Paper SAS118-2014

Using Metadata-Bound Libraries to Authorize Access to SAS® Data

Jack Wallace, SAS Institute Inc

ABSTRACT

Have you found OS file permissions to be insufficient to tailor access controls to meet your SAS® data security requirements? Have you found metadata permissions on tables useful for restricting access to SAS data, but then discovered that SAS programmers can avoid the permissions by issuing LIBNAME statements that do not utilize the metadata? Would you like to ensure that users have access to only particular rows or columns in SAS data sets, no matter how they access the SAS data sets?

Metadata-bound libraries provide the ability to authorize access to SAS data by authenticated Metadata User and Group Identities that cannot be bypassed by SAS programmers who attempt to avoid the metadata with direct LIBNAME statements. They also provide the ability to limit the rows and columns in SAS data sets that an authenticated user is allowed to see. The authorization decision is made in the bowels of the SAS® I/O system, where it cannot be avoided when data is accessed.

INTRODUCTION

Metadata-bound libraries were first implemented in the second maintenance release of SAS® 9.3 and were enhanced in SAS® 9.4. This paper overviews the feature and discusses best practices for administering libraries bound to metadata and user experiences with bound data. It also discusses enhancements included in the first maintenance release of SAS® 9.4.

WHAT IS A TRADITIONAL METADATA LIBRARY?

You are probably familiar with traditional metadata libraries, but readers who are not familiar with them will benefit from a quick overview before being introduced to metadata-bound libraries. A traditional metadata library is a metadata object that can be used by applications to locate and assign a SAS physical library. Table metadata objects can be associated with the library objects, and column metadata objects can be associated with the tables.

The primary purpose of these objects is to allow applications to find physical data and obtain information about the data. The metadata server is a directory service for the applications. When some of these metadata-aware applications needed a way to restrict users' access to the data, data access controls were added to the objects. But access control was a secondary purpose. This model for access control has administration issues and is not a security solution for all data access.

SECURITY ADMINISTRATION ISSUES

The traditional model only supports Read, Write, Create, and Delete data access permissions. While this might seem like a complete set of permissions, the permissions are ambiguous when applied to both libraries and tables. Does Create provide permission to create a table in a library, or does it provide permission to insert a record in an existing table? Similarly, does Delete provide permission to delete a table, or does it provide permission to delete a record in a table? Since both library objects and table objects inherit permissions directly from their folder objects, the ambiguity can cause unexpected behavior.

Since the primary purpose of these traditional metadata library and table objects is to help applications and their users find and locate data, users might want to copy and arrange the objects into their own folder trees. With multiple objects locating the physical data, applying access controls to the physical data through those objects is, at best, a challenge. Moreover, in user folders, the user can set permissions on the objects at will.

SECURITY PROBLEM

While the previously mentioned issues are problematic, the biggest security problem of this model is that it is not uniformly enforced by all SAS software access to the data. Only applications and features that use the metadata (for example, the metadata LIBNAME engine) can enforce permissions on the objects when accessing the physical data. A power SAS user with access to the SAS windowing environment, SAS Enterprise Guide, or batch job submission can issue LIBNAME statements directly to the physical library and bypass all the permission restrictions in the metadata objects.

WHAT IS A METADATA-BOUND LIBRARY?

A metadata-bound library is a SAS physical library on disk (and the SAS data sets within it) that is tightly bound to metadata objects in a metadata server, so that access controls on the metadata objects determine what types of access a user or group of users has to the data. The physical library is bound to a SecuredLibrary metadata object. Each of the SAS data sets is bound to a SecuredTable metadata object. The only purpose of these metadata objects is to grant or deny access to the physical data whenever the physical data is accessed by the SAS system. These metadata objects are not used to locate and assign libraries or to determine attributes of the SAS data sets.

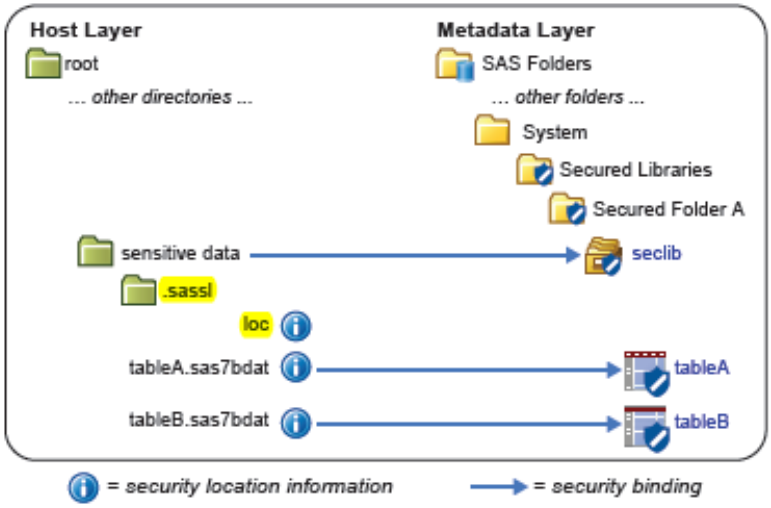


Figure 1. Depiction of a Metadata-Bound Library

Figure 1 depicts the binding of the physical data (represented on the left) with the metadata objects (shown on the right). The “sensitive data” directory is bound to the SecuredLibrary object named seclib under Secured Folder A in the /System/Secured Libraries tree. Note that the .sassl subdirectory and its loc file exist under the “sensitive data” directory to bind it to the seclib SecuredLibrary object in metadata. The loc file contains the binding information, and the .sassl directory is restricted by host permissions so that files within the subdirectory can be modified only by a host account or by a user with permissions to modify the permissions on the “sensitive data” directory. As a result, other users are prevented from tampering with the binding information. Location information in the header of each SAS data set binds the data set to its corresponding SecuredTable object.

DATA ACCESS REQUIREMENTS

The SAS I/O system allows access to metadata-bound data sets only if an authenticated identity establishes a connection to the metadata server. Access to the physical data is then **always** granted or denied in SAS based on the permissions the identity either has or inherits from the metadata object to which it is bound. You, the SAS library administrator, control the permissions that metadata identities have on the metadata objects.

If you have been granted WriteMemberMetadata access to the /System/Secured Libraries metadata folder (or some subfolder of it), then you can bind the physical libraries that you own to SecuredLibrary objects in the folder. You can set data permissions on the folder which are inherited by the SecuredLibrary objects. You can add, remove, or modify permissions that are inherited from the folder by explicitly granting or denying the permissions on each SecuredLibrary object. The SecuredTable objects are created and bound to existing data sets both when you bind the library and when new data sets are added to the library. You can add, remove, or modify permissions that are inherited from the SecuredLibrary by explicitly granting or denying the permissions on each SecuredTable object.

