Paper SAS3500-2019

# Honey, Where Are My Grants? Tips for Creating a SAS® Viya® Content-Security Model Based on SAS®9

Mark Dragone, SAS Institute Inc.

## ABSTRACT

Generally speaking, a well-designed content security model for a SAS® software deployment is an intentional balance between authorizations and those who require them. Maintaining this balance is an ongoing task for SAS administrators. The process for implementing a security model as part of a SAS® Viya® 3.4 deployment needs to be intentional as well. Fortunately, using an existing SAS®9 environment that has evolved in accordance with the corporate security model can prove very valuable. This value is not limited to the initial implementation of the SAS Viya 3.4 security model. The SAS®9 environment's model can also be referenced when the two environments run in parallel.

This paper introduces techniques and processes to identify SAS®9 metadata objects that include directly (and intentionally) applied authorizations, and it explains how to retain those authorizations in SAS Viya 3.4. One of the processes that are discussed includes an automated solution using metadata security macros to deconstruct an Access Control Template (ACT) that is applied to SAS®9 objects. Deconstructing the ACT enables you to retain the authorizations when you promote SAS®9 content to SAS Viya 3.4. For ongoing administrative tasks, or when automation is not necessary, this paper demonstrates how to use Platform Object Framework administration tools, SAS®9 Security Report Macros, SAS® Management Console, SAS Viya 3.4 command-line interface (CLI), and SAS Viya 3.4 Environment Manager.

## INTRODUCTION

Most likely, SAS administrators do not often have users who complain about having authorization to more content than they should have. However, authorization does become an issue when a user is denied access to some object. Designing and implementing a content security model that balances user requirements and institutional security requirements requires planning. This paper presents tools that might be helpful in the content-security design phase as well as the implementation phase.

SAS administrators with a complex SAS®9 security model who want to retain that balance might find some of these tools especially helpful as they transition to SAS Viya. One benefit of these tools includes identifying SAS®9 metadata folders with authorizations that are set by direct access controls and why that setting is important. This paper describes a process for using the promotion tools to retain those authorizations when you import them to SAS Viya. The discussion includes a program that automates the process to allow certain direct access controls to be included in the promotion process to SAS Viya that would otherwise be discarded.

## SETTING DIRECT ACCESS CONTROLS IN SAS®9 AND SAS® VIYA®

While SAS Viya and SAS®9 have differences in how authorizations are evaluated, the guidelines for implementing security are similar. As a best practice, you should use groups, not individual users, when you set direct access controls. You should set direct access controls on folders also, rather than directly on objects.[1,2] These guidelines create a predictable and manageable security model where those direct access controls define authorizations for existing content as well as any new content.

A *direct access control* is an authorization that is set directly on an object. This setting represents a deviation in the authorization-decision model because its precedence is higher than inheritance. The approach for setting direct access controls in SAS®9 and SAS Viya differ.

SAS®9 and SAS Viya both allow you to set direct access controls explicitly. When you set an authorization explicitly in SAS®9, it is referred to as an access control entry (ACE). These explicit SAS®9 authorizations are retained by default when that object is promoted to SAS Viya, which preserves the content-security model.

The difference between SAS®9 and SAS Viya is that SAS®9 allows you to set direct access controls by applying an access control template on an object.

Access control templates are defined in the *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition* as follows:

> . . . a reusable named authorization pattern that you can apply to multiple resources. An access control template consists of a list of users and groups and indicates, for each user or group, whether permissions are granted or denied.

An *ACT* is also a metadata object. The object contains everything that is listed in the definition above. Once the ACT metadata object is defined, that ACT is applied to other metadata objects (usually in a metadata folder).

Because SAS Viya does not have an ACT equivalent, the direct access controls that are defined on the object by the ACT are dropped when that object is promoted to SAS Viya.

To retain the impact that an ACT has on a SAS®9 security model, you need to set the authorizations on those objects in such a way that the promotion process can retain them when the objects are imported to SAS Viya.

## CONTENT PROMOTION CONSIDERATIONS

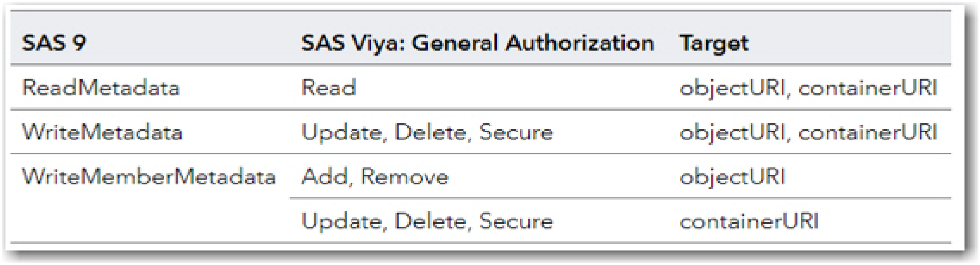
Before explaining how to retain grants, you should be aware that retaining grants is not always a good idea—at least not for all your content. For example, you might not want to promote an entire SAS®9 folder tree of objects that are not supported in the target SAS Viya release. Doing so just defines a folder tree that has no relevance. Similarly, even if the

[1] SAS Institute Inc. 2018. "About Permission Mappings." *SAS® Viya® 3.4 Administration: Orientation to Authorization*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calauthz&docsetTarget=p0g0vol9j5o4t0n1381yo6bc6slp.htm&locale=en#p0je7mhxn1vqzgn1xqikganq4g1y**.

[2] SAS Institute Inc. 2018. "Details: Authorizations." *SAS® Viya® 3.4 Administration: Promotion (Import and Export).* Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calpromotion&docsetTarget=p0lui94y7l3fz5n1uzg6rgzgdyjy.htm&locale=en#n05w20xb1i3xwnn187jgb3dgdfjb**.

objects are participating resources, if the content is no longer used, this is a good opportunity to purge those objects by omitting them from the promotion process altogether.

You should also be aware that  the permissions that these authorizations grant might not translate as you expect. As a result, you need to understand the significant differences for implementing content security between SAS®9 and SAS Viya can occur. When you promote content to SAS Viya, the authorizations are based on the permission mappings. The following table[3] shows how these authorizations are mapped.



| SAS 9 | SAS Viya: General Authorization | Target |
|---|---|---|
| ReadMetadata | Read | objectURI, containerURI |
| WriteMetadata | Update, Delete, Secure | objectURI, containerURI |
| WriteMemberMetadata | Add, Remove | objectURI |
| | Update, Delete, Secure | containerURI |

The section "About Permission Mappings" in *SAS Viya 3.4 Administration: Orientation to Authorization* states the following:

> For example, in SAS®9, a single permission (WriteMetadata) governs the ability to update, delete, and secure content objects. In SAS Viya, you can define access more precisely, enabling one group to update a content object only and another group to both update and secure that content object.[3]

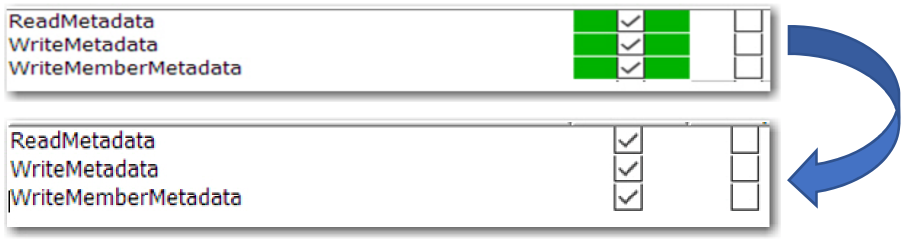
**Note:** Before you update any metadata, be certain to back up that metadata.

## DECONSTRUCTING THE ACCESS CONTROL TEMPLATE (ACT)

Deconstructing the ACT enables you to retain the authorizations when you promote SAS®9 content to SAS. The deconstruction solution is simple.

> If you deconstruct ACTs into individual access control entries (ACEs) before promotion, promotion can process the individual ACEs that are directly set on each promoted object.[4]

You can easily deconstruct an ACT by using SAS Management Console as an administrator to toggle all the ACT granted authorizations (shown in green below) so that they become explicit grants (white/clear).



| | | |
|---|---|---|
| ReadMetadata | 🟩✓🟩 | ☐ |
| WriteMetadata | 🟩✓🟩 | ☐ |
| WriteMemberMetadata | 🟩✓🟩 | ☐ |

| | | |
|---|---|---|
| ReadMetadata | ✓ | ☐ |
| WriteMetadata | ✓ | ☐ |
| WriteMemberMetadata | ✓ | ☐ |

---

[3] SAS Institute Inc. 2018. "About Permission Mappings." *SAS® Viya® 3.4 Administration: Orientation to Authorization*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calauthz&docsetTarget=p0g0vol9j5o4t0n1381yo6bc6slp.htm&locale=en#p0je7mhxn1vqzgn1xqikganq4g1y.

