THE POWER TO KNOW®

SAS® 9.4 Guide to Metadata-Bound Libraries

Second Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® 9.4 Guide to Metadata-Bound Libraries, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.4 Guide to Metadata-Bound Libraries, Second Edition

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

July 2015

Contents

SAS 9.4 Guide to Metadata-Bound Libraries, Second Edition

Audience

This document is intended for administrators who want SAS to always enforce its metadata-layer permission requirements before providing access to SAS data. Metadata-bound libraries provide enhanced protection for Base SAS data (SAS data sets and SAS views).

In general, only administrators who set up and maintain metadata-bound libraries need to know the information that this document contains. However, in order to access metadata-bound data, a connection to the metadata server is required. So a user who makes a direct request (for example, through a LIBNAME statement) does have to facilitate that connection. See "Metadata Server Connection Options" on page 51.

Requirements

Administration of metadata-bound libraries requires Base SAS, a SAS Metadata Server, and SAS Management Console.

What's New in Metadata-Bound Libraries in SAS 9.4

Overview

The following new features and enhancements affect metadata-bound libraries in SAS 9.4:

- New SAS Management Console features simplify administration.
- A new automatic process simplifies certain administrative interactions with encrypted tables.
- Effective with the first maintenance release for SAS 9.4, you can require the tables in a library to be encrypted. You can also store the AES encryption key for a library's data sets in the library's metadata.
- The REPAIR statement's DELETE LOCATION action is now a production feature.
- Effective with the third maintenance release for SAS 9.4, replaced metadata-bound library passwords and encryption keys are retained in metadata until all tables are successfully modified or the administrator explicitly purges them.

SAS Management Console

The following new features reduce the need to write SAS code to administer metadata-bound libraries. From within a `/System/Secured Libraries` branch on the **Folders** tab in SAS Management Console, you can perform the following tasks:

- bind a physical library and its contents to metadata
- change the password of a metadata-bound library
- unbind a metadata-bound library and delete the corresponding metadata objects
- require that the tables in a library be encrypted (available in the first maintenance release for SAS 9.4)
- validate a library to identify any discrepancies related to metadata bindings (for example, missing or mismatched physical tables, security location information, or metadata objects)
- store or modify an AES encryption key in a library's metadata (available in the first maintenance release for SAS 9.4)

- specify permission conditions that give users access to some but not all of the data within a physical table (available in the first maintenance release for SAS 9.4)
- specify whether a library's passwords and encryption keys are to be retained in metadata or automatically purged if all tables in the library are successfully modified to use the newer credentials (available in the third maintenance release for SAS 9.4)

Note: Unlike most actions in SAS Management Console, the actions that are described in the first four items in the preceding list affect not only metadata, but also the corresponding physical data. All of the actions build SAS code and execute it in a workspace server.

Encryption

- The process for modifying passwords for a metadata-bound library that contains encrypted tables has been simplified. A copy-in-place approach is automatically used when necessary to accomplish a task. See "Making Security-Related Changes to an Encrypted Table" on page 47.
- You can use the TABLES statement of the AUTHLIB procedure to supply a key to use in AES encryption of metadata-bound Base SAS libraries. See "Using AES Encryption with Metadata-Bound Libraries" on page 47.
- Effective with the first maintenance release for SAS 9.4, you can force encryption by specifying REQUIRE_ENCRYPTION=YES when you create or modify a metadata-bound library. By requiring some form of encryption for all tables within a metadata-bound library, you increase security. See CREATE Statement and MODIFY Statement.
- Effective with the first maintenance release for SAS 9.4, you can use ENCRYPTKEY= to store an AES encryption key in metadata when you create or modify a metadata-bound library. The stored key is used to attempt to open the library's AES-encrypted tables when no key is supplied by the user. The stored key is used to encrypt data sets in the following cases: when encryption is required, and when AES encryption is specified in SAS code but no key is supplied.

REPAIR Statement

- The REPAIR statement of the AUTHLIB procedure no longer supports password modification. See "REPAIR Statement" on page 89.
- The REPAIR DELETE LOCATION action of the AUTHLIB procedure is now a production feature. See "REPAIR Statement" on page 89.

PURGE Statement and PURGE= Option

In the third maintenance release for SAS 9.4, a new statement and a new option for the MODIFY statement were added to the AUTHLIB procedure.

- The PURGE statement removes any retained metadata-bound library credentials older than a given date of replacement.
- The MODIFY statement has a PURGE= option that automatically removes all retained metadata-bound library credentials if all tables in the library are successfully modified to use the newer credentials.

Accessibility

For information about the accessibility of any of the products mentioned in this document, see the usage documentation for that product.

Chapter 1
Overview of Metadata-Bound Libraries

What is a Metadata-Bound Library?

A metadata-bound library is a physical library that is tied to a corresponding metadata object. Creating a metadata-bound library generates a new metadata object and binds the physical library to that object.

Each metadata-bound library has information in its directory structure that points to a specific metadata object (a secured library object). Similarly, each physical table within a metadata-bound library has information in its header that points to a specific metadata object (a secured table object). These pointers create security bindings between the physical data and its corresponding metadata objects. The bindings ensure that SAS universally enforces metadata-layer permission requirements for the physical data—regardless of how a user requests access from SAS.