DATA PERMISSIONS

The data access permissions that you can set on metadata-bound libraries follow the SQL naming conventions. There are two categories of permissions: table-level and record-level.

Table-level permissions include Create Table, Alter Table, and Drop Table. If you grant an identity Create Table permission on a SecuredLibrary object, that identity can create new data sets in the bound library unless you have explicitly denied Create Permission for a SecuredTable object with that name under the SecuredLibrary object. If no SecuredTable object with that name exists, you can create one by creating a data set in the library with that name

and subsequently deleting it. However, if you want to allow the identity to replace an existing data set, then you must grant the identity Alter Table permission on the SecuredTable object to which the data set is bound, or the identity must inherit that permission through the SecuredLibrary object. Alter Table permission is also required to rename or modify the attributes of an existing data set. Finally, Drop Table permission is required to delete the SAS data set.

Record-level permissions include Select, Insert, Update, and Delete. Select permission is required to read any records in the SAS data set. Insert, Update, and Delete permissions are required to add, modify, and delete records respectively. In order to modify or delete a record, the system must first read the record. Therefore, Select permission is also required for those actions. Also note that it is possible to give an identity Create Table permission without Insert permission. In that case, the identity can define a new table and its attributes but cannot add any records to the table.

HOW DO YOU BIND AND ADMINISTER A METADATA-BOUND LIBRARY?

You use the AUTHLIB procedure in a SAS session to bind and administer a metadata-bound library. The procedure has action statements to CREATE, MODIFY, REPORT, REPAIR, and REMOVE bindings on a physical library. By default, these action statements are applied to all the data sets that are in the library. In addition, you can use the TABLES statement with each of these action statements to direct the action to specific data sets or to supply specific data set options.

In the 9.4 release of SAS Management Console, new actions on Secured Folder and SecuredLibrary objects display dialog boxes in which you can enter binding attributes. To create a new metadata-bound library, you select the **New -> SecuredLibrary** action on a Secured Folder. Actions on the SecuredLibrary object enable you to Modify, Report, or Delete and Unbind a metadata-bound library. The dialog boxes build the appropriate AUTHLIB code and submit it to the workspace server of your choosing. While these dialog boxes do not support data set-specific options by building TABLES statements, you will not need those table-specific options in most cases.

BINDING ATTRIBUTES

Obviously, you must specify the physical library that is currently bound (or that you want to bind) to the metadata. You can specify this either as a libref to an assigned physical library in the AUTHLIB procedure statements, or as a physical path to the library in the SAS Management Console dialog boxes.

You must also supply metadata-bound library passwords when creating a binding or administering an existing binding. These passwords are set on all bound data sets in the library. However, they are not used to authorize user access to the data sets. They are used by you and any other administrator of the library to authorize administrative actions with the AUTHLIB procedure. You should not share these passwords except with other administrators of the library. In addition, you should record the passwords in a secure location. SAS cannot recover them if they are lost.

Having passwords stored in the metadata and the data sets enables additional data set features. The data sets can be encrypted with SAS password encryption without the SAS code having to supply a READ password. They also enable the administrator to create views that provide restricted row and column level access to bound data sets, even if the user identity does not have Select permission to the base data sets that the view accesses. This approach is described later in the “How Do You Restrict a User’s Access to Specific Rows and Columns?” section of this paper.

The first maintenance release of SAS 9.4 supports new binding attributes on the library for encryption. You can specify the REQUIRE_ENCRYPTION option to require that all bound data sets be encrypted either with SAS password encryption or with AES encryption. When the REQUIRE_ENCRYPTION=YES attribute is set on the library, the SAS system will always encrypt the data sets bound in the library without the code specifying encryption options. The new ENCRYPTKEY attribute allows you to securely store an AES encryption key for the metadata library. The stored key is then used to encrypt and decrypt all data sets if encryption is required. If encryption is not required, the stored key is used to decrypt and encrypt pages of any data set that is AES encrypted if the code does not supply a key, and it is used to encrypt pages of a new data set if the code specifies encrypt=AES without specifying a key. Note that while the first maintenance release of SAS 9.4 is required to set these attributes in metadata, the SAS 9.4 release without maintenance will honor them. However, the second maintenance release of SAS 9.3 will not honor them.

ADMINISTRATOR ACCESS REQUIREMENTS

In order to use the AUTHLIB procedure to bind or administer a metadata-bound library, the SAS process must be executing under a host account or user that has OS file permissions on the physical library to set OS file permissions on that same library and the SAS data sets within it. This prevents a user who is not allowed to set OS file permissions from binding and setting restrictions on the files with SAS metadata.

On Windows operating systems, a user or group of users granted Full Control of the directory can set permissions on

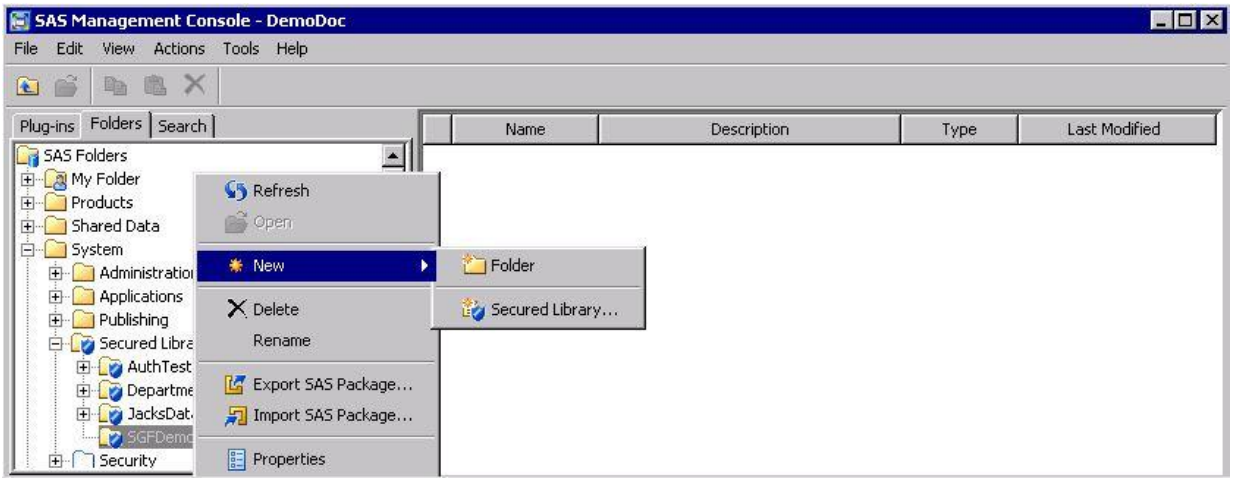
the directory and the files within it. Therefore, you can administer the metadata bindings on a Windows-based physical library if you have Full Control. With basic UNIX file and directory permissions, only the owner of the directory can set permissions on the directory or the files within it. So on UNIX systems and for the UFS directory libraries on z/OS, you must be the owner of the directory to administer metadata library bindings. If the library is a z/OS bound library, then you must have RACF Alter authority on the physical file that contains the library members.