[4] SAS Institute Inc. 2018. "Details: Authorizations." *SAS® Viya® 3.4 Administration: Promotion (Import and Export)*. Cary, NC: SAS Institute Inc. Available at go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calpromotion&docsetTarget=p0lui94y7l3fz5n1uzg6rgzgdyjy.htm&locale=en#n05w20xb1i3xwnn187jgb3dgdfjb.

A good place to start is by identifying the metadata objects to which ACTs have been applied. (**Caution:** Do not confuse identifying the actual ACT metadata object with a metadata object to which the ACT is applied.) Finding the metadata objects to which ACTs are applied requires some work.

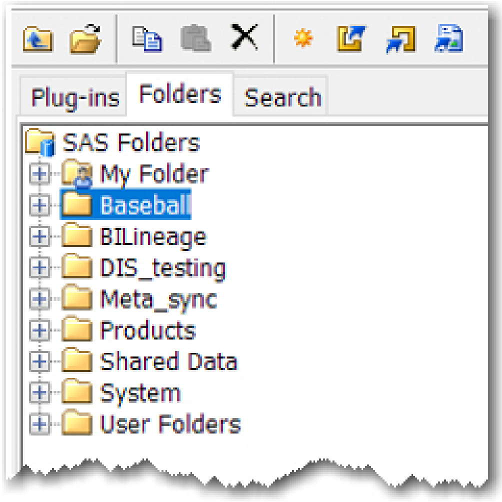
## BUILDING THE REPORT

The following steps demonstrate a programmatic way to create an output file (report) for all objects where an authorization is granted by an ACT that is directly applied to the object.

### Step 1: Build Authorization Tables

Start by choosing a location in metadata where you want to identify objects within its folder tree that have authorizations that are granted by an applied ACT.

Display 1 shows a root-level metadata folder named **`Baseball`**. The next example creates a report that is built from the authorization tables of this folder.



**Display 1. SAS Management Console View of a Folder Value for the %MDSECDS Macro**

These tables that are built with %MDSECDS are referenced throughout this paper.

In the following example, %MDSECDS creates several authorization tables. (These tables include a lot of useful information that might be helpful for other administrative tasks, too.)

```
libname secds "/tmp/secds/";

%mdsecds(folder="/Baseball",includesubfolders=yes,includetablecomponents
          =no,outdata=secds.perms)
```

**Note:** Additional arguments for generating the authorization tables are available in the %MDSECDS section of the *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*. Arguments that are not discussed in this paper might be useful in some instances (for example, when you specify MEMBERYTYPES="FOLDER").

## Step 2: Create a Table for ACT-Granted Authorizations

After you create the authorization tables, you need to use them to create another table, SECDS.ACTG. This table includes authorizations for an object that has an ACT-granted authorization.

Submit the following code to generate the SECDS.ACTG table:

```
proc sql;
   create table secds.actg as
   select s.objname, o.location, s.identityname, s.permission, s.objuri,
          s.authorization
   from secds.perms_permsl as s
   left join secds.perms_objs as o
        on s.objuri=o.objuri
   where s.objuri in
      (select distinct s.objuri from secds.perms_permsl
       where s.authorization in ("Granted by ACT") );
 quit;
```

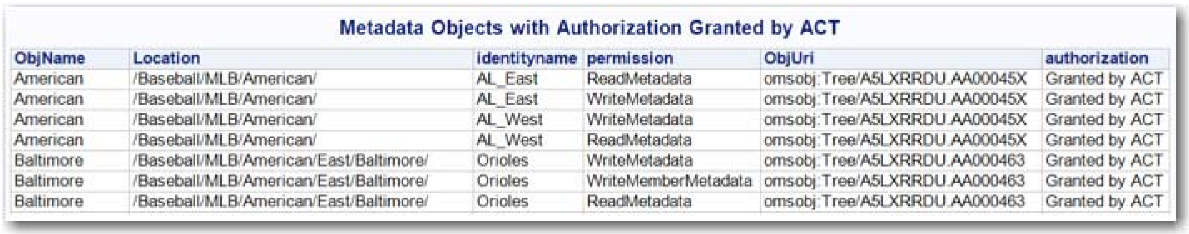
The table that is generated from this code includes all objects with authorizations that are granted by an ACT.

## Step 3: Create an Output File

To generate a report for those objects, submit the following PRINT procedure:

```
proc print data=secds.actg noobs;
    where authorization contains "Granted by ACT";
    title "Metadata Objects with Authorization Granted by ACT";
run;
```

The report should look similar to the following:

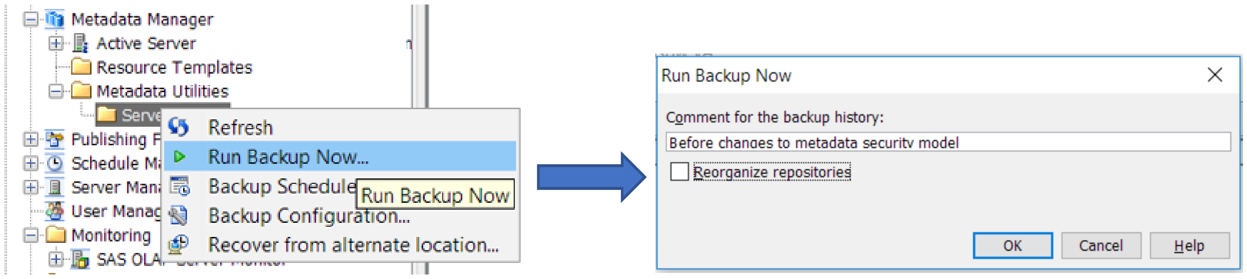


Metadata Objects with Authorization Granted by ACT

| ObjName | Location | identityname | permission | ObjUri | authorization |
|---|---|---|---|---|---|
| American | /Baseball/MLB/American/ | AL_East | ReadMetadata | omsobj:Tree/A5LXRRDU.AA00045X | Granted by ACT |
| American | /Baseball/MLB/American/ | AL_East | WriteMetadata | omsobj:Tree/A5LXRRDU.AA00045X | Granted by ACT |
| American | /Baseball/MLB/American/ | AL_West | WriteMetadata | omsobj:Tree/A5LXRRDU.AA00045X | Granted by ACT |
| American | /Baseball/MLB/American/ | AL_West | ReadMetadata | omsobj:Tree/A5LXRRDU.AA00045X | Granted by ACT |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMemberMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | ReadMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT |

Program 1 in the Appendix illustrates how to generate a similar table that includes all objects with any authorization that is granted or denied by a direct access control. This table can be useful for administrative tasks that are not limited to deconstructing an ACT. For example, you can use this table to identify changes in an environment's security model.

This is a good time to run a metadata backup (that is, before you make changes to the security model). When you run the backup, include a comment indicating that this backup has no changes to the security mode.



Run Backup Now

Comment for the backup history: Before changes to metadata security model

☐ Reorganize repositories

**Display 3. Running a Metadata Backup**

## UPDATING THE SECURITY MODEL

Now you have all the information that you need to deconstruct the ACTs, and your metadata is backed up. You should update the security model during an outage. The updates to the security model need to be in place only long enough to export the content. Keep in mind that the objects in your output are limited to the objects within the folder tree that are specified in the %MDSECDS macro.

## UPDATING THE SECURITY MODEL USING SAS® MANAGEMENT CONSOLE

With the report that is created in Step 3 (assuming it contains a manageable number of objects), you can use SAS Management Console to set the authorization for an object so that it is explicitly granted. This process is described earlier in the section "Deconstructing the Access Control Template (ACT)."