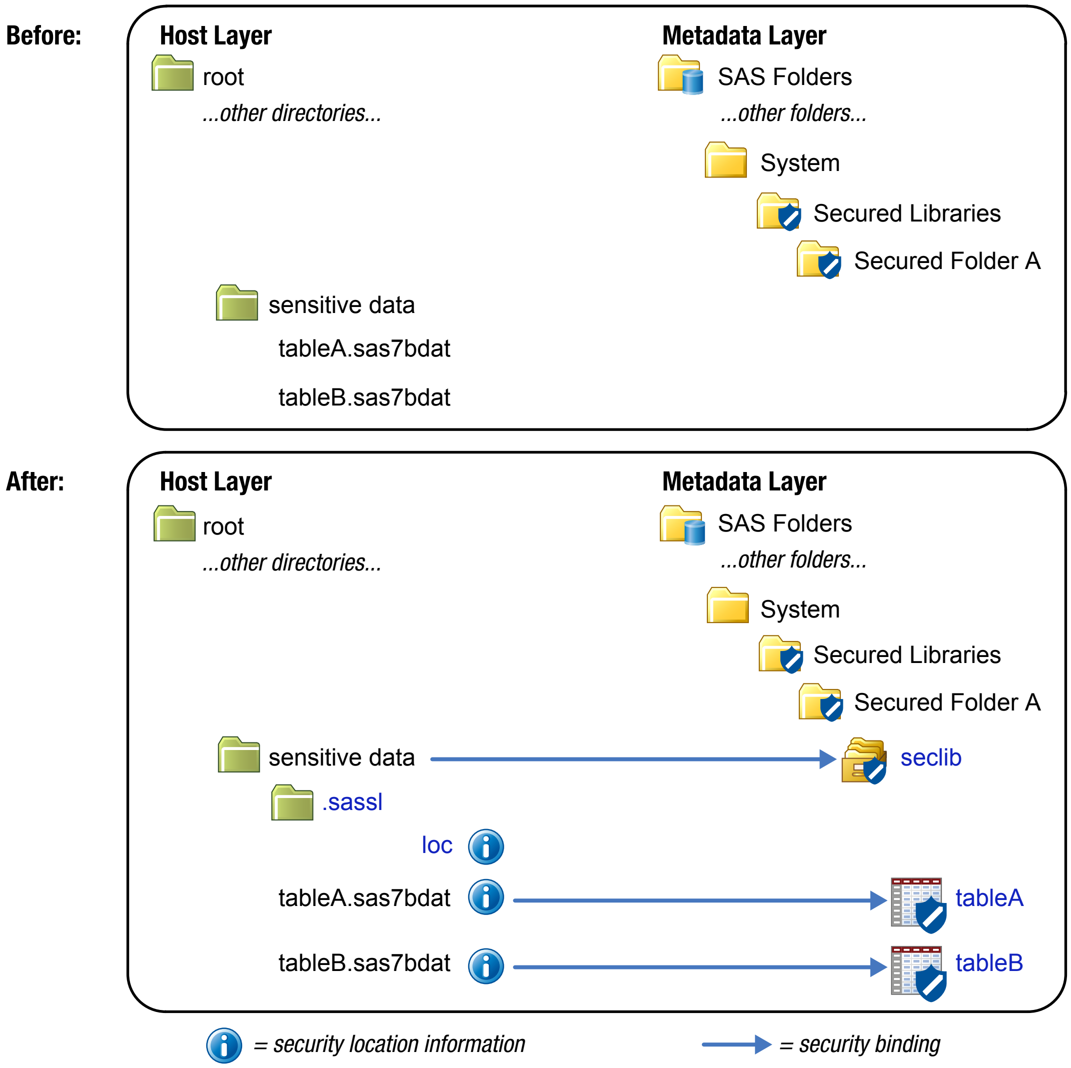
Access from SAS to metadata-bound data is provided only if all of the following conditions are met:

- The requesting user can connect to the metadata server in which the corresponding metadata objects are stored.
- The requesting user’s metadata identity has all required metadata-layer effective permissions for the requested action.
- The host identity with which the data is retrieved has all required host-layer access to the data.

Depiction of a Metadata-Bound Library

The following figure depicts the metadata objects and physical security information that are generated when you bind a library to metadata. The "Before" section shows the initial state and the "After" section shows the security location information, bindings, and metadata objects that are generated.

Figure 1.1 *Depiction of a Metadata-Bound Library*



Here are some key points about the "After" section of the preceding figure:

- The physical data includes references to corresponding objects within a SAS metadata repository.
  - For a physical library, the security information consists of a subdirectory and file. The corresponding metadata object is called a secured library object. In the figure, *seclib* is the secured library object that corresponds to the physical metadata-bound library called *sensitive data*.

    z/OS Specifics

On z/OS, the security information for a UNIX file system (UFS) library is stored as described in the preceding figure. However, the security information for a z/OS direct-access bound library is instead stored within the bound library data set itself. For this reason, z/OS sites that choose to use metadata-bound libraries might prefer the z/OS direct-access bound library implementation to the UFS library implementation. z/OS sequential-access bound libraries cannot be bound to metadata.

- For a physical table, the security information consists of information in the header. The corresponding metadata object is called a secured table object. In the figure, *tableA* and *tableB* are secured table objects that correspond to the physical metadata-bound tables *tableA.sas7bdat* and *tableB.sas7bdat*.