Finally, as mentioned earlier, you must also supply the current metadata-bound library passwords when modifying, repairing, or removing the binding of an existing metadata-bound library and the data sets within it.

EXAMPLE OF BINDING A METADATA LIBRARY

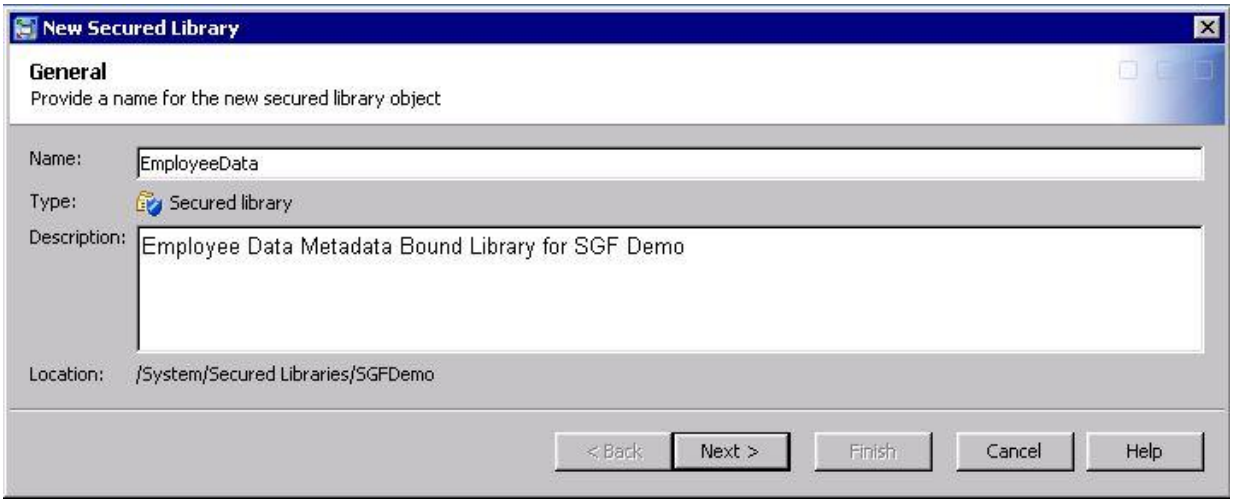
This example uses SAS Management Console as of the first maintenance release of SAS 9.4 to bind a physical library to metadata. The dialog boxes in the original SAS 9.4 release are similar, but they lack the fields to specify the new encryption attributes.

To create a new metadata-bound library in SAS Management Console, you navigate to the secured folder in the /System/Secured Libraries tree where you want to create the SecuredLibrary object, right-click the folder, and select **New**.



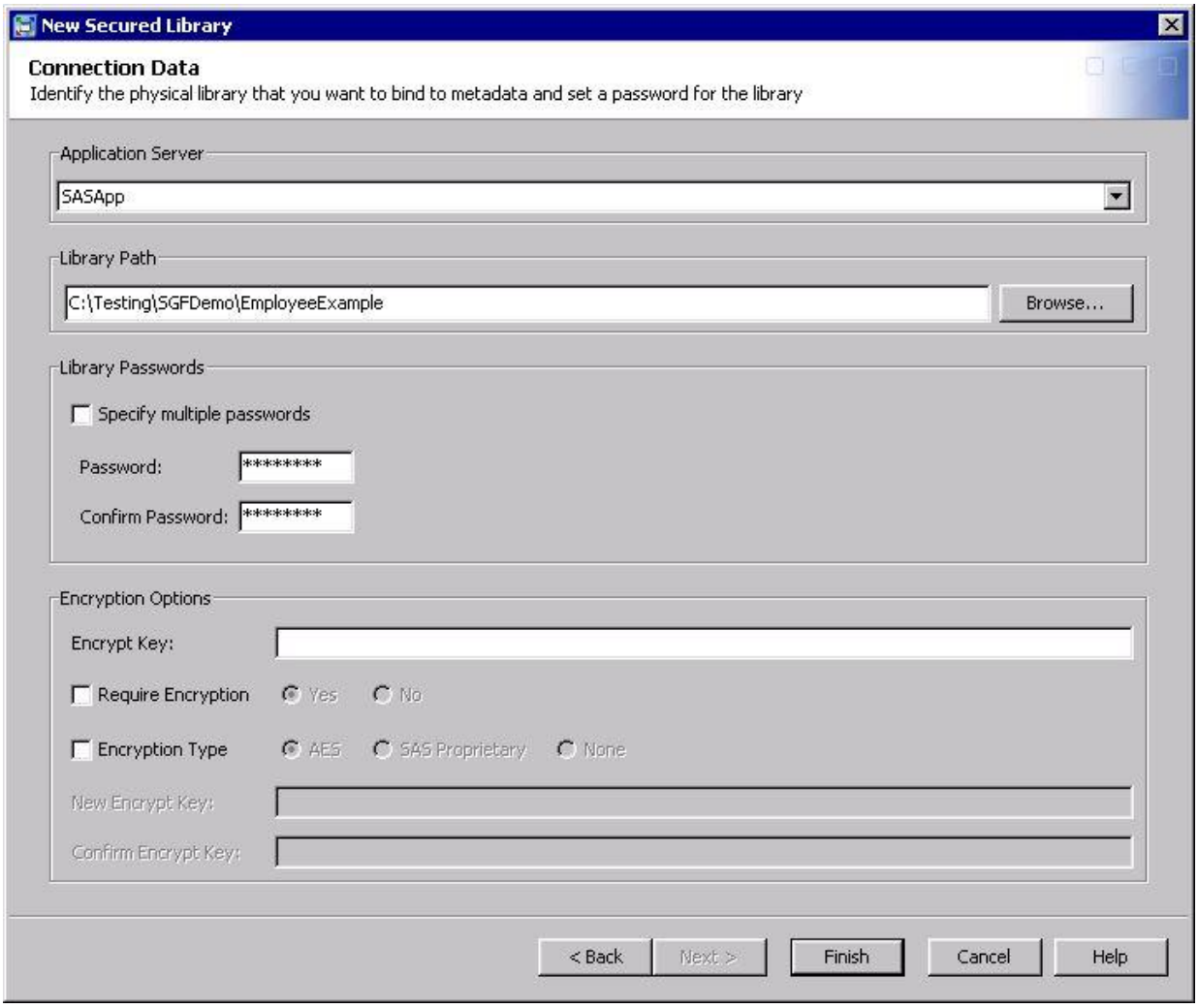
Display 1. Creating a New SecuredLibrary Object in SAS Management Console

Select **Secured Library** to bring up the New Secured Library wizard:



Display 2. Name the SecuredLibrary Object

Type a name for the SecuredLibrary object. You can also provide a description. Then select **Next**.