## UPDATING THE SECURITY MODEL PROGRAMMATICALLY

If the PRINT procedure that is shown earlier indicates that the environment relies heavily on ACTs, there is a programmatic approach to deconstructing the ACTs. You can use DATA step functions for metadata security administration to set an explicit grant on an object. The example program below uses the SECDS.ACTG table that was created earlier to automate the deconstruction process by setting an explicit grant (ACE grant) for all authorizations granted by the ACT:

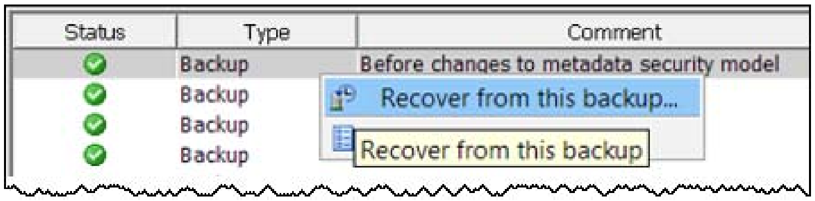
```
data _null_;
   set secds.actg(keep=objuri);
   format tc $20.;
   length uri $ 256 identitytype identityname permission name $ 60
          objuri $ 200 auth $ 18 id type $ 20 condval authorization $ 30
          authint 8;
   tc="";
   uri=objuri;
   rc=metadata_resolve(uri,type,id);
   rc=metadata_getattr(uri,"Name",name);
   put "Authorizations granted from ACTs that are directly applied are
        converted to explicit grants" type= name= id=;
```

*(code continued)*

```
    rc=0;
    n=1;
    do while (rc=0);
    condval=""; auth=""; identityname=""; identitytype="";
                authorization=""; permission="";
    rc=metasec_getnauth(tc,uri,n,identitytype,identityname,auth,permissio
    n,condval);
    if (rc=0)then do;
       n=n+1;
       authint = input(auth, 16.);
       authorization = "";
      if (band(authint,&_SECAD_PERM_ACTM)) then do;
         if (band(authint,&_SECAD_PERM_ACTG))
         then
            rc=METASEC_SETAUTH(tc,uri,identitytype,identityname,'G',per
            mission,condval);
         else put "This program is for objects with grants from the
                  explicitly applied ACT.";
         end;
      end;
    end;
  run;
```

## RESTORING A SECURITY MODEL USING SAS® MANAGEMENT CONSOLE

Regardless of which method you choose to update the security model, you can use SAS Management Console to restore the security model by recovering the backup that you took before the changes.



| Status | Type | Comment |
|---|---|---|
| ✓ | Backup | Before changes to metadata security model |
| ✓ | Backup | |
| ✓ | Backup | |
| ✓ | Backup | |

## RESTORING A SECURITY MODEL PROGRAMMATICALLY

Although recovery of metadata from the backup is the simplest approach for most people, you can also handle this programmatically. For the programmatic approach, you update a single line of code from the example that is used to set the explicit grants.

The following line is from the original program. The 'G' instructs the authorization to be granted explicitly.

```
  rc=metasec_setauth(tc,uri,identitytype,identityname,'G',permission,condv
  al)
```

If the input table (SECDS.ACTG) has not been re-created or changed in any way since you used it to set the explicit grants, then you can use that same table to identify all the objects that need to have the explicit grant removed. To do that, simply replace 'G' with 'R'. The updated line looks like this:

```
rc=metasec_setauth(tc,uri,identitytype,identityname,'R',permission,condv
al)
```

## PREPARING SAS® VIYA® IDENTITIES FOR CONTENT TO BE IMPORTED

At this point, you have exported content from SAS®9, and you have restored the SAS®9 environment security model. However, the SAS Viya environment might not be ready to have the content imported. The *SAS® Viya® 3.4 Administration* guide describes the tasks for promoting SAS®9 content to SAS Viya into four steps.

- Step 1: Create an initial map of system information
- Step 2: Promote internal groups
- Step 3: Promote data definitions
- Step 4: Promote content

This paper primarily focuses on an aspect of Step 4, which is for promoting content. The paper assumes that the first three steps in that section are implemented successfully.

The next sections discuss the following topics:

- differences in the impact that SAS®9 user and group definitions have on content security
- restoring custom group memberships in SAS Viya that are dropped during content promotion

### DIFFERENCES IN USER AND GROUP DEFINITIONS IMPACT ON CONTENT SECURITY

The section "Step 2: Promote Internal Groups" is especially relevant to maintaining the intended authorizations. So far, this paper has focused almost entirely on authorizations for metadata objects. However, maintaining identities for whom the authorizations are set is also important. Be aware that user management processes for SAS®9 and SAS Viya are quite different. Before you promote content to SAS Viya, you need to be familiar with the differences. The process for promoting SAS®9 identities to SAS Viya is covered in great detail in the section Details: Identities in *SAS® Viya® 3.4 Administration: Promotion (Import and Export)*.

SAS®9 enables metadata users and groups to be synchronized by using Lightweight Directory Access Protocol (LDAP). When LDAP synchronization is implemented, LDAP attributes are saved in the metadata definition for that user or group. Because LDAP is not required in SAS®9, not all users and groups have those metadata attributes. As described in Scope and Approach in *SAS® Viya® 3.4 Administration: Promotion (Import and Export),* these attributes impact how the identity is promoted to SAS Viya.

The next section discusses a single scenario that might not apply to all environments but that is significant to the security model.

### RESTORING CUSTOM GROUP MEMBERSHIPS IN SAS® VIYA® THAT ARE DROPPED DURING THE PROMOTION PROCESS

A SAS®9 group that is not included in an LDAP synchronization process is referred to as a *metadata-only group.* When a metadata-only group is promoted to SAS Viya, a SAS Viya custom group is defined. For that group to retain the same group members that it has in

SAS®9, the members must be LDAP accounts. Metadata-only groups are not retained as members.

For example, consider that a members of the Orioles metadata-only group are authorized to a metadata folder that is based on the Orioles membership in the AL_East metadata-only group. When these groups are promoted (during step 2), the Orioles membership in AL_East is not preserved. Subsequently in SAS Viya, when the corresponding folder and direct access control is defined, the members of the Orioles custom group do not have authorization to the folder.

## STEP 1: BUILD A GROUP MEMBERSHIP TABLE

You can use the SAS®9 environment to show the hierarchy of group memberships. Then you can update the groups in SAS Viya to create memberships that were lost during the promotion of the SAS®9 identities.

The following example uses the user-import macro %MDUEXTR[5] to create canonical tables that include and reshape the data for reporting:

```
libname xtra "/tmp/ident";
%mduextr(libref=xtra);

proc sql;
   create table xtra.gmemnorole as
          select g.id, g.name, g.memid, g.memname, r.grouptype
                 from xtra.groupmemgroups_info as g
                 left join xtra.group_info as r
                 on g.id=r.id where r.grouptype ne "ROLE";
quit;

proc sort data=xtra.gmemnorole
   out=xtra.sortgrpmem(rename=(name=Grpname));
   by name;
run;

proc transpose data=xtra.sortgrpmem
               out=xtra.grpmemt(drop=_name_ _label_) prefix=Grpmem;
    by grpname;
    var memName;
run;
```

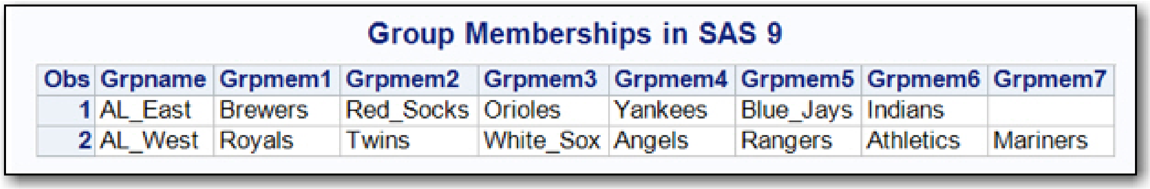
[5] SAS Institute Inc. 2016. "%MDUEXTR" in 'Appendix 2: User Import Macros." *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/api/docsets/bisecag/9.4/content/bisecag.pdf?locale=en#nameddest=n024i4nqa5b12qn1lfek77h69ns5**.

## STEP 2: CREATE AN OUTPUT FILE

Using those tables, create an output file that shows all nested group memberships in the SAS®9 environment.

```
proc print data=xtra.grpmemt;
   title 'Group Memberships in SAS 9';
run;
```

The following output is an example of how the groups memberships appear. The initial column (**`Grpname`**) specifies the group. Each subsequent column (**`Grpmem*`**) indicates a direct member of that group.