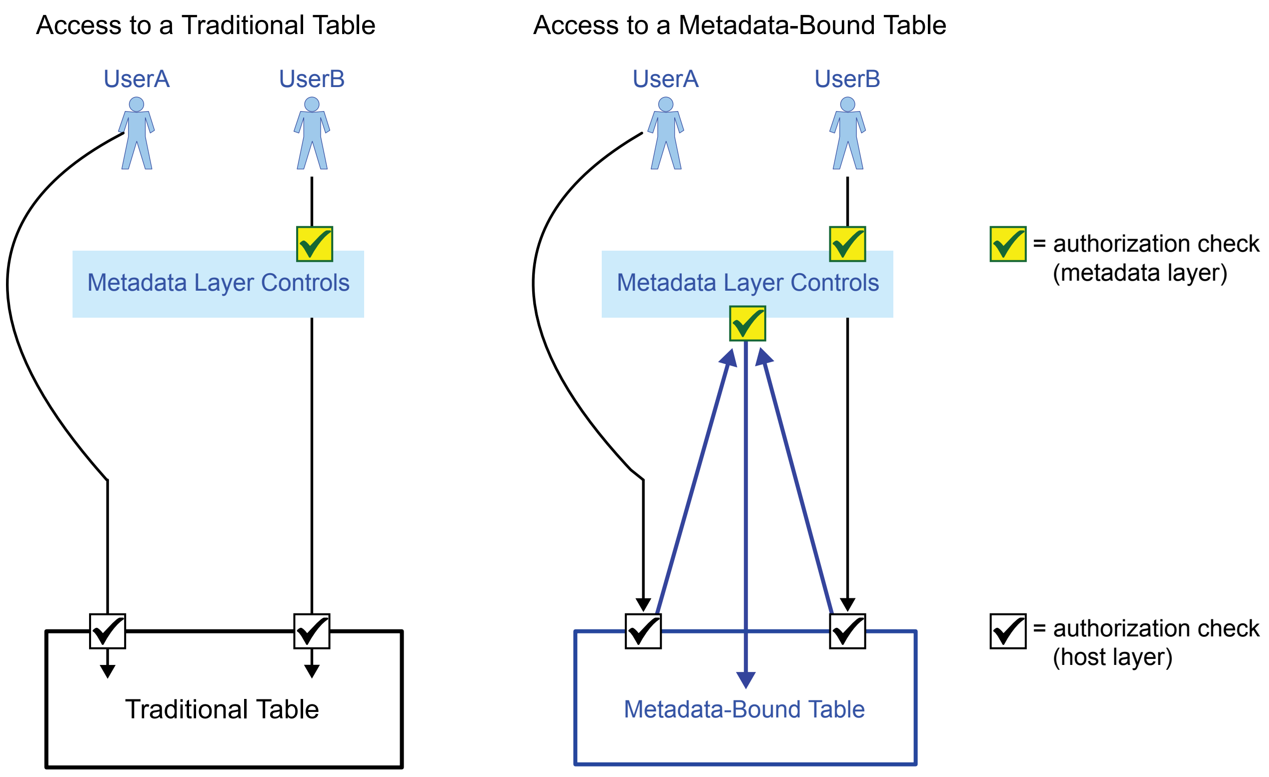
- Each security binding causes all access from SAS to be subject to the requesting user's effective metadata-layer permissions on the relevant corresponding metadata object.

Note: The figure assumes that the physical data is initially unprotected. If one or more of the physical tables already had a different password, the presence of that password would prevent that table from being affected by the bind action.

Authorization Model for Metadata-Bound Tables

The following figure depicts the authorization model for a traditional table and a metadata-bound table. In both cases, UserA references the target data directly (for example, through a LIBNAME statement) and UserB requests the target data through a client that uses metadata to locate data (for example, SAS Web Report Studio).

Figure 1.2 *Authorization Checks (by Data Type and Access Method)*


Access to a Traditional Table
UserA
UserB
Metadata Layer Controls
Traditional Table
Access to a Metadata-Bound Table
UserA
UserB
Metadata Layer Controls
Metadata-Bound Table
= authorization check (metadata layer)
= authorization check (host layer)

The preceding figure depicts the following key difference:

- When accessing a traditional table, a user can bypass metadata-layer controls by making a direct request.
- When accessing a metadata-bound table, a user cannot completely bypass metadata-layer controls. Even on a direct request, UserA is always subject to a metadata-layer permissions check before accessing SAS data from SAS.

For the metadata-bound table, the upwards-facing arrows are caused by the physical data's security binding. For each metadata-bound table, information within the table header identifies a corresponding metadata object (a secured table object). Metadata-layer permissions on each secured table object affect access from SAS to the corresponding physical table.

For the metadata-bound table, UserB is subject to two metadata-layer authorization checks against two different metadata objects.

- The first check is against a traditional table object (for example, verifying that UserB has the ReadMetadata permission).
- The second check is against a secured table object (for example, verifying that UserB has the Select permission).

TIP In the SAS metadata, traditional table objects and secured table objects are distinct and independent types of objects. See "Object Creation, Location, and Inheritance" on page 48.

Here are some additional details about the preceding figure:

- The requesting users do not supply library or table passwords.
- The metadata-layer authorization checks are against the metadata identity of the requesting user. The host-layer authorization checks are against the identity of the SAS process that retrieves the data.
- The figure addresses access to SAS data from SAS, not interaction through host commands.
- The figure is conceptual, simplified, and abstracted. It is not intended as a detailed technical specification.

See Also

- "Identity in Authorization Evaluations" on page 43
- "About Access to SAS Data" in *SAS Intelligence Platform: Security Administration Guide*

Benefits of Metadata-Bound Libraries

The benefits of metadata-bound libraries are as follows:

- Metadata-bound libraries can provide seamless, secure access to SAS data.
- Metadata-bound libraries offer more robust protection than do other metadata-based approaches to access control. Because enforcement for a metadata-bound library originates from the physical data, that enforcement occurs regardless of whether an access request from SAS is mediated by metadata (for example, from SAS Web

Report Studio) or direct (for example, from a LIBNAME statement that is submitted in SAS Enterprise Guide).

- The protection that metadata-bound libraries provide is persistent. Protections for a metadata-bound table apply to any other instances of a physical table with the same name in the same physical library. In other words, permissions that you set in the metadata survive activities that affect the underlying physical table. For example, the protections remain in place after you re-create or replace the underlying physical table.

Limitations of Metadata-Bound Libraries

The limitations of metadata-bound libraries are as follows:

- Only Base SAS data—SAS tables (data sets) and SAS views—can be bound to metadata. In the current release, you can create metadata-bound libraries for only data that is processed by the BASE engine.
- Concatenated libraries or temporary libraries cannot be bound to metadata. However, metadata-bound libraries can participate in a library concatenation.
- Binding data to metadata does not prevent the use of operating system commands against files and directories. For example, a user who has Write access to an operating system directory (in order to create physical tables) can use host commands to delete and replace files within that directory. Such commands operate independently of any metadata binding. However, replacement of files through operating system commands is detected and audited. See “Auditing for Metadata-Bound Libraries” on page 45.
- In the current release, metadata-bound libraries don’t support column-level permissions. However, you can create views that subset data by columns or rows, and then set permissions to specify who can access each view. Dynamic row-level filtering based on each requesting user’s authenticated user ID is supported. See “Providing Fine-Grained Access Using Views” on page 33.
- Advanced tasks (repairing a metadata-bound library or performing actions on only certain tables within a metadata-bound library) are not supported in SAS Management Console. You can use SAS code to perform these tasks.
- Clients that use metadata to locate data can’t use secured library objects or secured table objects for that purpose. To support access from such clients, each metadata-bound library must also be registered in metadata as a traditional library object.

