# METADATA SECURITY IN SAS® 9.4 – BEST PRACTICE: AN EASY START

Umberto Michelucci, Helsana AG
umberto.michelucci@helsana.ch

Version 1.0

Last Updated: September 2016

## Content

## Table of figures

## Background

This document contains a best practice method describing how to implement metadata security in SAS®. The document "Metadata Security in SAS® – Step-by-Step" (Jørgensen & Hoffritz, 2013) has been used as basis for this one. The goal of this paper is to outline a simpler and more streamlined version of metadata security when the number of people working with SAS is small (of the order of 20-30) and are divided in less than 10 groups. The small amount of people and groups allow to simplify heavily the metadata security structure that we should implement.

I will not describe how to do specific operations in SAS (like creating a user or a group) since tools change and new versions come often and assume that the reader has some knowledge on the working of the tool (in particular of the Management Console as I write). Our aim is to describe a methodology that will get you started regardless of what version of SAS you are using, and that will avoid problems along the road in the future.
Additionally note that the aim of this paper is not to give a complete overview on how security in SAS® works. I will describe only a few features, the one we need to implement our best practice methods and nothing more. I will use a "lazy approach" and describe only what I need.

The best practices have been reviewed by J. Fritzenwallner from SAS®.

Note also that referencing in this white paper has been done according to the Harvard Standard. For details see for example (Dhann, 2001).

## Basic Introduction on Security in SAS

Let's first explain briefly what possibilities you have in SAS related to metadata security and how to start.  Each person that work with SAS needs a user created in the Metadata. Without such a user you cannot login in SAS and therefore cannot work. Users can additionally be combined in groups.

Groups should reflect the internal security structure you have in your company. Let's make an example to clarify this idea. Let's suppose we have 4 persons and let's call them Joe, Sarah, Jack and Muriel. Let's say that Joe and Sarah need to access specific models but should not access the ones of Jack and Muriel and *vice versa*. After creating users for all four, you would need to create two groups (let's call them group1 and group2) and assign Joe and Sarah to group1 and Jack and Muriel to group2. Afterwards you will manage permissions only on group level, never on user level. The problem is that when you give direct access to a specific user you will soon forget. And it will be difficult to manage it when the person (for example) will change group or need to change his/her permissions, since you will need to check all resources for permissions given directly. That brings us to the first rule:

**Golden Rule on users and groups:** each user should be at least in one group.

The next important building block in the security setup of SAS is the "Access Control Template" (ACT). An ACT consists of a pattern of grants and denials that are assigned to different users and groups. When you apply an ACT to a resource, the ACT settings are added to the resources' permissions. When you want to assign the same settings to several disparate resources, using an ACT is beneficial for two big reasons:

- "It is easier to apply a pattern than it is to set each permission individually on each resource for which the pattern is appropriate". (SAS(R), 2016a)
- "If you need to change access to the items to which a pattern is applied, you can simply update the permission pattern, rather than revisiting each resource and individually modifying the settings". (SAS(R), 2016a)

There is another small thing one need to understand on how SAS works. When you look at a resource properties and check the "Authorization" tab, the specific permissions (called "Effective Permissions") have the background color coded: white, gray and green.

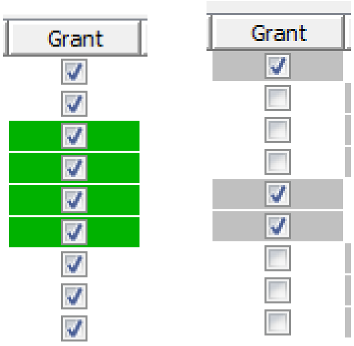

Figure 1 Background of permissions under a Resource Property are color coded.

The meaning of the colors is (SAS(R), 2016a):

- green: the permission is inherited from an ACT;
- gray: the permission are indirect. They usually comes from the parent Resource (when we are dealing with folders the permissions come from the parent folder);
- white: is a directly given permission.

Always remember the following Golden Rule (also known as the G-Rule[1]) (Jørgensen & Hoffritz, 2013):

**Golden Rule on SAS Permissions (G-Rule):**
Green and Gray are good. White is bad.

In the next section I will expand more on this.

## Security Golden Rules

First let's talk about some golden rules that we will need to adhere to. Note that abiding by these rules will allow you enough flexibility and will avoid a lot of problems down the road.