Group Memberships in SAS 9

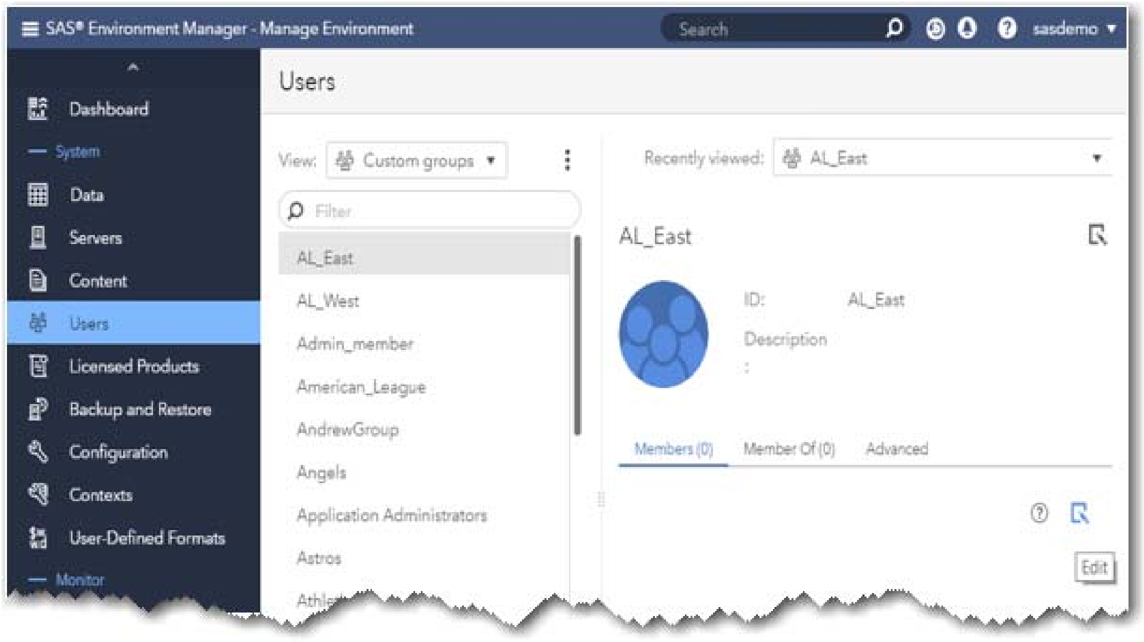
| Obs | Grpname | Grpmem1 | Grpmem2 | Grpmem3 | Grpmem4 | Grpmem5 | Grpmem6 | Grpmem7 |
|---|---|---|---|---|---|---|---|---|
| 1 | AL_East | Brewers | Red_Socks | Orioles | Yankees | Blue_Jays | Indians | |
| 2 | AL_West | Royals | Twins | White_Sox | Angels | Rangers | Athletics | Mariners |

SAS Technical Support does not recommend that you automate this step because doing so will likely include groups that are not relevant to SAS Viya.

Using the output list, you can make any necessary changes to the group memberships by using either the SAS® Environment Manager or the command-line interface, as follows:

In SAS Environment Manager:

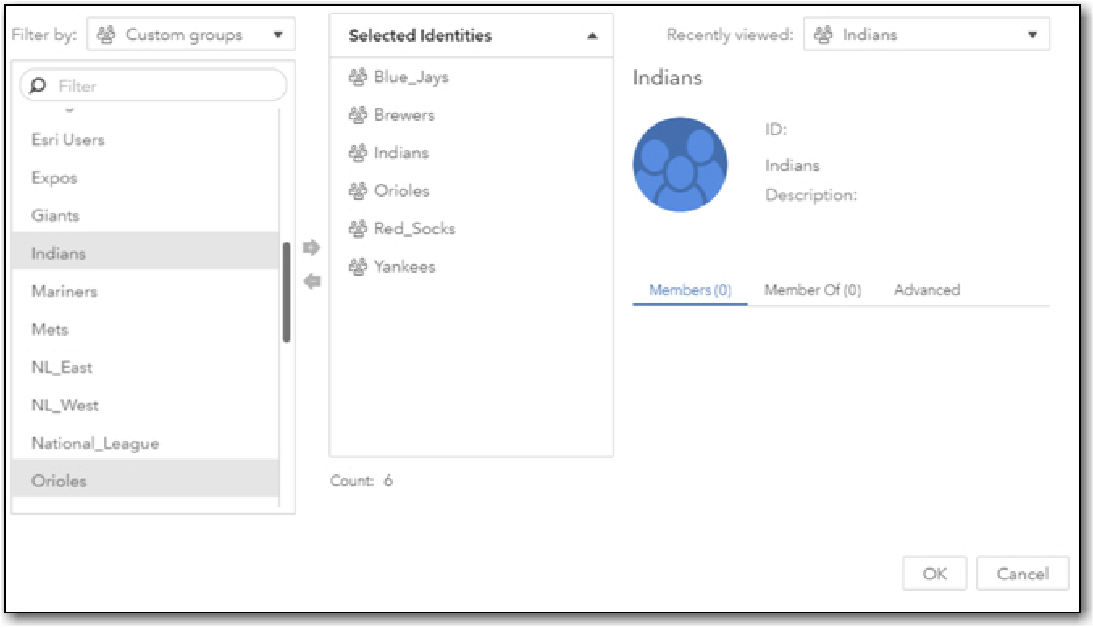
1. Select **Users**.
2. Click the **View** drop-down box and select **Custom Groups**.
3. Select the group that you want to update and click **Edit**, as shown in Display 4:



**Display 4. Using SAS® Environment Manager to Make Changes**

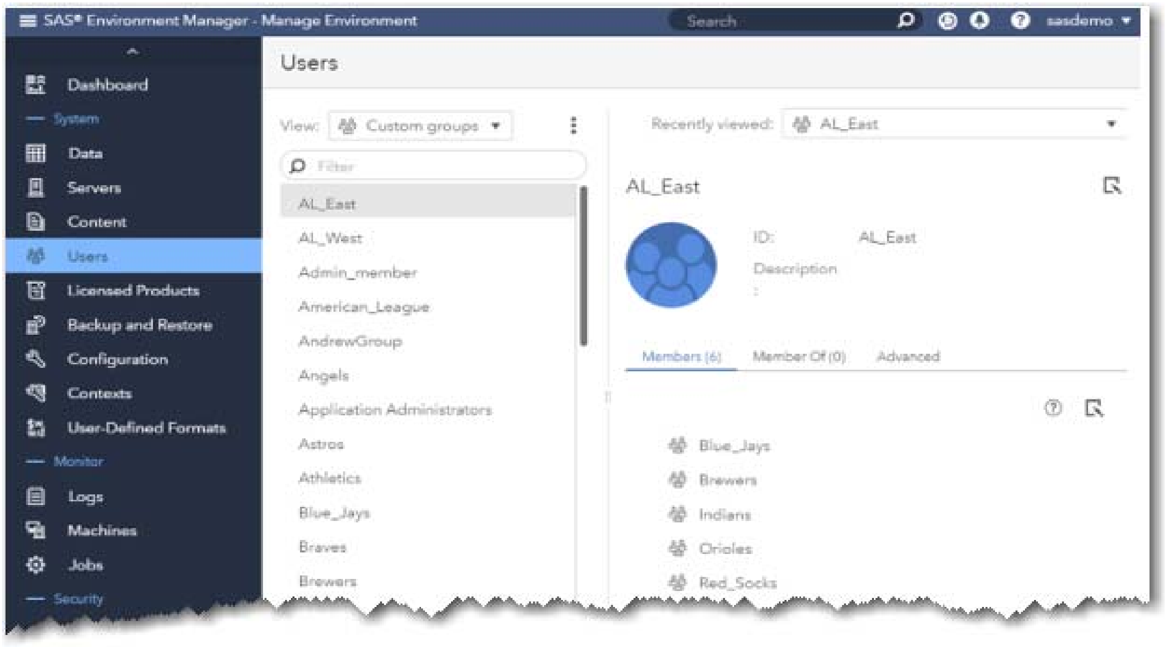
4. Click the **View** drop-down list and select **Custom Groups**.

5. Select a group that you want to add as a member. Then click the arrow to move that group to the **Selected Identities** column, as shown below. After you select and move all group members, click **OK**.