  Note: The metadata objects that serve as bind targets for physical data are distinct from and independent of the metadata objects that are used to register physical data. For example, a physical library can be bound to a secured library object without also being registered as a traditional library object. Similarly, tables that are within a metadata-bound library (and bound to corresponding secured table objects) might or might not also be registered as traditional table objects. There are no associations in metadata between secured library objects and traditional library objects. See “Object Creation, Location, and Inheritance” on page 48.

  Note: In the SAS metadata, a secured library object is functional only if it exists within the `/System/Secured Libraries` branch of a repository. You can’t bind physical libraries to secured library objects that are in other metadata locations. You can create subfolders within a `/System/Secured Libraries` branch. In the SAS metadata, a secured table object can exist only within a secured library object.

The following additional limitations affect the availability of metadata-bound data:

- Access to metadata-bound tables is not supported in any release prior to the second maintenance release of SAS 9.3.

  z/OS Specifics
  For a z/OS direct-access bound library that is bound to metadata, this constraint is slightly broader: neither the library nor any of its members can be accessed by earlier releases of SAS.

  Note: An exception is that access to metadata-bound tables through a SAS/SHARE server is available to earlier clients, if SQL statements are passed to the server and the server is in the second maintenance release of SAS 9.3 (or later).

- For the SAS OLE DB Local Data Provider, access to metadata-bound tables is not supported. The SAS OLE DB Local Data Provider uses a specialized stand-alone engine that provides access to data sets from external programs without running SAS.
- For the SAS/SHARE Data Provider, access to metadata-bound tables through a SAS/SHARE server is available only if SQL statements are passed to the server. The SAS/SHARE Data Provider is one of the SAS Providers for OLE DB.
- For the SAS/IntrNet Application Dispatcher, access to metadata-bound tables is supported only if the application server runs with AUTH=HOST.

Who Should Use Metadata-Bound Libraries?

As with any other security-related decision, a decision about whether to use metadata-bound libraries involves weighing the benefits of enhanced protection against increased administrative effort and complexity. This topic is intended to help you make a decision that is appropriate for your resources, environment, and security goals.

If all of the following circumstances exist, it makes sense to consider using metadata-bound libraries:

- You have SAS data sets that require a high level of security, with access distinctions at the user or group level.
- You are running (or planning to run) a SAS Metadata Server in which your users are registered.
- You have not already met your security requirements through a combination of physical layer (operating system) separation and customized configuration of your SAS servers.

The following prerequisite knowledge is essential for successful use of metadata-bound libraries:

- You have a basic understanding of the SAS metadata environment, including its authorization system.
- You know how to create folders and set permissions in SAS Management Console.
- You have read and understood at least the first two chapters of this document.

The following additional factors should be considered in a decision about whether to use metadata-bound libraries:

- If your metadata promotion strategy does not maintain a separate set of physical data for each deployment level (for example, development, test, and production), significant additional administrative complexity is involved (compared to using secured libraries against a single set of physical data).
- Recovering from actions that inadvertently disrupt coordination between the physical data and its corresponding metadata objects can be complex.
- Any batch processing against metadata-bound data requires that the metadata server is available and that the requesting user can connect to it.

Note: In working with metadata-bound libraries, it is also useful to know how to write and submit SAS code.

Chapter 2

Implementation of Metadata-Bound Libraries

Binding Data to Metadata

Overview

The following list outlines the process for setting up a metadata-bound library:

1. Use SAS Management Console to identify or create an appropriately secured folder for the data.
2. Use either SAS Management Console or SAS code to bind the physical library to metadata.

TIP Binding a physical library introduces additional constraints on access, so it is a good practice to review existing access patterns before you begin. For help with resolving any unanticipated disruptions in end-user access, see “Facilitate End-User Access” on page 55.

Note: If you want to support access from clients that use metadata in order to locate data, make sure that the data also has a traditional registration in metadata. See “Traditional Registration” on page 15.

Requirements

In order to bind a physical library to metadata, the following requirements must be met:

- The workspace server (or SAS session) that makes the bind request must have host-layer control of the target library. This ensures that only users who have host control can bind a physical library to metadata. For host-specific details, see “Requirement for Host-Layer Control” on page 42.
- The workspace server (or SAS session) that makes the bind request must connect to the metadata server as an identity that has the ReadMetadata and WriteMemberMetadata permissions to the target secured data folder.

Preparation

This introductory demonstration limits access to a library that contains tables copied from the SASHELP library. After you complete the steps that are in the preparation section, use either the graphical user interface (GUI) method or the SAS code method to bind the data.

1. In the operating system, create a directory called **`test`**. Copy some of the tables from your SASHELP directory into the **`test`** directory.

   TIP By default, SASHELP is in your SASHOME directory, under **`SASFoundation\<`*`version`*`>\core\`**.

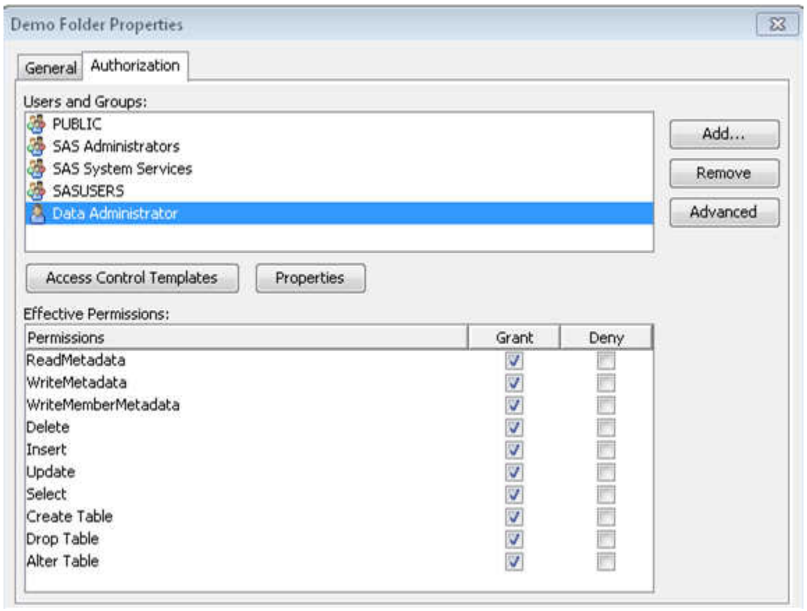
2. Create an appropriately secured metadata location.

