque table est un objet (type :**sasTable**) mais aussi chacune des variables de chaque table (type :**Column**)

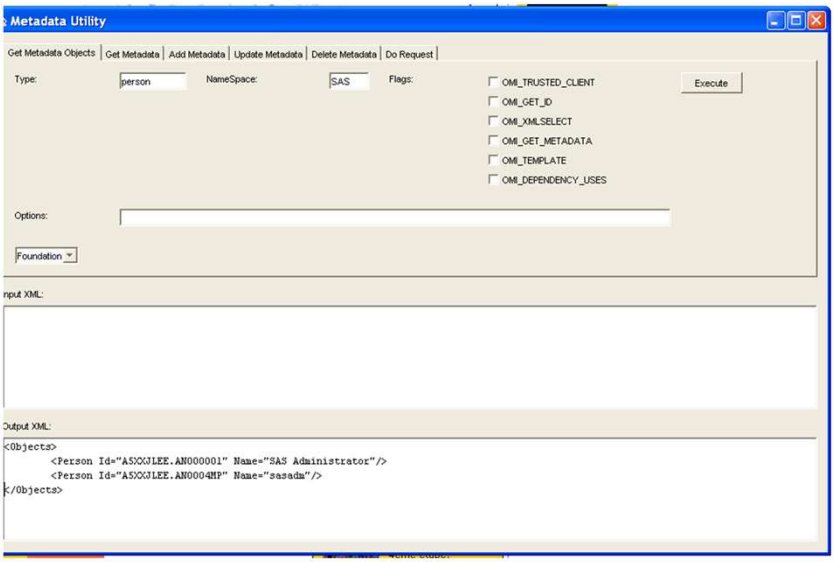
Metadata Utility



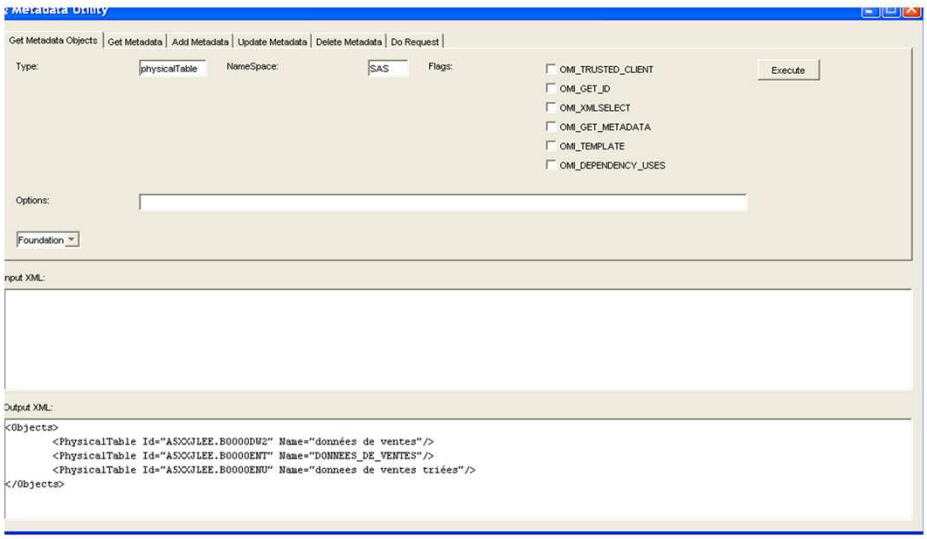


C’est l’utilitaire de métadonnées qui va vous permettre d’avoir une vision plus technique des métadonnées. En effet, ce dernier va être capable de vous fournir un état au format XML pour chaque objet. Dans cet état vous retrouverez facilement attributs et associations liées à cet objet.