**Display 5: Adding Groups as Members**

Now, when you select a custom group from the **Users** page, you can see the members, as shown in Display 6:



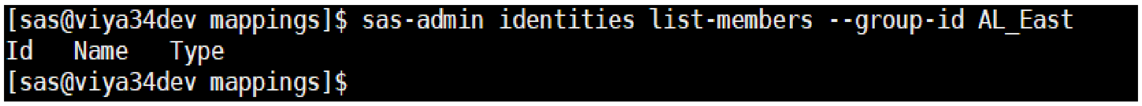
**Display 6. Viewing the Group Members from the Users Page**

You can also make these administrative changes by using the SAS Viya command-line interface (CLI). If you are not familiar with how to use the CLI, see to the Command-Line

Interface: Preliminary Instructions in *SAS® Viya® 3.4 Administration: Using the Command-Line Interfaces*.

The following example uses the **sas-admin** CLI with the **identities** plug-in.

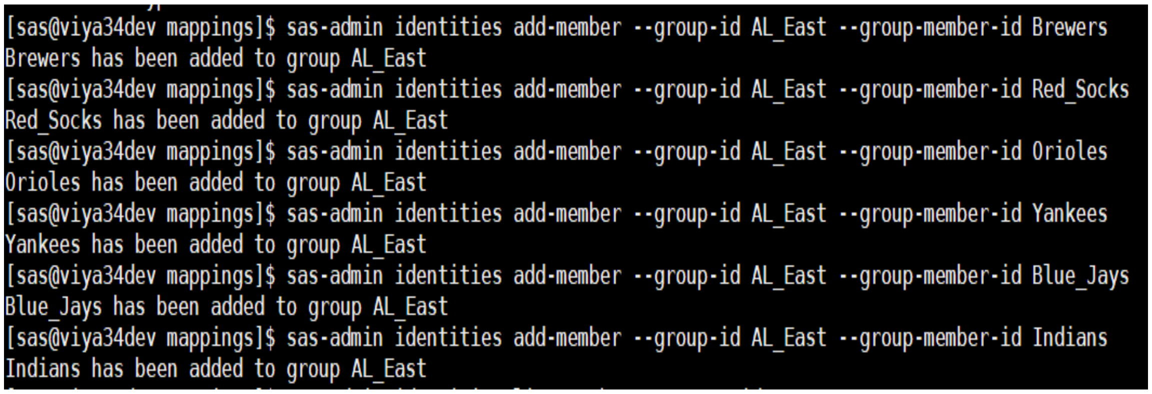
1. Submit the following command to obtain a list of group members of the AL_East group:



```
[sas@viya34dev mappings]$ sas-admin identities list-members --group-id AL_East
Id    Name    Type
[sas@viya34dev mappings]$
```

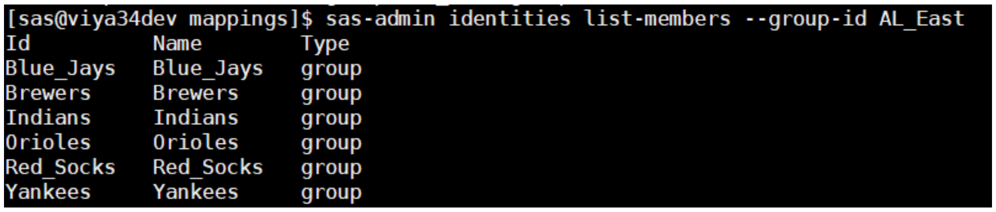
   Notice that there are no members currently in the AL_East group.

2. Submit the following commands to add the groups as members of the AL_East group. Enter the specific value for **–group-member-id** for each group that you want to add.



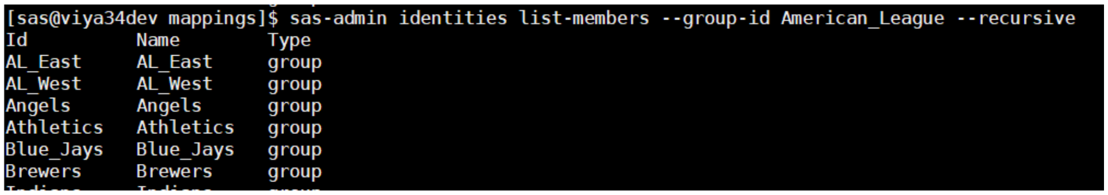
```
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Brewers
Brewers has been added to group AL_East
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Red_Socks
Red_Socks has been added to group AL_East
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Orioles
Orioles has been added to group AL_East
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Yankees
Yankees has been added to group AL_East
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Blue_Jays
Blue_Jays has been added to group AL_East
[sas@viya34dev mappings]$ sas-admin identities add-member --group-id AL_East --group-member-id Indians
Indians has been added to group AL_East
```

3. To verify that the members are added, resubmit the initial **list-members** command:



```
[sas@viya34dev mappings]$ sas-admin identities list-members --group-id AL_East
Id          Name        Type
Blue_Jays   Blue_Jays   group
Brewers     Brewers     group
Indians     Indians     group
Orioles     Orioles     group
Red_Socks   Red_Socks   group
Yankees     Yankees     group
```

As you work through the group members, you can add the **–recursive** option to see a more complete picture of nested group membership, as shown in this example:



```
[sas@viya34dev mappings]$ sas-admin identities list-members --group-id American_League --recursive
Id          Name        Type
AL_East     AL_East     group
AL_West     AL_West     group
Angels      Angels      group
Athletics   Athletics   group
Blue_Jays   Blue_Jays   group
Brewers     Brewers     group
```

Notice that this command returns members of the American_League group, including members of all groups in the hierarchy below this group.

## PROMOTION PROCESS

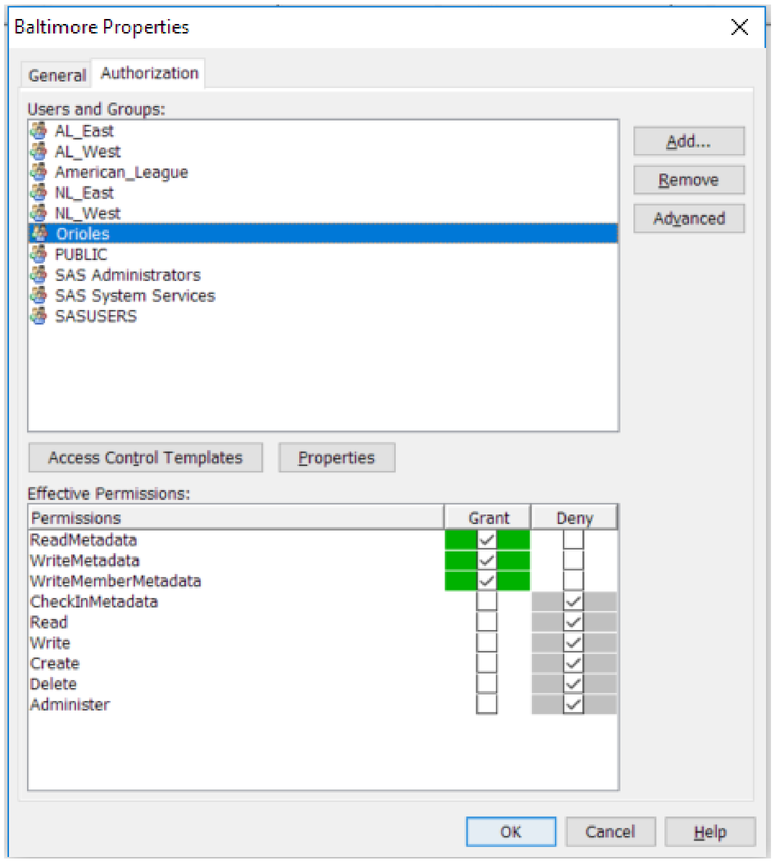
Once the identities are in order, you can proceed with the promotion process as documented in the *SAS® Viya® 3.4 Administration* guide. Be sure to familiarize yourself with the process (specifically, how to use mapping).

Make a backup of your SAS Viya environment before you import the package and verify the results of each import process.

## SCENARIO 1: PROMOTING A SAS®9 METADATA FOLDER TO SAS® VIYA®

This section provides a brief demonstration for a couple of scenarios about how authorizations are impacted during promotion to SAS Viya.

The first scenario involves promoting a SAS®9 metadata folder to SAS Viya. The authorizations, shown below, include authorization grants based on an ACT that is applied to the metadata folder. Then, the folder is exported with SAS Management Console.