   a. Log on to SAS Management Console as someone who has the ReadMetadata and WriteMemberMetadata permissions on the **`/System/Secured Libraries`** folder. In the standard configuration, only members of the SAS Administrators group (and unrestricted users) have the necessary access.

   b. On the **Folders** tab, navigate to **SAS Folders** ⇨ **System** ⇨ **Secured Libraries**. Add a new folder called **`Demo Folder`**.

   c. On the new folder's **Authorization** tab, adjust access. As a simplified introductory example, give yourself exclusive access to the data. One way to do this is by adding explicit controls as follows:

   - In the **Users and Groups** list box, select the PUBLIC group and explicitly deny all permissions for that group.
   - Add yourself to the tab (click the **Add** button next to the **Users and Groups** list box) and explicitly grant all permissions to yourself.



Permissions	Grant	Deny
ReadMetadata	✓	
WriteMetadata	✓	
WriteMemberMetadata	✓	
Delete	✓	
Insert	✓	
Update	✓	
Select	✓	
Create Table	✓	
Drop Table	✓	
Alter Table	✓	

TIP Read access to metadata-bound data is governed by the Select permission.

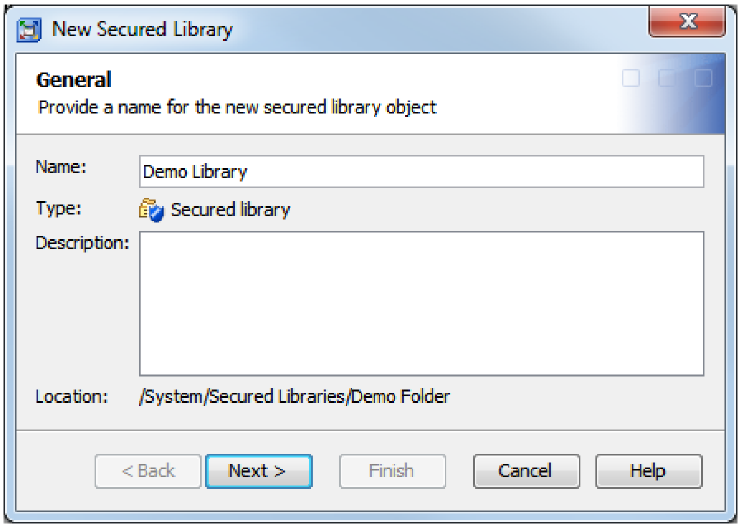
TIP In SAS Management Console, an explicit setting has a white background color (not gray or green).

TIP In practice, it would be a good idea to also apply the SAS Administrators Settings ACT (access control template).

GUI Method

To bind the physical library (your `test` directory) to metadata:

1. In SAS Management Console, right-click the `/System/Secured Libraries/Demo Folder` folder and select **New** ⇨ **Secured Library**.
2. On the **General** page of the New Secured Library wizard, enter *Demo Library* as the name for the object. Click **Next**.



New Secured Library

General

Provide a name for the new secured library object

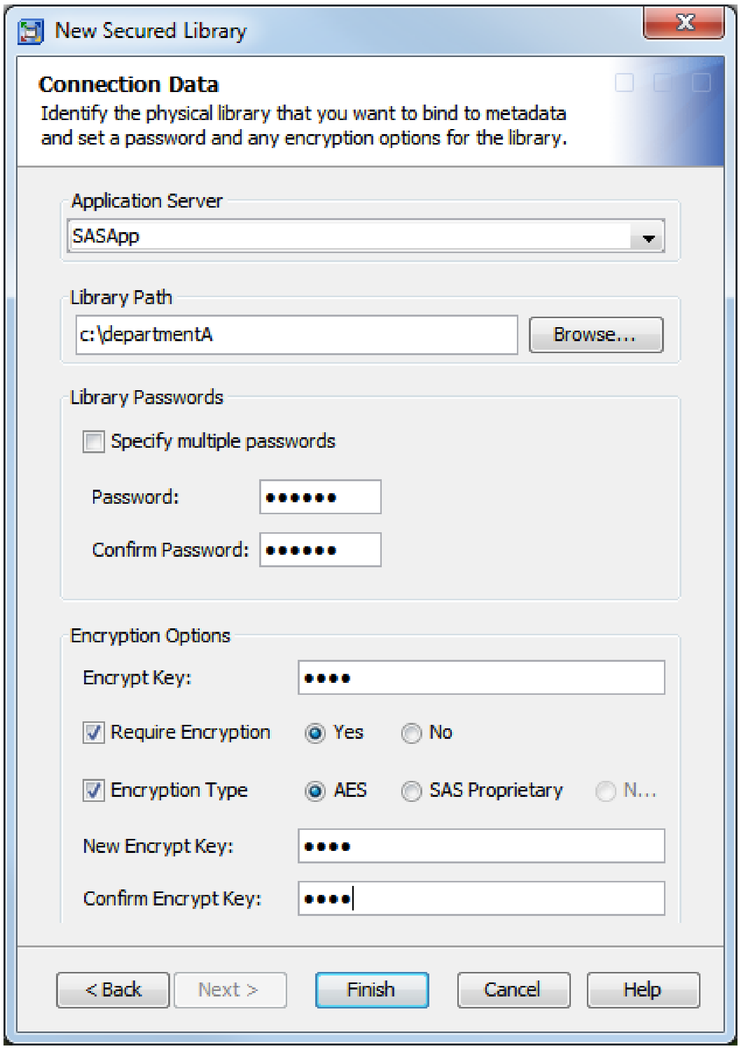
Name: Demo Library

Type: Secured library

Description:

Location: /System/Secured Libraries/Demo Folder

The **Connection Data** page of the New Secured Library wizard appears:


New Secured Library

Connection Data

Identify the physical library that you want to bind to metadata and set a password and any encryption options for the library.