Display 3. Complete Binding to Physical Library on a Workspace Server

In this window, you must select the application server that contains the workspace server to which the AUTHLIB procedure code is to be submitted. After selecting the application server, you can either type the library's full physical path on the workspace server, or you can select the **Browse** button to display a browser window in which you can navigate the workspace server's file system and select the directory that contains the physical library. You must also enter and confirm a password for the metadata-bound library. Be sure to record the password in a secure location.

The encryption options are displayed only if you are using SAS Management Console from the first maintenance release of SAS 9.4 or a subsequent release. In that case, the workspace server must also be running the first maintenance release of SAS 9.4 or a subsequent release.

When you select **Finish**, the code is submitted to the workspace server.

When the code executes, it creates the SecuredLibrary object in the metadata, and it creates the .sassl directory and loc file in the file system to bind the library's physical path to the new object. It also attempts to open every existing data set in the physical library, creates a SecuredTable object for the data set under the SecuredLibrary object, and stores the location information for that object in the SAS data set.

When the code has finished executing, a confirmation window gives you the option to view the log for the submitted code:

Display 4. Completion Dialog Box

You should always view the log and examine it for errors. Errors will occur if data sets are either password protected or encrypted with an AES key and you do not provide the password or key for the data set in the New SecuredLibrary wizard. You must use the AUTHLIB procedure's MODIFY and TABLES statements to correct the errors and bind the data sets that fail to open.

You might also want to use the AUTHLIB procedure's REPORT statement to reveal any inconsistencies between the data sets in the bound library and the SecuredTable objects. As a matter of fact, you can use the REPORT statement to monitor the status of the bound library on a regular basis.

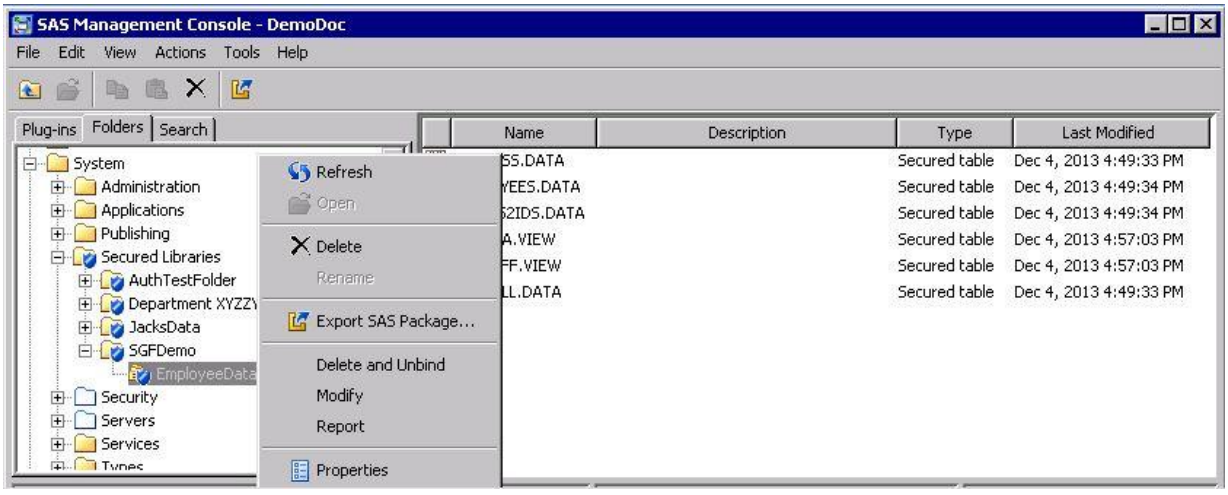
Once the library is bound to metadata, any data sets that you or your authorized users subsequently add to the physical library will create SecuredTable objects if the objects do not already exist. The data sets are bound to the objects.

HOW DO YOU SET PERMISSIONS ON METADATA-BOUND DATA?

DATA step functions can be used to programmatically set permissions on metadata data objects through access control entries (ACEs) and access control templates (ACTs). However, the easiest way to set the permissions on Secured Folder, SecuredLibrary, and SecuredTable objects is to use SAS Management Console. You can use the Authorization tab in the properties sheet of each object to do the following:

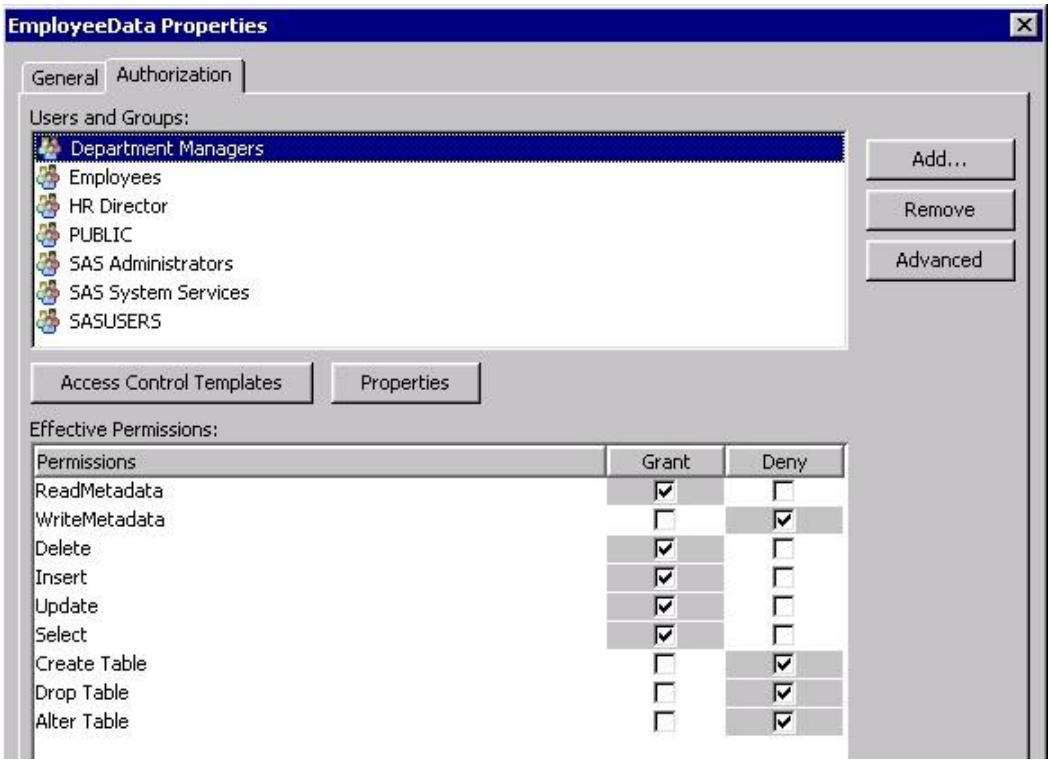
- Add or remove ACTs to or from the object.
- See which permissions have been inherited and which permissions have been explicitly set on the object.
- Add or remove associations of identities to or from the object.
- See the effective permissions each identity has on the object.
- Add or remove explicit permissions for each identity on the object.