Baltimore Properties

General | Authorization

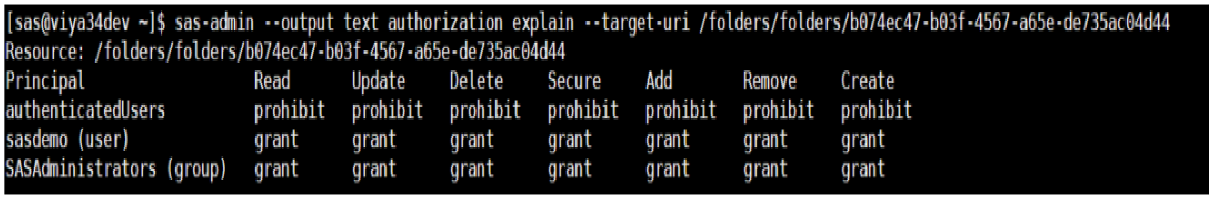
Users and Groups:
- AL_East
- AL_West
- American_League
- NL_East
- NL_West
- Orioles
- PUBLIC
- SAS Administrators
- SAS System Services
- SASUSERS

Effective Permissions:

| Permissions | Grant | Deny |
|---|---|---|
| ReadMetadata | ✓ | |
| WriteMetadata | ✓ | |
| WriteMemberMetadata | ✓ | |
| CheckInMetadata | | ✓ |
| Read | | ✓ |
| Write | | ✓ |
| Create | | ✓ |
| Delete | | ✓ |
| Administer | | ✓ |

**Display 7. Authorizations for the Orioles Group**

After the folder is promoted to SAS Viya, use the **sas-admin** CLI with the **authorization** plug-in to verify the authorizations.

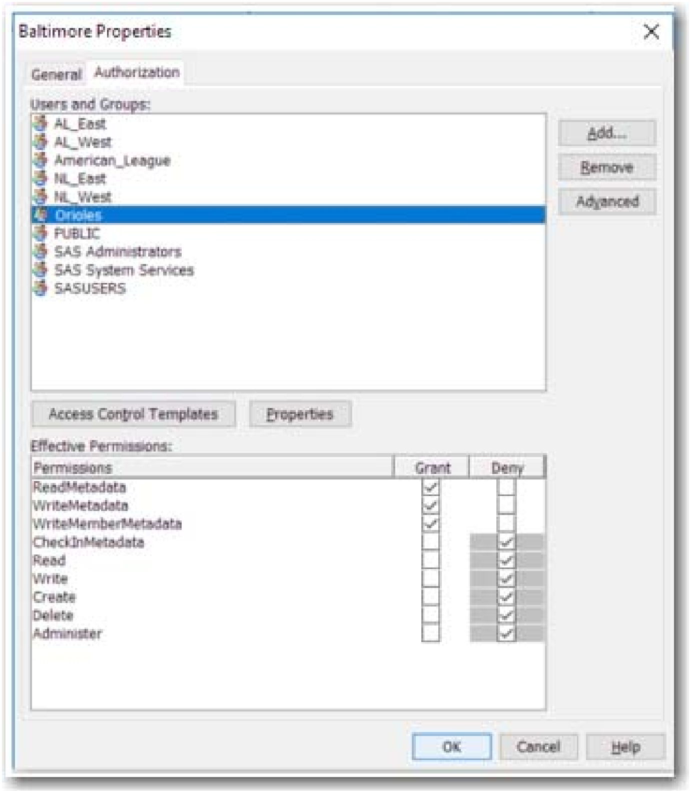


```
[sas@viya34dev ~]$ sas-admin --output text authorization explain --target-uri /folders/folders/b074ec47-b03f-4567-a65e-de735ac04d44
Resource: /folders/folders/b074ec47-b03f-4567-a65e-de735ac04d44
Principal                 Read      Update    Delete    Secure    Add       Remove    Create
authenticatedUsers        prohibit  prohibit  prohibit  prohibit  prohibit  prohibit  prohibit
sasdemo (user)            grant     grant     grant     grant     grant     grant     grant
SASAdministrators (group) grant     grant     grant     grant     grant     grant     grant
```

As expected, the grants are not retained. In keeping with the baseball theme, that is strike one!

## SCENARIO 2: RETAINING THE SECURITY MODEL BY CHANGING AUTHORIZATIONS TO EXPLICIT GRANT

However, you can take another swing at retaining the security model by changing the authorizations to an explicit grant, as shown below:



Baltimore Properties

General | Authorization

Users and Groups:
- AL_East
- AL_West
- American_League
- NL_East
- NL_West
- Orioles
- PUBLIC
- SAS Administrators
- SAS System Services
- SASUSERS

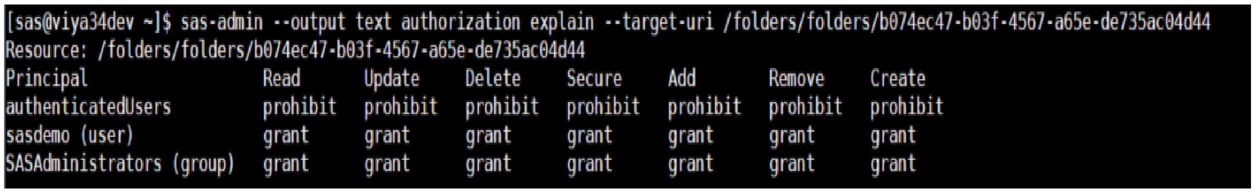
Effective Permissions:

| Permissions | Grant | Deny |
|---|---|---|
| ReadMetadata | ✓ | |
| WriteMetadata | ✓ | |
| WriteMemberMetadata | ✓ | |
| CheckInMetadata | | ✓ |
| Read | | ✓ |
| Write | | ✓ |
| Create | | ✓ |
| Delete | | ✓ |
| Administer | | ✓ |

**Display 8: Changing Authorizations to an Explicit Grant**

Next, re-create the export file and complete the import process to SAS Viya.

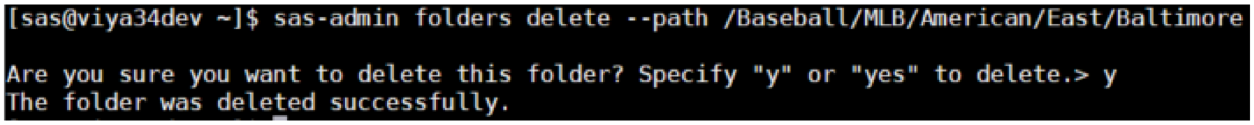
The import process should complete successfully. However, if you verify the authorizations again, you can see that nothing has changed!



```
[sas@viya34dev ~]$ sas-admin --output text authorization explain --target-uri /folders/folders/b074ec47-b03f-4567-a65e-de735ac04d44
Resource: /folders/folders/b074ec47-b03f-4567-a65e-de735ac04d44
Principal                 Read      Update    Delete    Secure    Add       Remove    Create
authenticatedUsers        prohibit  prohibit  prohibit  prohibit  prohibit  prohibit  prohibit
sasdemo (user)            grant     grant     grant     grant     grant     grant     grant
SASAdministrators (group) grant     grant     grant     grant     grant     grant     grant
```

Strike two! Actually, this is expected behavior because the authorizations are set on import only when the object is initially defined.

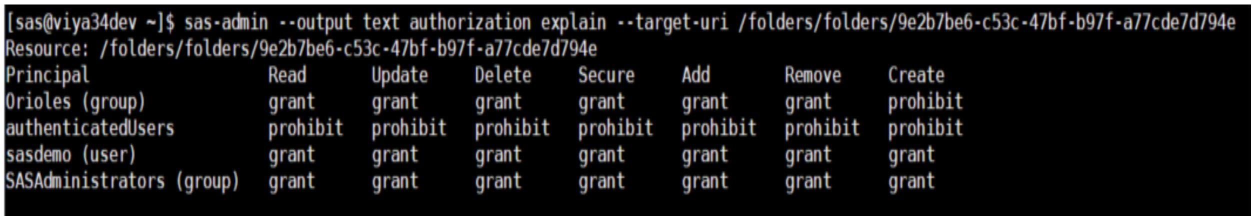
You have only one more swing left. This time, delete the metadata folder by using the **`folders`** plug-in with the **`delete`** option:



```
[sas@viya34dev ~]$ sas-admin folders delete --path /Baseball/MLB/American/East/Baltimore

Are you sure you want to delete this folder? Specify "y" or "yes" to delete.> y
The folder was deleted successfully.
```