Note that the following rules are coming from (Jørgensen & Hoffritz, 2013). They are slightly adapted and simplified for our purposes, but the main ideas remain the same. We will avoid to reference the document (Jørgensen & Hoffritz, 2013) for each rule to make the text more readable.

**Rule 1:**
Apply only ACT on resources.

It is a bad idea (simply don't do it) to use Access Control Entries (ACEs) (marks with white background on resources or permission directly assigned). The best you can do is always to have a mixture of inheritance (gray background) and ACTs (green background). This will allow you, as administrator, to maintain **all** security changes centrally in the *Authorization Manager Plugin* in the management console. You don't need to go and check all resources for changes. This is probably the most important rule.

**Rule 2:**
Add only groups in ACTs

As I have advised: work with groups, not directly with users. In this way you only need to assign users to groups, and will not need to check each resources for direct permissions to users. The following diagram should give you an idea of the hierarchy you should have in your security metadata in SAS®: users are assigned to groups, groups to ACT and ACT to resources.

[1] The name „G-Rule“ comes from the initials of the colors Green and Gray.

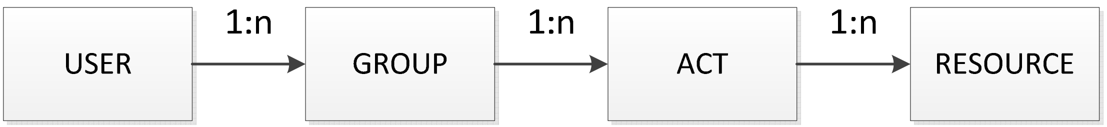

**Figure 2: Suggested hierarchy of SAS® entities**

**Rule 3:**
ACTs with explicit groups (except PUBLIC/SASUSERS) only grant access, never deny.

In this way you will never have ACTs that will work against one another. It will save you later on a lot of headaches. Of course this works only when you deny, by default, permissions for a standard user. The idea is that a standard user will never be able to do anything. With each additional ACT you slowly add permissions for each resource. That brings us to the next rule.

**Rule 4:**
Apply ACT denying RM (Read Metadata) for PUBLIC/SASUSERS groups to each resource.

Remember that if you are in the metadata as user you are always a member of the two groups.

The idea is that if you apply an ACT to a folder granting RM to a group (let's call it G1) and apply a second ACT to the same folder denying RM to group PUBLIC/SASUSERS, only group G1 will be able to see the folder. All others will not be able to.

**Rule 5:**
Implement specifically an ACT for administrators

Remember that also the administrators are members of PUBLIC/SASUSERS groups. If you don't do anything they will be denied access as all other users. Just something to remember.

There is another rule, but I feel this is almost common sense and is almost a meta-rule (a rule on rules): *design and document first and then implement (*another formulation is *plan before doing*). So I won't list it here as rule nr. 6. But keep it in mind when you start.

## Designing ACT – overview

The smallest set of ACTs you will need contains the following:

1. One ACT for each group you have in your organization (given the hypothesis that we are dealing with 20-30 developer, we are usually talking about less than 10 groups)
2. One "Private User Folder ACT" (this controls the permissions for each user's private folder)
3. One "PUBLIC and SASUSERS denied ACT"
4. One "SAS Administrator Settings ACT"

With those ACTs you can start administering your SAS developer community in the right way. Now let's examine each ACT and discuss briefly what kind of rights are necessary in each case.

## Permission patterns for ACTs

First note that I use these permissions in our ACTs
**RM** = Read Metadata: Ability to see a metadata object

**WM** = Write Metadata: Ability to add, modify and delete metadata.
**WMM** = Write Member Metadata: Ability to add, modify and delete metadata objects in folders.
**CM** = Check in Metadata: Ability to check metadata into primary repository from project repository.
**R** = Read: Ability to read data.
**W** = Write: Ability to modify existing data.
**C** = Create: Ability to add new data.
**D** = Delete: Ability to delete data.
**I** = Insert: Ability to add rows to a metadata-bound table
**U** = Update: Ability to update rows to a metadata-bound table.
**S** = Select: Ability to read rows in a physical table in a metadata-bound library.
**Create Table**: Ability to create a new physical table in a metadata-bound library.
**Drop Table**: Ability to delete a physical table in a metadata-bound library.
**Alter Table**: Ability to change or rename a physical table in a metadata-bound library.