Metadata Utility



Metadata Utility

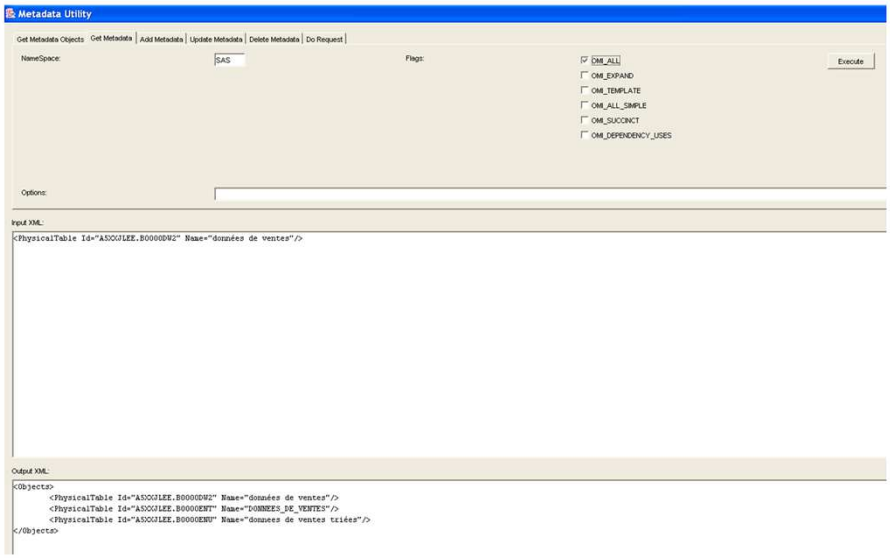


Pour obtenir la liste de tous les objets de type « Person », autrement dit la liste des utilisateurs définis dans les métadonnées, il vous suffit de saisir « Person » dans le champ « type » (cf ci-dessous) puis « Exécuter »

PhysicalTable nous affichera la liste des tables définies dans les métadonnées

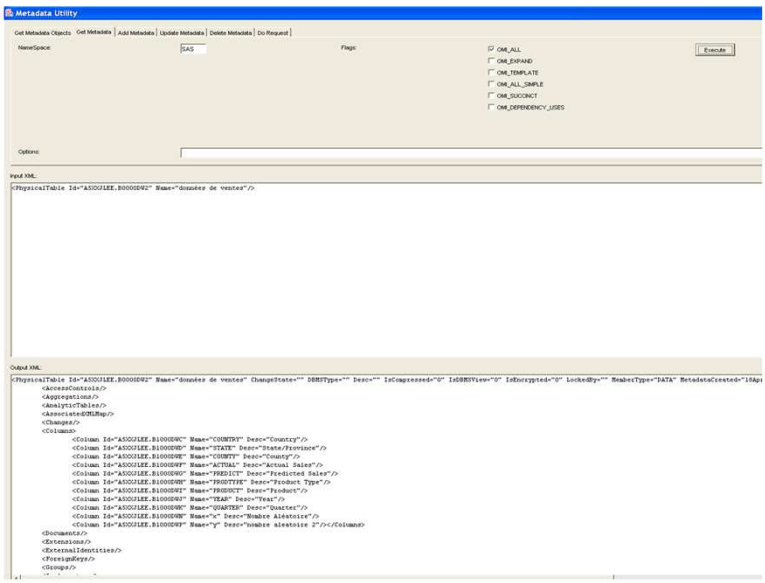
Nous n'avons ici que la liste des objets avec deux de leurs attributs :
L'identifiant de l'objet →ID
Le nom de l'objet →Name

Metadata Utility



Nous ne voyons ni tous les attributs ni les associations de ces derniers. L'onglet « Obtenir les métadonnées » va nous permettre d'accéder à toutes ces informations.

Metadata Utility



Metadata Utility

- Les attributs

```
<PhysicalTable Id="A5XXJLEE.B0000DW2«
     Name="données de ventes"
     ChangeState="" DBMSType="" Desc="«
     IsCompressed="0" IsDBMSView="0«
     IsEncrypted="0" LockedBy="" MemberType="DATA«
     MetadataCreated="18Apr2006:17:43:03«
     MetadataUpdated="18Apr2006:19:16:52«
     NumRows="-1" SASTableName="donnees_de_ventes«
     TableName="">
```