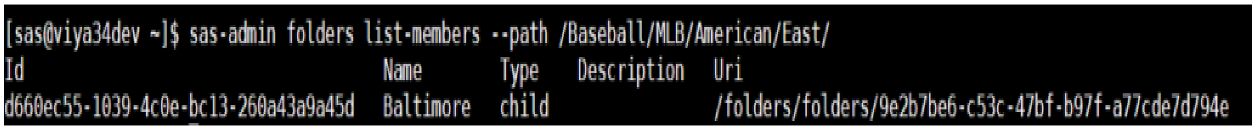
Re-use the second export file that was created above to import the folder. Again, the import process should complete successfully. When you verify the authorizations this time, you should see that the grants are there.



```
[sas@viya34dev ~]$ sas-admin --output text authorization explain --target-uri /folders/folders/9e2b7be6-c53c-47bf-b97f-a77cde7d794e
Resource: /folders/folders/9e2b7be6-c53c-47bf-b97f-a77cde7d794e
Principal                 Read      Update    Delete    Secure    Add       Remove    Create
Orioles (group)           grant     grant     grant     grant     grant     grant     prohibit
authenticatedUsers        prohibit  prohibit  prohibit  prohibit  prohibit  prohibit  prohibit
sasdemo (user)            grant     grant     grant     grant     grant     grant     grant
SASAdministrators (group) grant     grant     grant     grant     grant     grant     grant
```

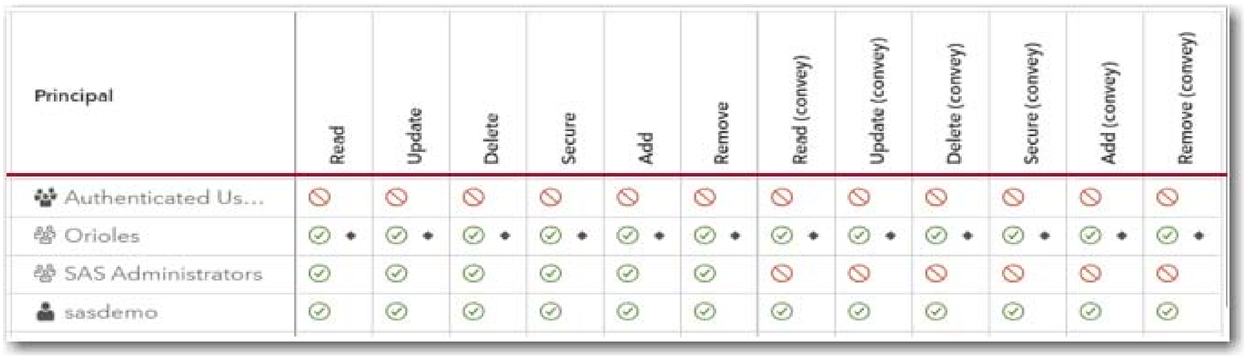
Success!! You did not strike out, and the Orioles group has its grants!

For these examples, the **`folders`** option is used to determine the value for the **`–target-uri`** option that should be used in the authorization command.



```
[sas@viya34dev ~]$ sas-admin folders list-members --path /Baseball/MLB/American/East/
Id                                    Name        Type    Description   Uri
d660ec55-1039-4c0e-bc13-260a43a9a45d  Baltimore   child                 /folders/folders/9e2b7be6-c53c-47bf-b97f-a77cde7d794e
```

You can also use SAS Environment Manager to perform each of these steps. The following display shows how the authorizations look in SAS Environment Manager.



| Principal | Read | Update | Delete | Secure | Add | Remove | Read (convey) | Update (convey) | Delete (convey) | Secure (convey) | Add (convey) | Remove (convey) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Authenticated Us... | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ |
| Orioles | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • | ✓ • |
| SAS Administrators | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ | ⊘ |
| sasdemo | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

**Display 9. Authorizations as They Look in SAS® Environment Manager**

Repeat this process for each folder tree that is exported from SAS®9. Verify that the security model is being defined as intended after each import.

## CONCLUSION

Content security models can be complex and unforgiving. Having a SAS®9 security model that achieves balance between the user and institutional requirements is very useful in implementing content security in SAS Viya. In addition, implementing that security model by retaining it as part of a content promotion process is even more helpful. Hopefully, the tips in this paper will prove useful in helping you to "keep your grants on" as you transition to SAS Viya!

## APPENDIX: ADDITIONAL PROGRAMS FOR ADMINISTRATING SECURITY MODELS

This section provides some additional programs that you can use to administer security models. The first program explains how to list all objects with direct access controls in SAS®9; the second program explains a process for identifying changes and validating updates.

## PROGRAM 1: LISTING ALL OBJECTS WITH DIRECT ACCESS CONTROLS IN A SAS®9 ENVIRONMENT

If you are interested in a better understanding of the existing SAS®9 security model, the following program might be helpful. This program lists all the objects in a SAS®9 environment that have direct access controls. You can use any of the authorization values that are quoted in the SQL procedure's SELECT statement in the WHERE statement of PRINT procedure to create a subsetted list of objects and authorizations, based on your needs.

```
libname secds2 "/tmp/secds2/";
%mdsecds(folder="/Baseball",includesubfolders=yes,includetablecomponents
                                           =no,outdata=secds2.perms)
proc sql;
   create table secds2.actg as
          select s.objname, o.location, s.identityname, s.permission,
                 s.objuri, s.authorization
          from secds2.perms_permsl as s
          left join secds2.perms_objs as o on s.objuri=o.objuri
          where s.objuri in (select distinct s.objuri from
          secds2.perms_permsl
          where s.authorization in ("Granted Explicitly","Granted
          Explicit Condition","Denied Explicitly","Granted by
          ACT","Denied by ACT") );
quit;
```

The table that is created includes all objects with direct-control authorizations. However, if you want to see only those objects with explicit grants, submit the following PROC PRINT statement:

```
proc print data=secds2.actg noobs;
   where authorization contains "Granted Explicitly";
run;
```

## PROGRAM 2: IDENTIFYING CHANGES IN THE SECURITY MODEL AND VALIDATING UPDATES

You can use the example program in this section to identify changes in the security model as well as to validate updates to the security model.  Identifying changes is particularly valuable under the following circumstances:

- You implement a SAS Viya security model that is based on the SAS®9 environment.
- The SAS®9 environment remains online either in parallel to SAS Viya or while SAS Viya is in user acceptance testing.

The following PROC SQL code creates the ACTG table so that it includes all authorizations for all objects:

```
proc sql;
   create table secds.actg as
          select s.objname, o.location, s.identityname, s.permission,
                 s.objuri, s.authorization
```

*(code continued)*

```
          from secds.perms_permsl as s
          left join secds.perms_objs as o
          on s.objuri=o.objuri
          where s.objuri in
               (select distinct s.objuri from secds.perms_permsl);
quit;
```

The next part of the example compares the two ACTG tables through a series of PROC SQL steps that identify the differences between them. In this program, the initial ACTG table is written to the **/tmp/secds** directory, and the new ACTG table is written to **/tmp/new_secds**.

```
libname secds "/tmp/secds/";       /* Location of initial ACTG table */
libname newsecds "/tmp/new_secds"; /* Location of new ACTG table     */

proc sort data=secds.actg out=newsecds.srtace;
   by location identityname;
run;

proc sort data=newsecds.actg
          out=newsecds.newsrtace(rename=(authorization=newauthorization)
          );
    by location identityname;
run;

proc sql;
    create table newsecds.newdiffs as
          select * from newsecds.newsrtace
      except
          select * from newsecds.srtace;
run;

proc sql;
   create table newsecds.diffs as
          select * from newsecds.srtace
                 except
                    select * from newsecds.newsrtace;
run;

proc sql;
   create table newsecds.leftjointgtdiffs as
         select  s.*, t.newauthorization
         from newsecds.diffs as s left join newsecds.newdiffs as t
         on s.location=t.location and s.identityname=t.identityname
            and s.permission=t.permission and s.objname=t.objname;
quit;
```

*(code continued)*

```
proc sql;
    create table newsecds.leftjoinsrcdiffs as
           select  t.*, s.authorization
           from newsecds.newdiffs as t left join newsecds.diffs as s
           on s.location=t.location and s.identityname=t.identityname
             and s.permission=t.permission and s.objname=t.objname;
quit;
```

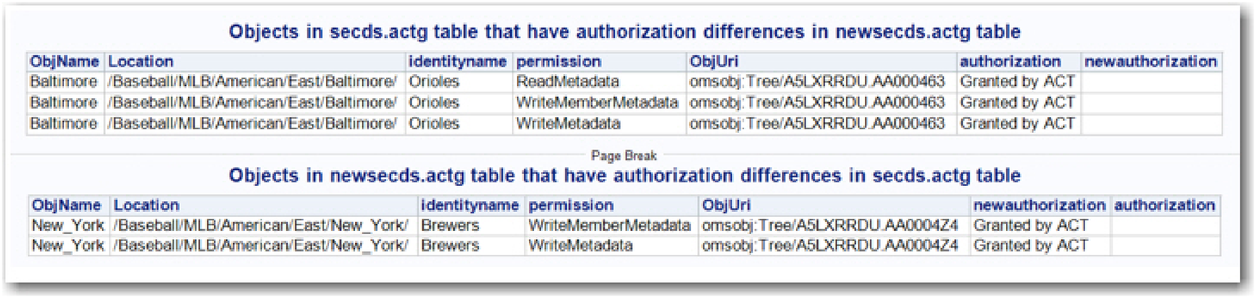
Two tables are created in the code above in order to show the differences. The comparison identifies not only authorization changes in the environments, but whether an object is deleted or added.