To set permissions on a SecuredLibrary object, scroll to the SecuredLibrary object on the **Folders** navigation tab (located on the left side of the SAS Management Console window), and right-click the object to display the pop-up action menu:



Display 5. Select Properties on a SecuredLibrary Object in SAS Management Console

Select **Properties**, and then select the **Authorization** tab in the Properties window.



Display 6: A SecuredLibrary Object's Authorization Tab in SAS Management Console

The identities that are currently associated with the object are listed in the top panel. Use the buttons on the right to add or remove associations to identities. Select an identity in the top panel to see the effective permissions that the identity has on the object in the bottom panel. The table and record-level permissions are listed on the left, and for each permission, check marks appear on the right in either the **Grant** column or the **Deny** column. A gray background indicates that the permission is inherited from the parent Secured Folder object. A white background indicates that the permission is explicitly set on the object. A green background indicates that the permission is derived from an access control template on the object. If a check box's background is not white, you can set an explicit permission of Grant or Deny on the object by clicking the appropriate check box. You can remove an explicit permission by clicking the checked box when the background is white.

Setting permissions on SecuredTable objects behaves similarly when you right-click a SecuredTable object in the right panel of SAS Management Console and select the **Properties** action. In the first maintenance release of SAS 9.4, the Authorization tab contains an **Add Condition** button which can be used to filter the rows of data that an identity can read. That difference is discussed under "Permission Conditions" in the next section.

HOW DO YOU RESTRICT A USER'S ACCESS TO SPECIFIC ROWS AND COLUMNS?

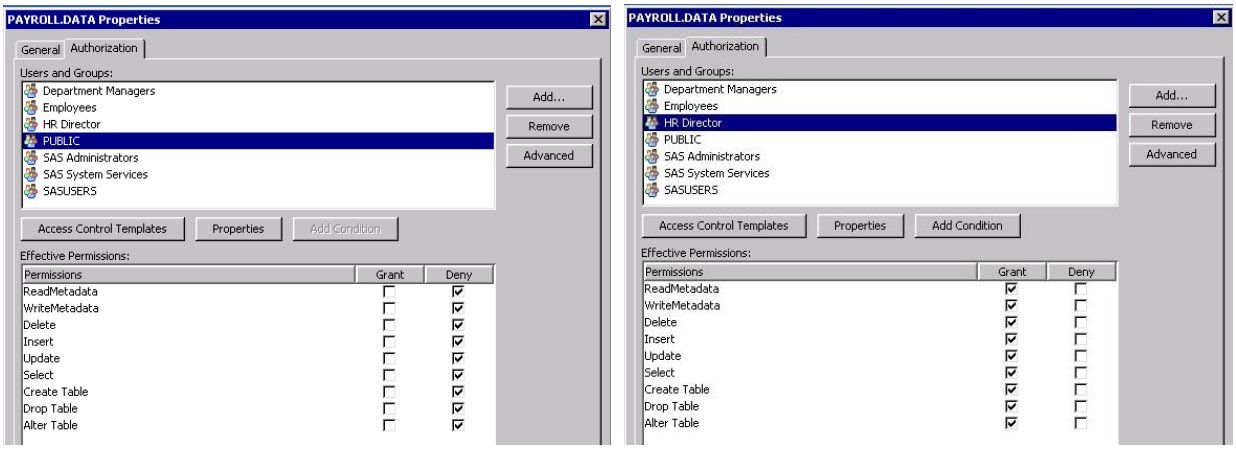
There is only one way to permit users to read some columns in a metadata-bound data set and not all columns. To do so, you deny the identity Select permission on the SecuredTable object to which the data set is bound. Then, create an SQL or DATA step view of the data set that includes only those columns from a base data set that the users are allowed to see. When defining the view, you authorize the view file to access a bound base data set by specifying the metadata-bound library password as a data set option. Then you can grant user identities Select permission on the SecuredTable object to which the view file in the library is bound.

A similar approach with view files can be used to restrict users' access to selected rows in a data set. This approach works in all SAS releases that support metadata-bound libraries. You use WHERE clauses in the SQL or DATA step view definition to restrict which rows from a base data set the view returns to the user. To keep from having to define separate views for each group of rows that require restricted access, you can use the pseudo variable _METADATA_AUTHENTICATED_USERID_ in the WHERE clause of the view definition. This variable resolves to the authenticated metadata identity's login ID when the view is accessed. You can create a user attribute data set (if you do not have one already) with a column containing the user login IDs and other user attributes such as department or role. You then define an SQL view that selects only the row from the user attribute data set that matches the authenticated user's login ID, and join it with any other base data set on other user attributes present in the joined data sets.

EXAMPLE VIEW PERMITTING ACCESS TO A SUBSET OF ROWS AND COLUMNS

In this example, a company has an Employee data set with one row per employee. The data set contains non-sensitive fields such as EmployeeId, Name, Gender, DepartmentCode, JobCode, and ManagerId. The company has other data sets containing company-sensitive information about employees (such as a Payroll data set containing salaries) and employee-sensitive data (such as an Address data set containing home contact information). All of the data sets contain an EmployeeId field to identify the employee. All of these data sets are in a metadata-bound library controlled by the HR data administrator.

The HR data administrator wants to allow any company employee with a metadata identity Select access to all the data sets in the library except Payroll and Address. He has applied an access control template to a folder higher in the Secured Libraries folder tree that grants appropriate default permissions to metadata identities. To restrict access to the Payroll and Address data sets, he needs to explicitly deny data access to the Public group on these two SecuredTable objects. Then he needs to grant the permissions he wants the HR employees to have. On the left side of the following display, the administrator has checked the **Deny** box for all data permissions for the Public group on the authorization tab for the Payroll data set. On the right side of the display, the administrator has granted all permissions to the HR director. Note that each check box has a white background, indicating that it is explicitly set.