Metadata Utility

- Les associations

```
<AccessControls/>
<Aggregations/>
<AnalyticTables/>
<AssociatedXMLMap/>
<Changes/>
<Columns>
        <Column Id="A5XXJLEE.B1000DWC" Name="COUNTRY" Desc="Country"/>
        <Column Id="A5XXJLEE.B1000DWD" Name="STATE" Desc="State/Province"/>
        <Column Id="A5XXJLEE.B1000DWE" Name="COUNTY" Desc="County"/>
        <Column Id="A5XXJLEE.B1000DWF" Name="ACTUAL" Desc="Actual Sales"/>
        <Column Id="A5XXJLEE.B1000DWG" Name="PREDICT" Desc="Predicted Sales"/>
        <Column Id="A5XXJLEE.B1000DWH" Name="PRODTYPE" Desc="Product Type"/>
        <Column Id="A5XXJLEE.B1000DWI" Name="PRODUCT" Desc="Product"/>
        <Column Id="A5XXJLEE.B1000DWJ" Name="YEAR" Desc="Year"/>
        <Column Id="A5XXJLEE.B1000DWK" Name="QUARTER" Desc="Quarter"/>
<Columns>
```

Ordre du jour

- Introduction
- La SAS Management Console: outil de découverte des métadonnées
- Manipulation des métadonnées
  - Exemple: la définition des utilisateurs
- Open Metadata Interface, ou comment manipuler les métadonnées par programmation
- Conclusion
- Ressources

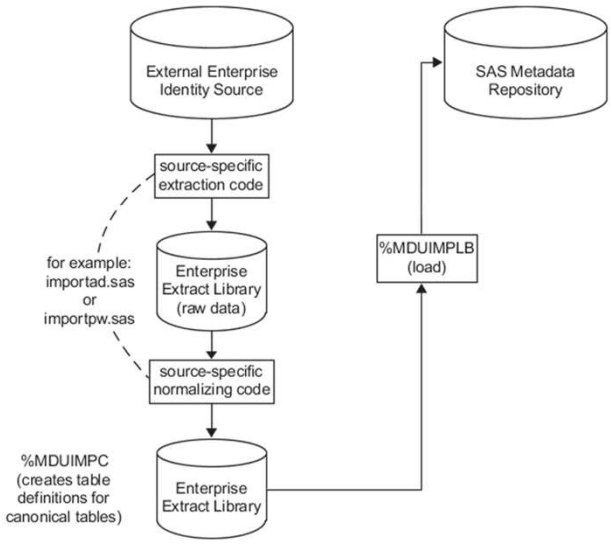
Manipuler les métadonnées via programme

Premier exemple: gestion des utilisateurs

- Définition des utilisateurs: Person et Group
- Les macros SAS:
  1. Extraction des tables canoniques cible : **%mduEXTR**
  2. Extraction des tables canoniques maitre : **%mduEXTR**
  3. Alimentation des tables cible : vos programme SAS
  4. Comparaison cible/maître : **%mduCMP**
  5. Validation de la cohérence des modification : **%mduCHGV**
  6. Chargement **:%mduCHGL**

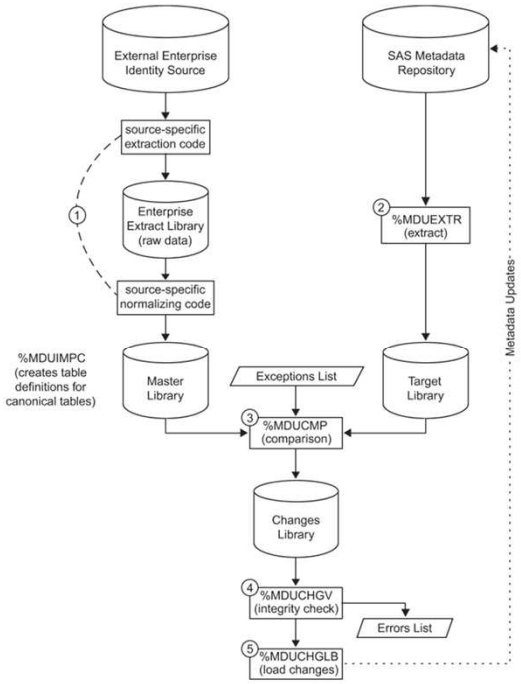
Gestion des utilisateurs

Importation (Bulk Load)



Gestion des utilisateurs

Synchronisation périodique



%mduEXTR

Utilisation : Permet d'extraire les informations contenues dans les méta-données et de les stocker dans des tables SAS.

Syntaxe :

```
%mduEXTR(libref=LibraryName) ;
```

%mduCMP

Utilisation : Permet de comparer les tables canoniques de la librairie cible/Target à celle de la librairie MASTER. Alimente la librairie CHANGES qui contiendra les modifications à charger dans les méta-données.