Application Server: SASApp

Library Path: c:\departmentA

Library Passwords

☐ Specify multiple passwords

Password: ••••••

Confirm Password: ••••••

Encryption Options

Encrypt Key: ••••

☑ Require Encryption [✓ Yes] No

☑ Encryption Type [✓ AES] SAS Proprietary N...

New Encrypt Key: ••••

Confirm Encrypt Key: ••••

Refer to the entries in the preceding example as you complete steps 3 through 8.

3. Select the application server that you want to use to bind the target directory to metadata.

   Note: The application server must include a standard workspace server that has host access to the target directory.

4. Specify the directory path that you want to bind to metadata.

5. Set and confirm a password for the new metadata-bound library.

   Note: The password can be no more than eight characters long. To create a more complex password, select the **Specify multiple passwords** check box and supply three distinct passwords. Using multiple passwords for a metadata-bound library only increases security; the different passwords do not manage different types of access.

   Note: Users do not supply metadata-bound library passwords in order to access data, so they should neither know, nor have access to, the password values.

   CAUTION:

   If you lose the password for a metadata-bound library, you cannot unbind the library or change its password. Keep track of passwords that you assign.

6. If any of the library's data sets are currently encrypted using AES encryption, enter the current key in the **Encrypt Key** field.

TIP The key that you enter is placed in quotation marks when it is submitted to SAS and is therefore case sensitive. If the key was originally specified in SAS code without quotation marks, then be sure to use uppercase letters when entering it here.

Note: If the tables do not all have the same key, then you must use the code method to bind the library to metadata.

7. If you want to require encryption for all tables that are bound to the library, select the **Require Encryption** check box and select **Yes**. Then take one of the following actions:

   - To require AES encryption, you must specify an AES encryption key to store in the library's metadata. See step 8.
   - To require SAS Proprietary encryption, select the **Encryption Type** check box and select **SAS Proprietary**. Then go to step 9.

8. Follow these steps if you are requiring AES encryption or if you want to store a default key to use for encrypting some data sets:

   a. Select the **Encryption Type** check box and select **AES**.

   b. Enter an encryption key in the **New Encrypt Key** and **Confirm Encrypt Key** fields.

      TIP The value that you enter is a passphrase of up to 64 characters in length, from which the actual AES encryption key is derived. Most SAS documentation refers to the passphrase as the encryption key.

      TIP The key is placed in quotation marks when it is submitted to SAS and is therefore case sensitive.

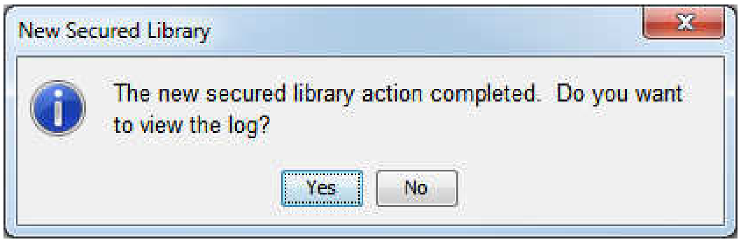
      TIP Be sure to keep a record of the encryption key, even though it is stored in metadata.

   If encryption is required, the stored encryption key will be used to encrypt every data set that is bound in the library. If encryption is not required, the stored key will be used to encrypt new tables when AES encryption is specified in SAS code but no key is supplied.

   CAUTION:
   For AES-encrypted data sets that are referentially related to one another, follow these best practices to ensure that the data does not become inaccessible: Store the encryption key in the library's metadata. You can modify the stored key, but do not remove the key from metadata and do not unbind the library.

9. Click **Finish**.

10. In the New Secured Library window, click **Yes** to view the log.


New Secured Library

The new secured library action completed. Do you want to view the log?

It is strongly recommended that you always check the log for warnings after you perform an action on a secured library object.

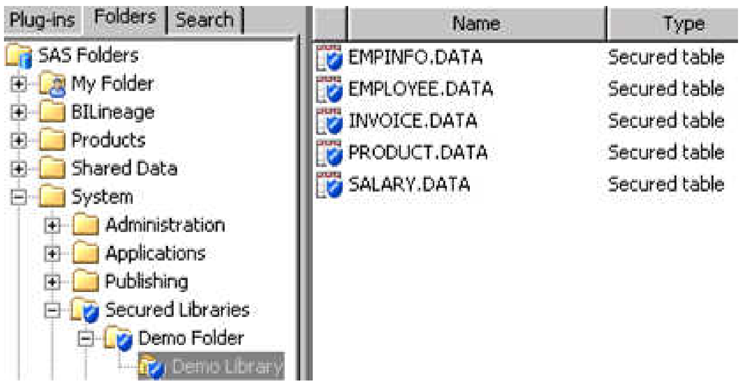
Code Method

As an alternative to using SAS Management Console, you can use SAS code to bind the data. See "CREATE Statement" on page 77.

Results

After the data is bound, each eligible physical table in your test library is represented in the metadata as a new secured table object. A table is eligible if it is not already secured with a password that differs from the password that you supply.

The following image depicts the new secured library object and secured table objects in SAS Management Console. Your list of secured table objects corresponds to the tables that you copied in the preparation phase. See "Preparation" on page 10.



Name	Type
EMPINFO.DATA	Secured table
EMPLOYEE.DATA	Secured table
INVOICE.DATA	Secured table
PRODUCT.DATA	Secured table
SALARY.DATA	Secured table

TIP If the new objects are not immediately visible, right-click the **Secured Libraries** folder and select **Refresh**. The new secured table objects are visible in the right panel when their respective secured library object is selected in the folder tree.