Display 7. Payroll Has Explicit Public Denies and HR Director Grants

Now the administrator wants to allow employees access to their own data by creating a view that only selects that user's data. While he could add a LoginId field to the Employee data set, he decides not to disturb the current data structure. Instead, he creates a separate Logins2Ids data set with records containing the LoginId and EmployeeId of all employees who have a metadata identity. Then he uses the following SAS code to create an SQL view containing all the fields from the Employee data set for the user who has authenticated to the metadata server:

```
PROC SQL;
  CREATE VIEW HR.Mydata AS
        SELECT e.* FROM Logins2Ids AS l, Employees AS e
        WHERE l._METADATA_AUTHENTICATED_USERID_  = l.LoginId AND
              l.EmployeeId = e.EmployeeId;
Quit;
```

SAS Code 1. Mydata SQL View

Now the HR administrator can join this view with the other data sets that contain sensitive information and restrict what rows are returned in the joined table by attributes in the Employee data set. For example, if he wants to allow managers to see their direct report employees' sensitive data in Payroll and Address, he can use the following SAS code (in which <mblpword> is the metadata-bound library password) to create a view:

```
PROC SQL;
  CREATE VIEW HR.MyStaff AS
       SELECT m.EmployeeId AS MyId, e.Name, p.Salary, a.*
       FROM Mydata AS m,  Employee AS e, Payroll(pw=<mblpword>) as p, Address(pw=<mblpword>) AS a
       WHERE MyId=e.ManagerID AND e.EmployeeId = p.EmployeeId AND e.EmployeeId = a.EmployeeId;
Quit;
```

SAS Code 2. MyStaff SQL View

PERMISSION CONDITIONS

As an alternative to using views to restrict row access, the first maintenance release of SAS 9.4 has added support for defining conditions to apply on the Select permission. This feature is available from the Authorization tab on a SecuredTable object. Permission conditions on SecuredTable objects are similar to BI row-level permissions as implemented in information maps and row-level security as implemented for LASR tables. For metadata-bound libraries, the conditions are simply WHERE clauses that are applied to the data set retrieval when the authenticated user is granted Select permission with the condition applied.

There might be advantages to using permission conditions if you are familiar with their implementation in information maps or LASR tables. However, in this initial supporting release, you must realize that prior releases of SAS that support metadata-bound data sets will not allow access to a bound data set when the authenticated identity has a condition added to its Select permission. So you should consider using permission conditions only when all such users will be accessing the data through the first maintenance release of SAS 9.4 or subsequent releases.

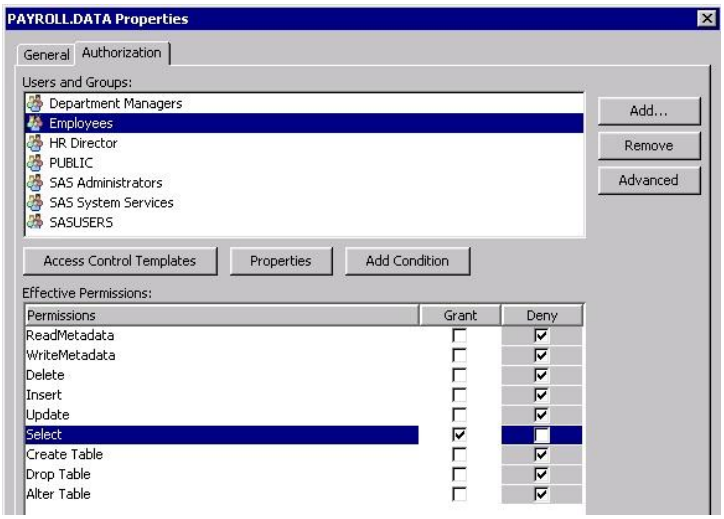
One advantage of permission conditions is the ability to use identity-driven properties in the WHERE clause. In addition to the authenticated identity's user ID (which can be used in a view), permission conditions can reference the following properties associated with the identity:

- ExternalIdentity – a value identifying the user in another system (which could have the value of the EmployeeId in our example)
- IdentityGroups – a list of groups and roles to which this identity belongs in metadata
- PersonName – the user name field of the identity's Person object

These are the most useful properties. Additional, less useful properties are described in the documentation.

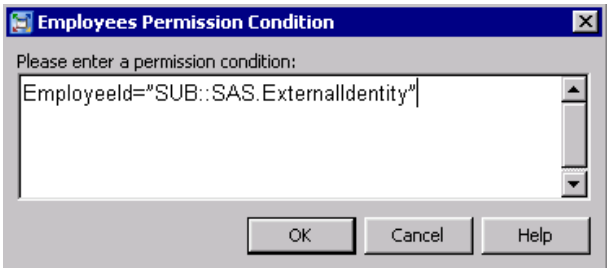
To include the values of these properties in a WHERE clause in the permission condition, the property name above is preceded by "SUB::SAS.". So, if EmployeeId is recorded as the ExternalIdentity for the Person objects of all employees registered in metadata, and if each of these Person objects is directly connected to an Employees group, then you could add the following permission condition to the Payroll and Address data sets: EmployeeId="SUB::SAS.ExternalIdentity". As a result, members of the Employees group could see only their own payroll and address information.

The screenshot below show the Authorization tab for the Payroll data set. The administrator has explicitly granted the Select permission for the Employees group.



Display 8. Payroll Employee Explicit Select

He now can select the **Add Condition** button and enter the condition in the Permission Condition text box:



Note that the permission condition is not saved and effective until the administrator selects **OK** in both this dialog box and the Properties window in which the Authorization tab is displayed. Also note that the dialog box contains a free-form text box for entering the condition, with no syntax or semantic error checking. Any errors are seen only when a user with the specified identity accesses the data set and the WHERE clauses are used to filter the data.

If you are not already familiar with other uses of permission conditions, you might find that the view method of restricting access to rows is more flexible and simpler to learn and understand. For example, in the previously described scenario, there would be no way to allow managers to see their direct reports' sensitive data without resorting to views to select the manager's direct report employees. Furthermore, mixing the use of permission conditions with the use of views (using _METADATA_AUTHENTICATED_USERID_) could produce unexpected results. For example, if you were to add the previously described permission condition for Employees to the Payroll data set, the HR.Mystaff view would contain only the manager's Payroll data. You would have to create a metadata group for managers and give that group a more relaxed permission condition to allow access to the Payroll data set.

BEST PRACTICES FOR USING METADATA-BOUND LIBRARIES

You can bind any SAS library that uses the BASE engine to metadata in the second maintenance release of SAS 9.3 and any subsequent release. However, bound data sets cannot be accessed by prior releases of SAS. Therefore, do not bind data libraries when they might still be accessed by SAS systems that have not been updated to one of these releases. Also note that there are a few more access restrictions documented in the *SAS Guide to Metadata-Bound Libraries* for each release.

To bind existing data libraries that contain password-protected or AES-encrypted data sets, you will likely need to use the AUTHLIB procedure with TABLES statements so that you can specify the current passwords or encryption keys for individual data sets. This approach will ensure that you are authorized to access the contents of the current data sets before binding them.