Syntaxe :

```
%mduCMP (   master = LibraryName,
            target  =  LibraryName,
            change = LibraryName,
            exceptions=LibraryName.SASDataSet,
            externonly=0|1,
            authdomcompare=name|keyid) ;
```

master : Librarie SAS qui contient les tables canoniques mises à jour
target : Librairie SAS qui contient les tables canoniques non mises à jour (image des métadonnées avant modification)
change : Librairie SAS qui contiendra les mises à jour
exceptions : table SAS qui contient les exceptions à respecter pendant la comparaison des librairies MASTER et TARGET.
Externonly = 0 indique que tous les utilisateurs et groupes sont comparés (ceux entrés par le biais de la SAS Management Console ainsi que ceux entrés par programme). 1 indique que ceux saisis dans la SAS Management Console sont exclus de cette comparaison.
Authdomcompare : mode de comparaison des informations concernant le domaine d'authentification. Les valeurs possibles sont **name** et **keyid**.

%mduCHGV

Utilisation : Permet de valider que les modifications identifiés par la macro **%mduCMP** et présentes dans la librairie **CHANGES** respectent les contraintes d'intégrités référentielles définies au niveau des tables canoniques (non duplicité d'un utilisateur, pas d'ajout d'un utilisateur dans un groupe inexistant, etc.)

Syntaxe :

```
%mduCHGV (  target  = LibraryName,
            change = LibraryName,
            temp    = LibraryName,
            errorsds = LibraryName.SASDataSet ) ;
```

target : Librairie SAS qui contient les tables canoniques non mises à jour (image des méta-données avant modification)

change : Librairie SAS qui contient les mises à jour
temp : Librairie SAS de travail temporaire
errorsds : Table SAS contenant les erreurs éventuelles

%mduCHGL

Utilisation : Permet de charger les mises à jours validées vers le serveur de métadonnées SAS.

Syntaxe :

```
%mduCHGL (      change= LibraryName,
                temp   = LibraryName,
                outrequest = Filename,
                outresponse= Filename,
                submit=1|0 ) ;
```

change : Librairie SAS qui contient les mises à jour
temp : Librairie SAS de travail temporaire
outrequest : **filename** vers un fichier XML qui contiendra les mises à jours issues de la librairie CHANGES. Ces mises à jours au format XML seront envoyées au serveur de méta-données via une **procedure metadata**.
Outresponse : **filename** vers un fichier XML qui contiendra le résultat de la **procédure metadata**.
Submit : 1 pour soumettre, 0 pour faire un test et ne pas charger

Ordre du jour

- Introduction
- La SAS Management Console: outil de découverte des métadonnées
- Manipulation des métadonnées
  - Exemple: la définition des utilisateurs
- Open Metadata Interface, ou comment manipuler les métadonnées par programmation
- Conclusion
- Ressources

L'architecture ouverte SAS9 permet de modifier les métadonnées depuis différents environnement : Java, C++ etc. Nous avons choisi de le faire depuis SAS pour plusieurs raisons. La principale étant que les utilisateurs de SAS9 sont très souvent d'anciens utilisateurs SAS et que de ce fait le langage SAS leur est déjà familier.

Open Metadata Interface: créer un objet

```
data _null_ ;
        length uriPerson $256 ;
        rc=metadata_newObj("Person",uriPerson,"USER1");
        put uriPerson= ;
        put rc= ;
run ;
```

La fonction **metadata_newObj** permet de créer un objet.

```
        rc=metadata_newObj("Person"❶,uriPerson❷,"USER1"❸);
```

Nous allons prendre l'exemple de la création d'un utilisateur.