Traditional Registration

If you want to support access from clients that use metadata to locate data, register the library and tables in metadata (using the Data Library Manager plug-in within SAS Management Console).

For example, to make the data available from within SAS Web Report Studio, you might register it beneath the `Shared Data` folder.

Use the Data Library Manager plug-in within SAS Management Console to perform this task.

TIP Permissions on a traditional library or table object can further limit access. For example, a user who reads data through the META LIBNAME engine (MLE) must have permissions on both the traditional table object (the ReadMetadata and Read permissions) and the secured table object (the ReadMetadata and Select permissions).

Validating a Metadata-Bound Library

Overview

Validating a metadata-bound library identifies any inconsistencies between the physical metadata-bound library and its corresponding metadata objects. The validation report lists any missing or mismatched physical tables, security location information, and metadata objects for a specified physical library.

To validate a metadata-bound library, use either SAS Management Console or the AUTHLIB procedure.

TIP In general, you perform this validation against a physical library that is already bound to metadata. However, you can also perform this validation against a traditional library, in order to determine whether any individual physical tables within that library are bound to metadata.

Requirements

In order to validate a metadata-bound library, the following requirements must be met:

- The workspace server (or SAS session) that makes the request must run under an account that has host-layer Read access to the target physical library. This is necessary in order to assign the libref.
- The workspace server (or SAS session) that makes the request must connect to the metadata server as an identity that has the ReadMetadata permission for the target secured library object and secured table objects.

GUI Method

Introduction

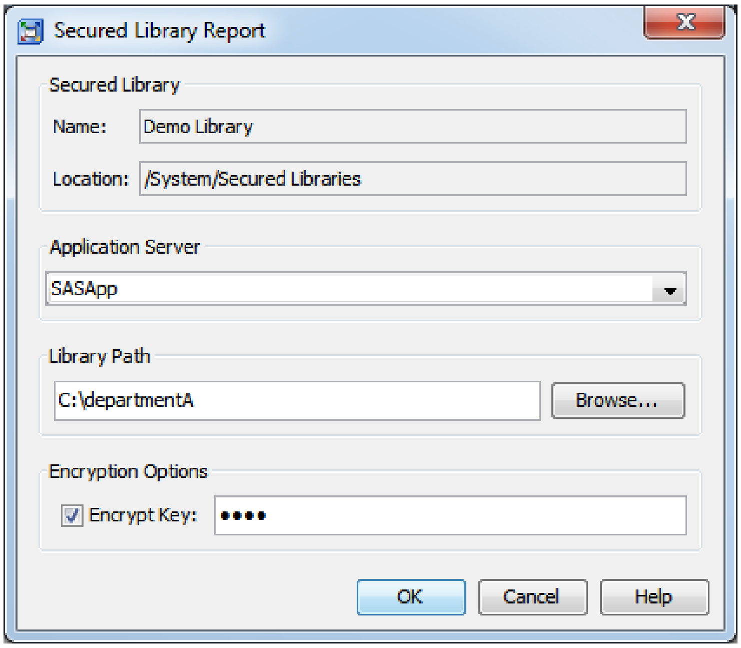
To generate a metadata-bound library report in SAS Management Console, select the **Report** action on a secured library object. The report identifies any missing or mismatched security information for the corresponding metadata-bound library and its tables.

Note: In SAS Management Console, the report action is available for only secured library objects. To use SAS Management Console to generate a report for a traditional (unbound) library, initiate the task from any secured library object and enter the path of the target physical library in the Secured Library Report window.

Instructions

1. On the **Folders** tab in SAS Management Console, beneath a **`/System/Secured Libraries`** branch, locate the secured library object that corresponds to the metadata-bound library that you want to validate.
2. Right-click the object and select **Report**.

   The Secured Library Report dialog box appears:


Secured Library Report

Secured Library

Name: Demo Library

Location: /System/Secured Libraries

Application Server

SASApp

Library Path

C:\departmentA

Encryption Options

[✓] Encrypt Key: ●●●●

Refer to the entries in the preceding example as you complete steps 3 through 5.

3. Select the application server that you want to use to generate the report.

   Note: The application server must include a standard workspace server that has host access to the target directory.

4. Specify the directory path of the target metadata-bound library.

5. If the library includes one or more AES-encrypted tables and the key is not recorded in the library's metadata, select the **Encrypt Key** check box and enter the key to open the data sets.

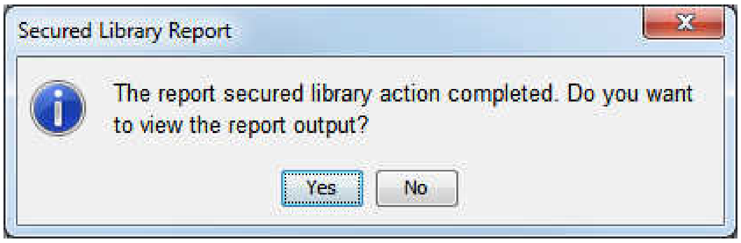
   Note: The key that you enter is placed in quotation marks when it is submitted to SAS and is therefore case sensitive. If the key was originally specified in SAS code without quotation marks, then be sure to use uppercase letters when entering it here.

   Note: If the library contains a table that is encrypted with a different key, the message `Invalid ENCRYPTKEY` will appear in the report output.

   If a key is recorded in the library's metadata, you can leave the **Encrypt Key** field blank. The stored key will be used to open the data sets.

6. Click **OK**.

7. In the Secured Library Report window, click **Yes** to view the report and the log.


Secured Library Report

The report secured library action completed. Do you want to view the report output?

Code Method

As an alternative to using SAS Management Console, you can use the AUTHLIB procedure to generate a metadata-bound library report. See “REPORT Statement” on page 94.

Results

It is important to carefully examine the entire report. The first section of the report lists all physical tables that are properly bound. Subsequent sections list any physical tables that are not properly bound and identify any other discrepancies.

TIP To generate the report in HTML format, use the AUTHLIB procedure (instead of SAS Management Console). For an example, see Output A2.2 on page 113.