The list of permissions can be found when you check the Properties of an ACT under the "Permission Pattern" tab. Each can have only two values: "Grant" or "Deny".

| ACT Name | ACT description | Permissions Pattern (**G**: Grant, **D**: Deny) |
|---|---|---|
| **Group ACT** | This ACT will give the correct permissions to a one of the group of developers/users. (Change the name to identify easily to which group it refers). | **G:** *RM, WM, CM, WMM, R, W, C, D, A, S, I, U, Create, Table, Drop, Table, Alter, Table* |
| **Private User Folder ACT** | This ACT give the correct permission to the private folder of each user. | **G:** *RM, WM, CM, WMM, R, W, C, D, A, S, I, U, Create, Table, Drop, Table, Alter, Table* |
| **PUBLIC and SASUSERS denied ACT** | This ACT denies all rights to PUBLIC and SASUSERS. | PUBLIC **D:** *ALL*<br>SASUSERS **D:** *ALL* |
| **SAS Administrator Settings ACT** | This ACT contains the rights for the administrators. | SAS Administrators **G:** *RM, WM, WMM, CM, A*<br>SAS System Services **G:** *RM* |

## Resolving Conflicts

Sometime permissions (grants and denies) coming from different sources (groups, inheritance, ACTs, ACEs) may be in conflict. A user may have a “grant” and a “deny” for a given resource at the same time and it may be difficult to understand which permission has actually precedence. One need to understand well how permissions are evaluated. The following two Figures give examples of conflicts.

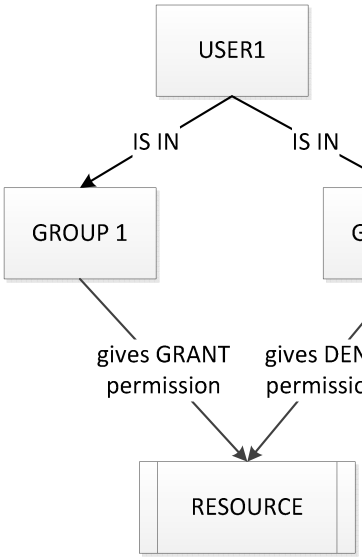

Figure 3: A user is in two groups that gives two different permissions. Which one wins?

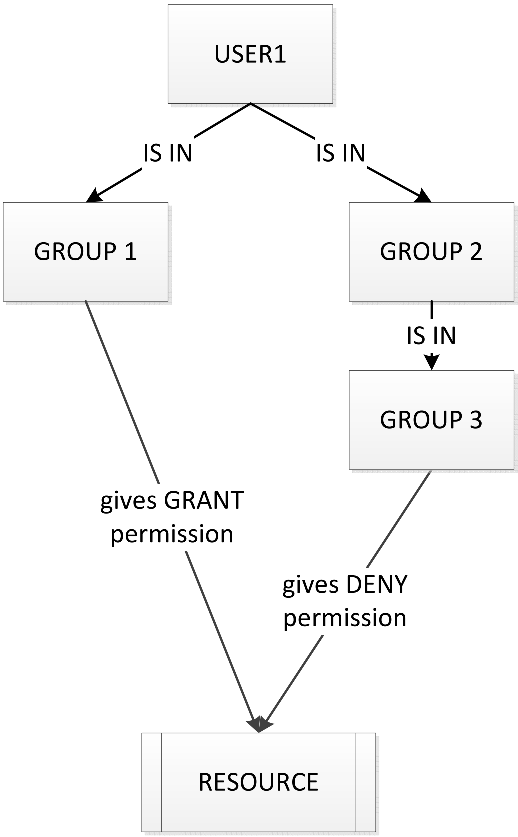

Figure 4: A user is in two groups: GROUP1 and GROUP2. But GROUP2 is in another group (GROUP3). Which permission win in this case?

An useful diagram that will explain how this work is the following (Hopkins, 2008)