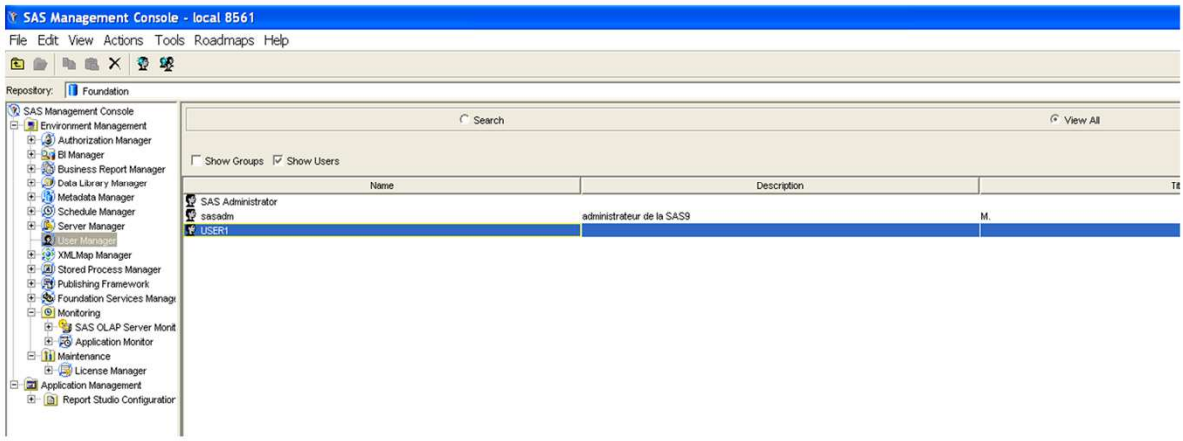
❶ : Type de l'objet à créer
❷ : Variable qui contiendra l'identifiant au sens métadonnées SAS
❸ : Nom de l'objet (valeur de son attribut NAME)

Open Metadata Interface

```
38   data _null_ ;
39      length uriPerson $256 ;
40      rc=metadata_newObj("Person",uriPerson,"USER1");
41      put uriPerson= ;
42      put rc= ;
43   run ;
NOTE: Variable uriPerson is uninitialized.
uriPerson=OMSOBJ:Person\A5XXJLEE.AN0008HN ❶
rc=0
NOTE: DATA statement used (Total process time):
    real time          0.05 seconds
```

❶ Remarquons ici la forme de l'identifiant. Cette identifiant (aussi appelé **URI**), est « ***la plaque d'immatriculation*** » de notre objet dans les méta-données.

Open Metadata Interface



Open Metadata Interface: modifier un attribut

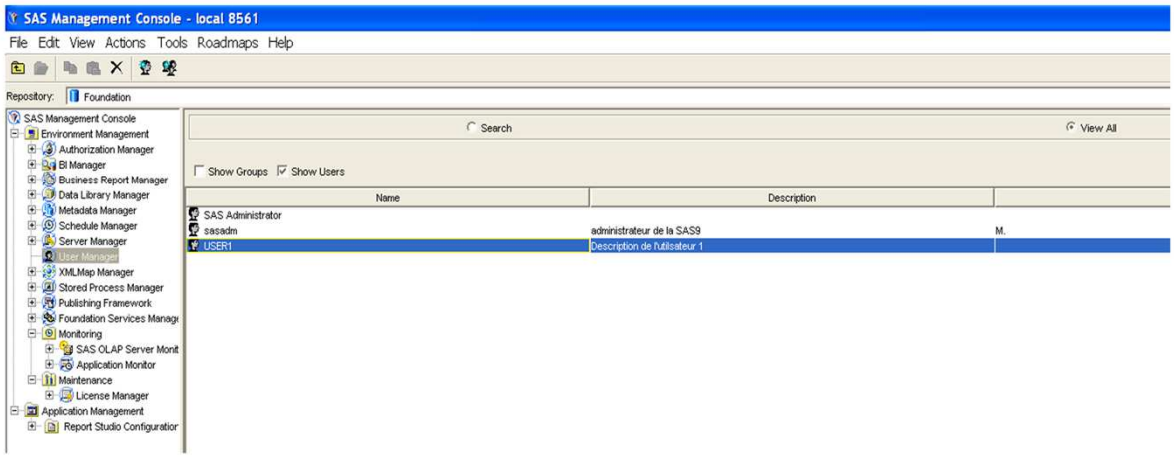
```
data _null_ ;
rc=metadata_setAttr("A5XXJLEE.AN0008HN"❶
                    ,"desc"❷
                    ,"Description de l'utilisateur 1"❸);
put rc= ;
run ;

rc=metadata_setAttr("A5XXJLEE.AN0008HN"❶,"desc"❷,"Description
de l'utilisateur 1"❸);
```

l'utilisateur **USER1** n'a pas de description. En parcourant le documentation ou le XML en sortie détaillant l'objet on constate que les objets de type « **PERSON** » ont un attribut « **Desc** ».