The following PROC PRINT steps print output that show the differences between the tables:

```
proc print data=newsecds.leftjointgtdiffs noobs;
   title "Objects in secds.actg table that have authorization
         differences in newsecds.actg table";
run;

proc print data=newsecds.leftjoinsrcdiffs noobs;
   where newauthorization eq "Granted by ACT" and authorization ne
                            "Granted by ACT";
   title "Objects in newsecds.actg table that have authorization
         differences in secds.actg table";
run;
```

If you follow all the steps from earlier in this paper (where the new ACTG table is created to include only objects with authorizations granted by an ACT), the output looks like the following:



Objects in secds.actg table that have authorization differences in newsecds.actg table

| ObjName | Location | identityname | permission | ObjUri | authorization | newauthorization |
|---|---|---|---|---|---|---|
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | ReadMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMemberMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | |

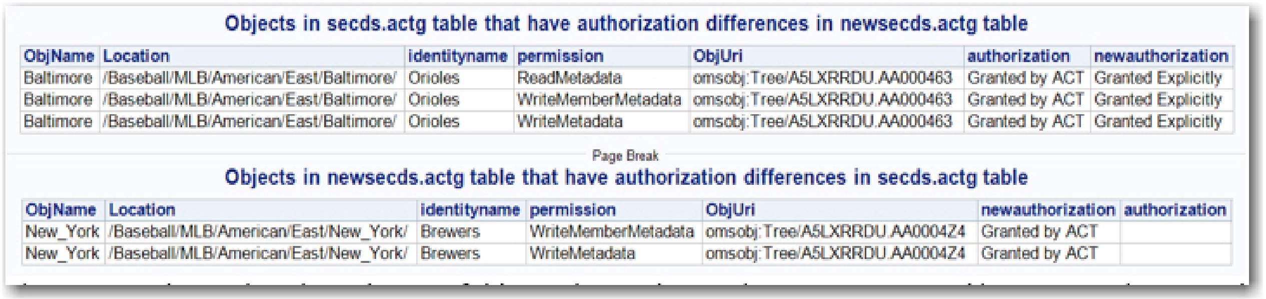
Page Break

Objects in newsecds.actg table that have authorization differences in secds.actg table

| ObjName | Location | identityname | permission | ObjUri | newauthorization | authorization |
|---|---|---|---|---|---|---|
| New_York | /Baseball/MLB/American/East/New_York/ | Brewers | WriteMemberMetadata | omsobj:Tree/A5LXRRDU.AA0004Z4 | Granted by ACT | |
| New_York | /Baseball/MLB/American/East/New_York/ | Brewers | WriteMetadata | omsobj:Tree/A5LXRRDU.AA0004Z4 | Granted by ACT | |

**Output 1. Output That Is Generated When Both Comparison Tables Include Only Authorizations That Are Granted by ACT**

If you create the new ACTG table to include all authorizations and if an authorization has changed, you should see the new effective authorization for an object that previously was granted by an ACT, as shown in Output 2.



Objects in secds.actg table that have authorization differences in newsecds.actg table

| ObjName | Location | identityname | permission | ObjUri | authorization | newauthorization |
|---|---|---|---|---|---|---|
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | ReadMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | Granted Explicitly |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMemberMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | Granted Explicitly |
| Baltimore | /Baseball/MLB/American/East/Baltimore/ | Orioles | WriteMetadata | omsobj:Tree/A5LXRRDU.AA000463 | Granted by ACT | Granted Explicitly |

Page Break

Objects in newsecds.actg table that have authorization differences in secds.actg table

| ObjName | Location | identityname | permission | ObjUri | newauthorization | authorization |
|---|---|---|---|---|---|---|
| New_York | /Baseball/MLB/American/East/New_York/ | Brewers | WriteMemberMetadata | omsobj:Tree/A5LXRRDU.AA0004Z4 | Granted by ACT | |
| New_York | /Baseball/MLB/American/East/New_York/ | Brewers | WriteMetadata | omsobj:Tree/A5LXRRDU.AA0004Z4 | Granted by ACT | |

**Output 2. Output That Is Generated When Additional Authorizations Are Included in Current Authorization Tables**

Output 1 shows that the **`Baltimore`** folder no longer has authorizations that are granted by an ACT. Output 2 indicates a change in the **`New_York`** folder authorizations. Since the original ATCG table was created, an ACT was applied to the folder granting the Brewers authorizations they previously did not have. If you want to implement this change in SAS Viya, use SAS Viya Environment Manager or the SAS Viya CLI to make that change.

## CONCLUSION

Content security models can be complex and unforgiving. Having a SAS®9 security model that achieves balance between the user and institutional requirements is very useful in implementing content security in SAS Viya. In addition, implementing that security model by retaining it as part of a content promotion process is even more helpful. Hopefully, the tips in this paper will prove useful in helping you to "keep your grants on" as you transition to SAS Viya!

## REFERENCES

SAS Institute Inc. 2015. "General Instructions for Using the SAS Intelligence Platform Batch Tools." *SAS® 9.4 Intelligence Platform: System Administration Guide, Fourth Edition*. Cary, NC: SAS Institute Inc. Available at **`go.documentation.sas.com/api/docsets/bisag/9.4/content/bisag.pdf?locale=en#nameddest=p1qepyyxibo846n1morn4w2xuldk`**

SAS Institute Inc. 2018. "Basics of Metadata Administration." *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*. Cary, NC: SAS Institute Inc. Available at **`go.documentation.sas.com/?docsetId=bisecag&docsetTarget=n1gwrrpfx9ujqun17syl8yknhgy0.htm&docsetVersion=9.4&locale=en`**.

SAS Institute Inc. 2018. "Glossary." *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*. Cary, NC: SAS Institute Inc. Available at **`go.documentation.sas.com/?docsetId=bisecag&docsetTarget=glossary.htm&docsetVersion=9.4&locale=en#n1qfzqe7rlk1ctn1p2kqr0j03hfv`**.

SAS Institute Inc. 2018. "General Authorization: Guidelines." *SAS® Viya® 3.4 Administration*. Cary, NC: SAS Institute Inc. Available at **`go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calchkcfg&docsetTarget=n00004saschecklist0000config.htm&locale=en`**.

SAS Institute Inc. 2018. "About Permission Mappings." *SAS® Viya® 3.4 Administration: Orientation to Authorization*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calauthz&docsetTarget=p0g0vol9j5o4t0n1381yo6bc6slp.htm&locale=en#p0je7mhxn1vqzgn1xqikganq4g1y**.

SAS Institute Inc. 2018 "Details: Authorizations." *SAS® Viya® 3.4 Administration: Promotion (Import and Export)*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calpromotion&docsetTarget=p0lui94y7l3fz5n1uzg6rgzgdyjy.htm&locale=en#n05w20xb1i3xwnn187jgb3dgdfjb**.

SAS Institute Inc. 2018. "Details: Identities." *SAS® Viya® 3.4 Administration: Promotion (Import and Export)*. Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=calpromotion&docsetTarget=p0lui94y7l3fz5n1uzg6rgzgdyjy.htm&locale=en#p16apw2lfrxjzxn19cf8hlhwxphx**.

## CONTACT INFORMATION

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

Mark Dragone
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
**Email:** **support@sas.com**
**Web:** **support.sas.com/en/support-home.html**