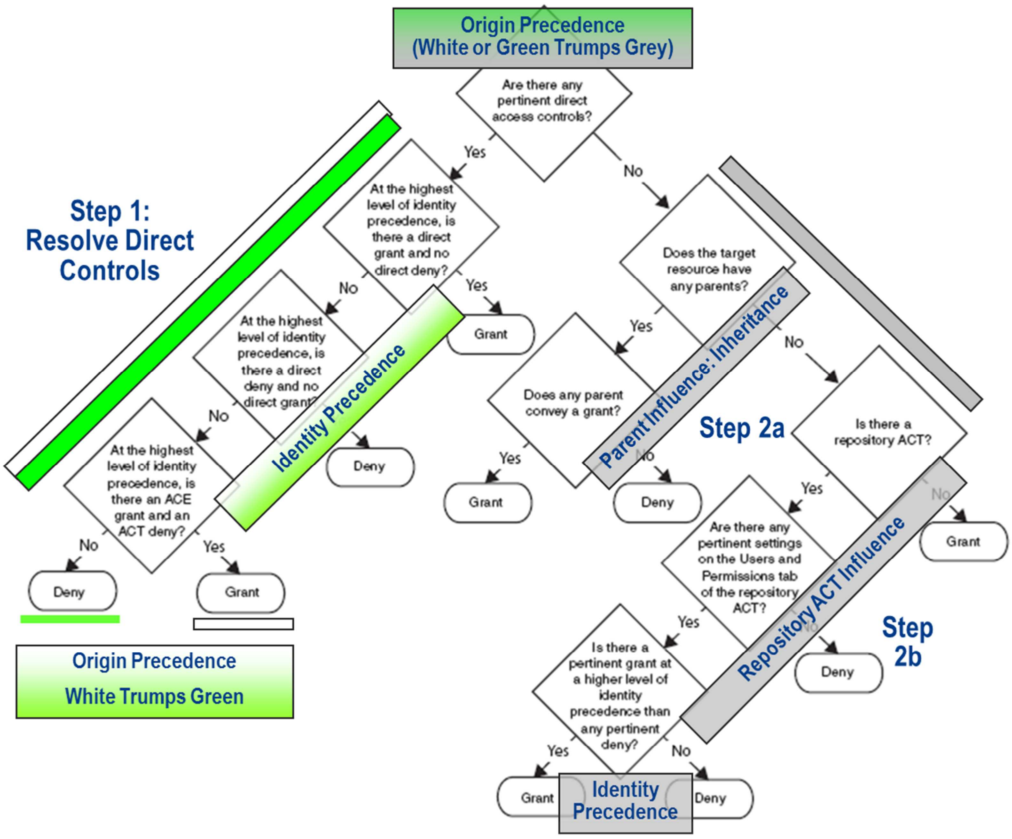
Origin Precedence
(White or Green Trumps Grey)
Are there any pertinent direct access controls?
Yes
No
Step 1: Resolve Direct Controls
At the highest level of identity precedence, is there a direct grant and no direct deny?
Yes
Grant
No
At the highest level of identity precedence, is there a direct deny and no direct grant?
Deny
No
At the highest level of identity precedence, is there an ACE grant and an ACT deny?
No
Yes
Deny
Grant
Identity Precedence
Origin Precedence
White Trumps Green
Does the target resource have any parents?
Yes
No
Does any parent convey a grant?
Yes
No
Grant
Deny
Parent Influence: Inheritance
Step 2a
Is there a repository ACT?
No
Grant
Yes
Are there any pertinent settings on the Users and Permissions tab of the repository ACT?
No
Deny
Yes
Repository ACT Influence
Step 2b
Is there a pertinent grant at a higher level of identity precedence than any pertinent deny?
Yes
No
Grant
Deny
Identity Precedence

**Figure 5: Flowchart that describe how to solve conflicts in Metadata security. Pertinent Direct Access Controls are those, that are given directly to an object (and are not inherited). They can be explicit (white) and coming from a template (Green).**

It is useful to discuss how this work in words and gives some example on how to resolve conflicts since it can get very complicated very quickly.

**Origin Precedence** – when multiple permissions exist from different origins (a grant and deny for the same permission), then permissions originating from directly applied ACEs override permissions originating from directly applied custom ACTs, which override permissions that are Indirect, or "*ACE trumps custom ACT trumps Indirect*". **White trumps green trumps grey**!

**Identity Precedence -** when multiple permissions conflict due to an Identity Context containing multiple identities, Identity Precedence dictates that a permission from the Primary identity overrides a permission from any Group Identities the subject is directly a member of, which override any permissions from groups the subject is indirectly a member of. Between identities at the same precedence level, Deny supersedes Grant.
Identity precedence is used to resolve conflicting permissions of the same Origin and also to resolve conflict within the Identities of the Repository ACT.
Remember also that everyone is a member of SASUSER and PUBLIC groups. Those have always "last precedence level" (that is the reason why we give "deny" to all for those groups).

**Parent Influence**: Multiple Inheritance Rule – when inherited permissions conflict due to multiple parent objects, one parent contributing a grant and another contributing a deny, Grant supersedes Deny. A child only needs one parent to say yes to stay up past bedtime.