❶ **URI** (Identifiant) de l'objet concerné. Par souci de simplicité nous avons utilisé ici l'URI de l'objet. En réalité, il conviendra de récupérer *dynamiquement* cet URI. Nous aborderons cela un peu plus tard dans ce document.
❷ Nom de l'attribut.
❸ Valeur de l'attribut.

Open Metadata Interface



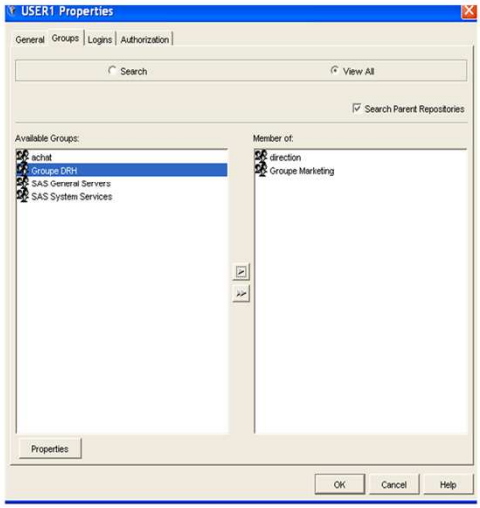
Open Metadata Interface: ajouter une association

```
data _null_ ;
        rc=metadata_setAssn("A5XXJLEE.AN0008HN"❶,
                "IdentityGroups"❷,
                "append"❸,
                "A5XXJLEE.A30001JL"❹);
        put rc= ;
run ;
```

- ❶ URI de l'objet utilisateur
- ❷ Nom de l'association
- ❸Type d'intervention au niveau de l'association, ici on ajoute un objet de type **IdentityGroup**. Autre choix possible : **remove**
- ❹ URI de l'objet **group (Le groupe direction)** à ajouter au niveau de l'association.

Si l'on souhaite associer notre utilisateur à un groupe il va falloir ajouter un objet de type **IdentityGroup** dans l'association « **IdentityGroups** ».

Open Metadata Interface



Open Metadata Interface: récupérer URI

```
data _null_ ;
        length  uriBibliotheque $256;
        nobj=1;
        i=1 ;
        do while (nobj > 0 );
                nobj=metadata_getnobj❶
                ("omsobj:sasLibrary?@Name ?'données'"❷
                , i❸
                , uriBibliotheque❹);
        put uriBibliotheque=;
        i+1;
        end ;
run ;
```

Nous allons voir maintenant comment récupérer l'identifiant (**URI**) d'un objet afin de pouvoir manipuler ce dernier sans connaître cet identifiant.

L'idée est simple, plutôt que de passer un URI en paramètre nous allons passer une requête qui se chargera de trouver l'URI.

❶ Nous utilisons ici la fonction **metadata_getNobj** permettant d'obtenir le Nième objet répondant au critère (requête) passé comme premier argument.

❷ La requête de la forme
omsobj :typeobj ?@nomAttribut operateur 'valeur attribut'
Notre exemple : le Nième objet de type « **sasLibrary** » dont l'attribut « NAME » contient le mot '**données'**.

❸ Indice de l'objet , ici nous prenons successivement tous les objets **tant que** le code retour (**nobj**) est >0. Dans notre exemple il y a 3 objets correspondant à la requête, la fonction ne cherchera pas à retourner l'URI du quatrième (quand i vaudra 4) car **nobj** sera alors inférieur à 0.

Open Metadata Interface

```
113  data _null_ ;
114     length rc uriBibliotheque $256;
115     nobj=1;
116     i=1 ;
117     do while (nobj > 0 );
118        nobj=metadata_getnobj("omsobj:sasLibrary?@Name ? 'données'",i,uriBibliotheque);
119        put uriBibliotheque=;
120        i+1;
121     end ;
122  run ;

NOTE: Variable rc is uninitialized.
NOTE: Variable uriBibliotheque is uninitialized.
uriBibliotheque=OMSOBJ:SASLibrary\A5XXJLEE.AY000A15
uriBibliotheque=OMSOBJ:SASLibrary\A5XXJLEE.AY00099D
uriBibliotheque=OMSOBJ:SASLibrary\A5XXJLEE.AY0008HM
uriBibliotheque=OMSOBJ:SASLibrary\A5XXJLEE.AY0008HM
NOTE: DATA statement used (Total process time):
      real time           0.19 seconds
```