ORGANIZE YOUR SECUREDLIBRARIES IN METADATA, AND USE A GRANT-CENTRIC AUTHORIZATION PLAN

You can specify many layers of folders in the /System/Secured Libraries tree to group libraries that have similar business and security requirements. The only restriction is that the combined pathname of a SecuredLibrary object cannot exceed 256 characters (including the characters "/System/Secured Libraries"). You can also delegate permission administration to other administrators by granting WriteMemberMetadata access to their metadata identities at any level of the folder tree. But you should not grant that permission to anyone at any level in the tree that contains libraries whose permissions they should not control.

Data access permissions on /Systems/Secured Libraries are denied to all identities when the metadata server is first configured. You should ensure that is still true, and then grant explicit data access permissions on a secured folder in the tree only to those metadata identities that should be granted that access for all libraries that are assigned under the folder or its subfolders. Then, after you bind libraries into the folder tree, grant explicit data access permissions on the SecuredLibrary object to any metadata identities that should have those permissions and did not inherit them from the secured folder tree. Finally, grant explicit data access permissions to specific SecuredTable objects to any metadata identities that should have those permissions and did not inherit them from the SecuredLibrary object. Remember to review and repeat this last step if new data sets are created in the data library that do not already have SecuredTable objects.

In some cases, you might find that only one or two tables require you to restrict the data access of identities that do inherit that access from the SecuredLibrary or higher objects in the tree. If that is the case, then you should explicitly deny the Public group all data access permissions on the SecuredTable objects, and then explicitly grant the data access permissions to all identities that should have them on the SecuredTable objects. This will prevent unintended inheritance of permissions from parent objects; that is, it will ensure that no identity that is subsequently granted data access permissions on a parent object will inherit the permissions on the SecuredTable object when the permissions on the SecuredTable object should be denied.

USE THE SAS SYSTEM TO MANAGE THE PHYSICAL DATA (AVOID OS UTILITIES)

Only the SAS system can properly manage both the physical data and the metadata, since only the SAS system is aware of the bindings. OS utilities are unaware of both the bindings and the metadata permissions. Renames and deletes of the physical files by OS utilities do not honor the SAS metadata permissions; they honor only the OS file permissions. An OS copy utility will make an identical copy of the physical file or directory, including the bindings to metadata. Using the copy utility to back up the physical data and restore it to the same location when necessary is an acceptable use of the OS utility, but creating a second copy of the data bound to the same metadata objects for any other purpose will cause administration problems.

Consider the following three figures to understand the difference between using SAS to copy files and an OS utility to copy files.

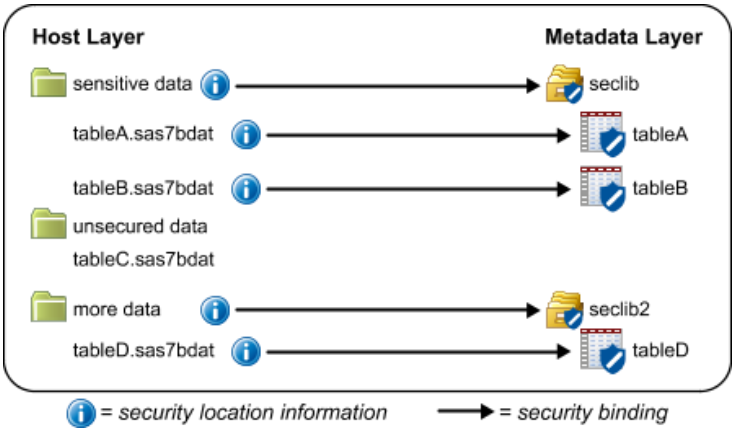


Figure 2. Figure 2: Data and Bindings Prior to Any Copy

Figure 2 depicts three physical libraries with four tables. TableC in the “unsecured data” directory has no binding to metadata. TableD in the “more data” directory is bound to a SecuredTable object under the seclib2 SecuredLibrary object. The following figure shows what happens when an OS copy utility is used to copy both tableC and tableD to the “sensitive data” directory that is bound to the seclib SecuredLibrary object. An OS copy utility does not know about SAS metadata bindings and just copies them as part of the physical file, as depicted here:

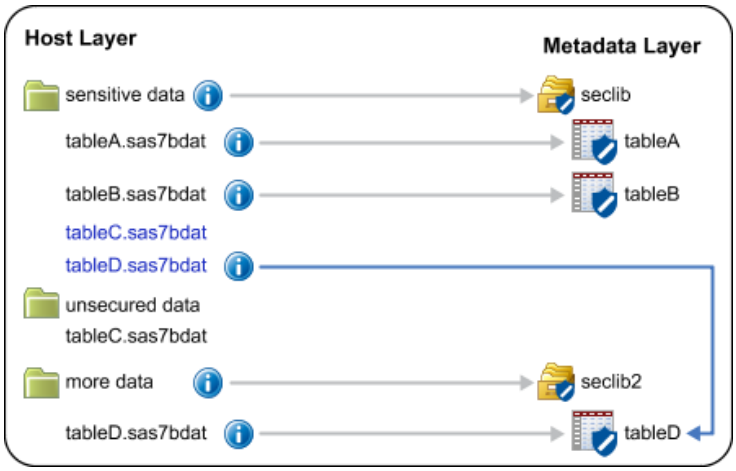


Figure 3. Data and Bindings After Copying TableC and TableD to “Sensitive Data” with OS Copy Utility

The only permissions that are validated by the OS copy utility are the OS file permissions. Notice that the copied tableC is still not bound even though it resides in a metadata-bound library. The new TableD, on the other hand, remains bound to the original SecuredTable object under seclib2. Access to the new tableC has no permission checking, while access to the new tableD is subject to the authenticated user’s data access permissions through the original tableD SecuredTable object.

Having two physical tables using the same SecuredTable object can cause administration problems. If the administrator for the “more data” library (bound to the seclib2 SecuredLibrary object) modifies or removes the binding, the new tableD data set in the “sensitive data” directory will be inaccessible.