**Repository ACT Influence** – since there is only one Repository ACT, controls of this origin cannot conflict, however we still may have Identity based conflicts within the Repository ACT. In the case of multiple relevant Identities contributing conflicting permissions in the Repository ACT, resolution is done using Identity Precedence.

Evaluating for Effective Permissions can be done following two points:

1. Identify and Evaluate Direct Access Controls using Access Control Origin Precedence (ACE trumps ACT, or white trumps green), if multiple Direct Access Controls of the same Origin are in conflict – resolve using Identity Precedence.
2. Evaluate Indirect Permissions
   a. Parent object influence is checked first, all it takes is one grant!
   b. If no parent influence, Repository ACT is used. Any conflicts coming from multiple Identities in the ACT are resolved using Identity Precedence.

NOTE: Although steps 2a and 2b are discrete steps, the Authorization Manager's effective permission tab merges these such that one merely has to resolve indirect permissions, or the "grey" checkboxes, using Identity Precedence conflict resolution rules.

Here are some examples of conflicts that can be resolved with **Identity Precedence**:

- A user is in two groups at the same level. From one he/she receive a "deny" and from the other a "grant" for a specific resource. In this case the "deny" wins (remember between identities at the same level "deny" wins over "grant") (see Figure 3: A user is in two groups that gives two different permissions. Which one wins?).
- A user is in a group that gives him/her a "grant", and in a more distant group (the group is a member of another group) that gives him/her a "deny". In this case a "grant" wins (see Figure 4: A user is in two groups: GROUP1 and GROUP2. But GROUP2 is in another group (GROUP3). Which permission win in this case?).

## How ACTs help in some common cases

Here are some use cases that should make clear enough how using the rules described above will make your life much easier.

### A temporary user

A typical case you may be confronted with, is the need to add a temporary user (for example an external consultant) to your user base. He may need access to several folders.
It is a very bad idea to give explicit permission to the specific user in this case. Since you are dealing with a reduced (remember, I said this document applies to a user base of roughly 20-30 users) user base you can follow the steps described above to be able to add (and remove) temporary users. You simply add him/her to specific groups, and he will automatically have the same permissions the group have. You will not need to go and check each resource.

### A user has left the company

This is a very easy case. Since I told you to work only with groups and not directly with users is just a matter of removing a user from the groups he/she is in and you will have removed all the rights he/she has. You may even leave it in the metadata and not worrying of him/her having any right left. Although I suggest to clean up your user base regularly pruning users that are not in the company anymore.

### A user has changed department

In this case you just need to move the user from his original group to the new one. He will automatically have all the new rights of the new department. That is all you need to do.

### A number of users need special temporary grants

In this case I suggest you create a special group for the users and a special ACT with the grants for the group. Then you simply need to assign the ACT to the relevant resources. Later on you can delete the ACT and the group to remove the special permissions.

## Meta Security (Protecting ACTs)

To protect your ACTs from unwanted modification is better first to deny WM (Write Metadata) to PUBLIC. After that you can give all the rights to administrators.

## References

Dhann, S., 2001. *Harvard Referencing.* [Online]
Available at: https://education.exeter.ac.uk/dll/studyskills/harvard_referencing.htm
[Accessed 22nd August 2016].
Hopkins, P., 2008. *SAS9 Security in 6 Steps,* s.l.: SAS(R).
Jørgensen, J. & Hoffritz, C., 2013. *http://support.sas.com.* [Online]
Available at: http://support.sas.com/resources/papers/proceedings11/376-2011.pdf
[Accessed 22nd August 2016].
SAS(R), 2016a. *SAS Support - ACT Settings.* [Online]
Available at:
http://support.sas.com/documentation/cdl/en/bisecag/61133/HTML/default/viewer.htm#a003178109.htm
[Accessed 22nd August 2016].
SAS(R), 2016b. *SAS Support - Inherited settings.* [Online]
Available at:
http://support.sas.com/documentation/cdl/en/bisecag/61133/HTML/default/viewer.htm#a003178108.htm
[Accessed 22nd August 2016].
SAS(R), 2016c. *SAS Support - Explicit Settings.* [Online]
Available at:
http://support.sas.com/documentation/cdl/en/bisecag/61133/HTML/default/viewer.htm#a003209280.htm
[Accessed 22nd August 2016].