Open Metadata Interface: requête attributs

```
data _null_ ;
        length  uriOBJ userDesc $256;
        nobj=metadata_getnobj
                ("omsobj:person?@Name = 'USER1'"
                ,1
                ,uriOBJ) ❶;
        put nobj = ;
        if nobj=1 then do ;
                rc = metadata_getattr(uriOBJ,"Desc",userDesc) ❷;
                put userDesc= ;
        end;
run ;
```

❶ Maintenant que nous savons récupérer **l'URI** d'un objet il est plus facile (et plus maintenable) de travailler avec la **variable contenant l'URI** (ici **uriOBJ**) plutôt que l'URI explicité en clair. **("A5XXJLEE.AN0008HN")**

❷ Utilisation de la méthode **metadata_getAttr** permettant d'obtenir la valeur d'un attribut. Le premier argument est **l'objet sur lequel vous souhaitez travailler**, le second est **l'attribut** que vous souhaitez connaître et enfin le troisième est **le nom de la variable qui contiendra la valeur de l'attribut**.

Open Metadata Interface

```
79   data _null_ ;
80      length  uriOBJ userDesc $256;
81      nobj=metadata_getnobj ("omsobj:person?@Name = 'USER1'",1,uriOBJ);
82      put nobj = ;
83      if nobj=1 then do ;
84         rc = metadata_getattr(uriOBJ,"Desc",userDesc);
85         put userDesc= ;
86      end;
87   run ;

NOTE: Variable uriOBJ is uninitialized.
NOTE: Variable userDesc is uninitialized.
nobj=1
userDesc=Description de l'utilisateur 1
NOTE: DATA statement used (Total process time):
      real time          0.12 seconds
      cpu time           0.04 seconds
```

Open Metadata Interface: requête association

```
data _null_ ;
        length uriOBJ assoName associatedObjName $256 ;
        nobj= metadata_getnobj
                ("omsobj:login?@Name ? 'Login.utilisateur marketing'"
                ,1
                ,uriOBJ);
        put nobj = ;
        put uriObj= ;
        indiceAsso = 1 ;
        rc = 1 ;

        indiceObjAsso=1 ;
        rc2=1 ;
```

De la même manière qu'il est possible de « scanner » les différents attributs d'un objet il vous sera possible de parcourir les nombreuses associations de ce dernier. Par exemple pour vérifier qu'un objet que vous souhaitez effacer n'est pas lié à un autre objet.

Pour illustrer cela nous allons dans un premier temps parcourir toutes les associations de l'objet de type « login » (**metadata_getNASL**) puis tous les objets de chacune de ces associations (**metadata_getNasN**).

Open Metadata Interface

```
do while (rc>0) ❶ ;
        rc = METADATA_GETNASL(uriOBJ, indiceAsso, assoName);
        put assoName= ;
        do while (rc2 >0 and indiceObjAsso <= rc2) ❷ ;
                rc2 = METADATA_GETNASN(uriOBJ
                        , assoName
                        , indiceObjAsso
                        , associatedObjName);
                put "       "               associatedObjName= ;

                indiceObjAsso+1 ;
        end;
        indiceAsso=indiceAsso+1 ;
        rc2=1 ;
        indiceObjAsso=1 ;
        associatedObjName="";
end;
run ;
```

❶ Dans cette première boucle nous allons parcourir une à une toutes les associations. Pour ce faire nous allons utiliser la fonction **metadata_getNASL** .
argument 1 : objet dont on veut parcourir les associations
argument 2 : indice de l'association
argument 3 : retourne le nom de la Nième association (indice)

❷ Pour chaque association nous parcourons la liste des objets présents dans cette dernière grâce à la fonction **metadata_getNASN.**
Argument 1 : Objet dont in veut parcourir les associations
Argument 2 : Nom de l'association en cours
Argument 3 : Indice de l'objet dans l'association en cours
Argument 4 : Retourne l'URI du Nième objet de l'association en cours