A SAS copy (performed using the COPY procedure or other SAS utility) knows about SAS metadata bindings and completes authorization checks and metadata bindings, as shown in this figure:

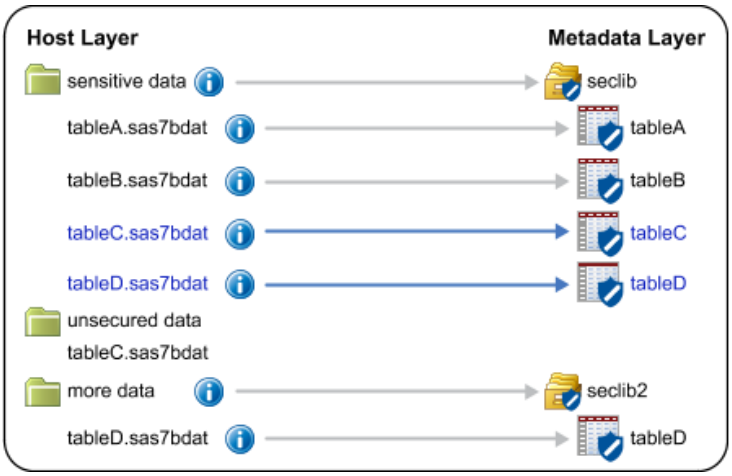


Figure 4. Data and Bindings After Copying TableC and TableD to "Sensitive Data" with COPY Procedure

The COPY procedure in SAS first validates that the authenticated user has Select permission to read any bound input data set. Since tableD is bound to the tableD SecuredTable object under seclib2, it authorizes the user's access according to the Select permission on that object. Since tableC is not bound, it can be copied without an authorization check (though if it were password protected, the copy might still fail, as discussed later).

Now the COPY procedure must validate that the authenticated user has permission to create these tables in the "sensitive data" directory that is bound to the seclib SecuredLibrary object. Since there are no SecuredTable objects named tableC and tableD under the seclib object, the user's permission to create tables in the seclib object is validated. New SecuredTable objects, tableC and tableD, are created under the seclib object, and the copied data sets in the "sensitive data" directory are bound to the new objects. If the tableC and tableD SecuredTable objects pre-exist under the seclib object, then the permission to create each data set is validated through the existing SecuredTable object with the same name.

The COPY procedure preserves passwords when copying password-protected data sets from unbound libraries to other unbound libraries. The copied data is still protected by the original password. But when data sets are copied to a metadata-bound library, the passwords must match the metadata-bound library passwords or the copy will fail. Before you can copy a password-protected file in an unbound library to a metadata-bound library, you must use either the DATASETS procedure or the AUTHLIB procedure to remove the passwords or to modify the passwords to match.

When copying a metadata-bound data set to another data library, the COPY procedure does not preserve the metadata-bound library passwords. The procedure has already authorized that the user has Select access to read the data in the metadata-bound data set and therefore is authorized to do anything with the data, including creating a copy. If the copy is being made to another metadata-bound library, then the data will be protected by the permissions in that library and will get that library's passwords. If the copy is made to an unbound library, the copy is no longer protected by passwords.

KEEP ALL DATA SETS IN A METADATA-BOUND LIBRARY BOUND TO SECUREDTABLE OBJECTS UNDER THE SECUREDLIBRARY TO WHICH THE LIBRARY IS BOUND

Administration problems can occur when a physical library that is bound to a SecuredLibrary contains data sets that are not bound to SecuredTables under that SecuredLibrary. This situation can occur in two circumstances:

- When OS copy utilities copy the data set files, as described previously.
- If data sets fail to open when an AUTHLIB procedure is binding or modifying the binding of the physical library and its data sets. This can occur, for example, if the data set is being accessed by another SAS process. For this reason, you should use AUTHLIB to set or modify bindings only when no other SAS session might be accessing any data sets in the physical library.

When a data set fails to open, it is not modified, although other data sets and the metadata attributes of the library can be modified. When this happens, the MODIFY statement can be used with the TABLESONLY option and TABLES statements to provide the missing passwords or encryption keys, open the data sets, and bind and apply the metadata-bound library passwords or encryption options to the data set.

When data sets in a bound library are not bound or are bound to SecuredTable objects under a different SecuredLibrary object, SAS programs that access those data sets will get notes and warnings printed to the log. These are meant to inform the user and you that action should be taken to make the data sets conform to this best practice.

Another way to discover data sets that do not conform is to use the Report action on the SecuredLibrary object in SAS Management Console (or run the REPORT statement of the AUTHLIB procedure). This feature reports all data sets that share similar binding attributes in report groups, with the first reported group being those data sets that are properly bound and conform to this best practice.

Finally, you can set up an Audit logger to report deviations from this best practice to a file, database, or system logger.

IF YOU USE AES ENCRYPTION, USE THE SAME KEY THAT IS RECORDED IN THE METADATA-BOUND LIBRARY

In SAS 9.4, SAS programs accessing AES-encrypted data must provide the encryption key to open the data set unless the data set is metadata bound and the key for the metadata-bound library is recorded in metadata. Without a recorded key in metadata, you would have the problem of securing any code that specifies the key. With a recorded key, if encryption is required on the metadata-bound library, all data sets bound in the library will be AES encrypted and decrypted with the recorded key whether or not encryption is specified in the SAS code. If encryption is not required, then any program that creates or replaces a bound data set and specifies ENCRYPT=AES without supplying an ENCRYPTKEY option will be encrypted with the recorded key. Data sets that are bound to the library and AES encrypted will be decrypted with the recorded key if no other key is supplied when the data set is opened.

When you record the key, you should still maintain a record of the key value in a secure location. If you remove the binding of the data library (either intentionally or by mistake) without removing the AES encryption of the data sets, the key will be needed to recover the data. For this reason, once you have bound a library to metadata and recorded a key, you should not intentionally remove the binding unless you also remove the AES encryption from the data sets.

PERMIT ALL DATA ACCESS IN TRADITIONAL LIBRARIES THAT REFERENCE METADATA-BOUND LIBRARIES

There is no reason to maintain multiple sets of data access controls to data. Doing so will only cause added confusion for you and your users. You might still want to set ReadMetadata controls on the traditional objects to control who can navigate to the data through metadata-aware applications. But the data access controls in the metadata-bound library objects should be used to control all identities' access to the data.

CONCLUSION

You can use metadata-bound libraries to tailor access controls for SAS data sets in the library with finer-grained privileges than is possible with OS file permissions. Unlike the permissions set on traditional folder and table objects, the access controls on metadata-bound data are always enforced when the SAS system is used to access the data. However, you must understand that the metadata objects and the physical files should not be manipulated independently by metadata utilities or OS utilities. Doing so can prevent access to the data and require you to take remedial action to repair the data bindings.

REFERENCES

SAS Institute Inc. 2013. SAS® 9.4 Guide to Metadata-Bound Libraries. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author would like to acknowledge Rose Howell, whose constant devotion to testing the code and providing feedback on the implementation and documentation has been instrumental to the development of metadata-bound Libraries. Many other people have contributed as well, but Rose has been with the development from the early prototypical implementation.

RECOMMENDED READING

- *SAS® 9.4 Guide to Metadata-Bound Libraries*

- *SAS® 9.4 Intelligence Platform: Security Administration Guide*
- *SAS® 9.4 Intelligence Platform: System Administration Guide*

CONTACT INFORMATION

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

Jack Wallace
100 SAS Campus Dr.
Cary, NC 27513
SAS Institute Inc.
Jack.Wallace@sas.com
http://www.sas.com