Open Metadata Interface: supprimer association

```
data _null_ ;
length uriOBJ $256;
        nobj= metadata_getnobj
   ("omsobj:person?@Name = 'USER1'"
        ,1
        ,uriOBJ);
         rc = metadata_delassn(uriOBJ,"IdentityGroups") ;
         put rc= ;
run ;
```

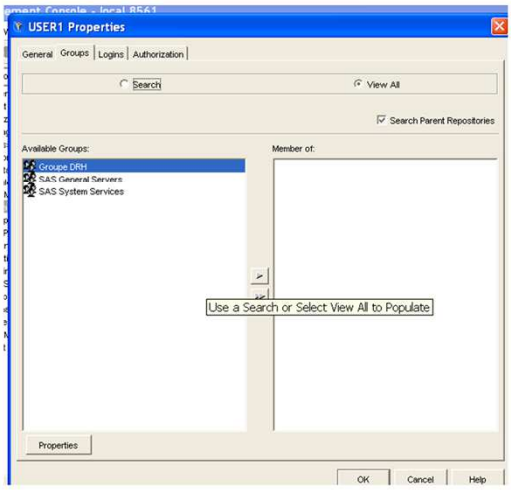
Dans bien des cas vous aurez besoin de « séparer » des objets. Dans la suite logique de nos exemples, vous voudrez enlever un utilisateur des groupes auxquels il appartient. La fonction **metadata_delASSN** permet d'effacer non pas une association mais tous les objets présents au sein de celle ci.

Open Metadata Interface

```
840  data _null_ ;
841  length uriOBJ $256;
842      nobj=metadata_getnobj ("omsobj:person?@Name = 'USER1'",1,uriOBJ);
843      rc = metadata_delassn(uriOBJ,"IdentityGroups") ;
844      put rc= ;
845  run ;

NOTE: Variable uriOBJ is uninitialized.
rc=0
NOTE: DATA statement used (Total process time):
      real time          0.34 seconds
      cpu time           0.05 seconds
```

Open Metadata Interface



Open Metadata Interface: supprimer objet

```
data _null_ ;
length uriOBJ $256;
        nobj= metadata_getnobj
         ("omsobj:person?@Name = 'USER1'"
         ,1
         ,uriOBJ);
        rc = metadata_delobj(uriOBJ) ;
        put rc= ;
run ;
```

Enfin, il vous faudra souvent effacer des objets. La fonction **metadata_delobj** s'acquittera de cette tâche.

Attention, comme nous l'avons déjà évoqué dans la partie concernant la suppression des méta-données à travers l'onglet « **Supprimer des méta données** » de l'utilitaire de méta-données , **la suppression d'un objet n'entraîne pas forcement la suppression des objets qui lui sont attachés**. Nous allons ici supprimer l'utilisateur « USER1 » mais cette suppression n'aura pas d'effet sur l'objet de type « login » attaché à ce dernier.

Open Metadata Interface

```
846  data _null_ ;
847  length uriOBJ $256;
848      nobj=metadata_getnobj ("omsobj:person?@Name = 'USER1'",1,uriOBJ);
849      rc = metadata_delobj(uriOBJ) ;
850      put rc= ;
851  run ;

NOTE: Variable uriOBJ is uninitialized.
rc=0
NOTE: DATA statement used (Total process time):
      real time          0.11 seconds
      cpu time           0.03 seconds
```

Ordre du jour

- Introduction
- La SAS Management Console: outil de découverte des métadonnées
- Manipulation des métadonnées
  - Exemple: la définition des utilisateurs
- Open Metadata Interface, ou comment manipuler les métadonnées par programmation
- Conclusion
- Ressources

Conclusion

Attention:

Le serveur de métadonnées est le « maillon faible » de la plate-forme, la disponibilité de la plate-forme est directement liée à la disponibilité du serveur de métadonnées.

Une requête sur les métadonnées sera en concurrence avec les requêtes des utilisateurs générées par les logiciels clients (SAS Enterprise Guide).

Sauvegardez systématiquement les métadonnées avant toute modification…

Bonne programmation !!

Ressources

- SAS® 9.3 Metadata Model: Reference
- SAS® 9.4 Language Interfaces to Metadata
- SAS® 9.4 Open Metadata Interface: Reference and Usage
- SAS® 9.4 Intelligence Platform Security Administration Guide

p.maurice@decision-network.eu