Unbinding a Metadata-Bound Library

Overview

You unbind a library in order to remove its metadata-based protections. When you perform this action, physical and metadata content are affected as follows:

- Physical security information is deleted from the library directory and table headers.
- Corresponding secured library and secured table objects are deleted from the SAS metadata.

You can use either SAS Management Console or the AUTHLIB procedure to unbind a library.

TIP It is a good practice to validate a library before you unbind it. See “Validating a Metadata-Bound Library” on page 16.

Note: An unbind action affects only those physical tables that are located within the specified library. If a bound table is host-copied to another directory, that table’s security location information is unaffected by subsequent unbind actions against its parent metadata-bound library. In order to reestablish access to the table, corrective action against the table is necessary. See “REPAIR Statement” on page 89.

Requirements

In order to unbind a library, the following requirements must be met:

- You must know the current password for the metadata-bound library.
- The requesting workspace server (or SAS session) must run under an account that has host-layer control of the target physical library. For host-specific details, see “Requirement for Host-Layer Control” on page 42.
- The requesting workspace server (or SAS session) must connect to the metadata server as an identity that has the necessary metadata-layer permissions for the target secured data folder, secured library object, and secured table objects. See “Permissions for Metadata-Bound Data” on page 39.

GUI Method

Introduction

When you delete a secured library object in SAS Management Console, access to the associated physical directory is affected as follows:

- If you right-click a secured library object and select **Delete**, the associated physical directory and tables remain bound (to a no-longer existing secured library object), so the data is inaccessible.

  CAUTION:
  Use the Delete action only if you are certain that the associated physical directory no longer exists.

- If you right-click a secured library object and select **Delete and Unbind**, the associated physical directory and tables are unbound from, and no longer protected by, metadata. This is the standard method for deleting a secured library object.

  After you perform a **Delete and Unbind** action, the remaining protection for the library's contents is as follows:

  - If you do not supply new passwords, the current passwords are preserved on the affected tables.

    TIP If you do not want the current passwords to be preserved, select the **Change passwords** check box but leave the fields for new passwords blank. With this approach, the only remaining protection for the data is host-layer access controls.

  - If you supply new passwords, the data is available to only those users who know the new passwords and have host-layer access.
  - If you do not supply new encryption options, the current encryption is preserved in the affected tables.

    TIP If you do not want encryption to be preserved, select **None** as the encryption method.

  - You can specify a different encryption method or a new AES encryption key to be applied to the physical data.
  - If AES encryption is applied or preserved, or if you specify a new AES encryption key, the data is available to only those users who supply the key and have host-layer access.

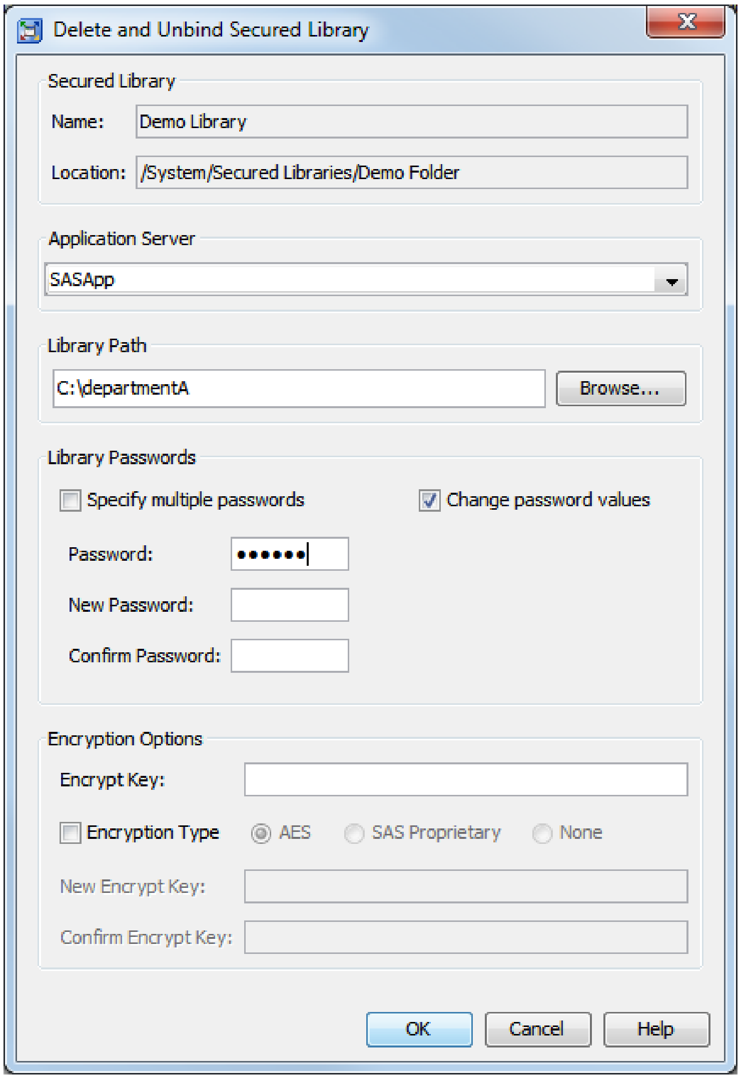
Instructions

To unbind a physical library and delete its corresponding secured library and secured table objects:

1. On the **Folders** tab in SAS Management Console, beneath a **`/System/Secured Libraries`** branch, locate the secured library object that corresponds to the metadata-bound library that you want to unbind.
2. Right-click the object and select **Delete and Unbind**.

   CAUTION:
   The Delete action removes the secured library object and leaves any corresponding data bound. Be sure to select the **Delete and Unbind** action, not the **Delete** action.

The Delete and Unbind Secured Library dialog box appears:


Delete and Unbind Secured Library

Secured Library

Name: Demo Library

Location: /System/Secured Libraries/Demo Folder

Application Server

SASApp

Library Path

C:\departmentA

Library Passwords

[ ] Specify multiple passwords [✓] Change password values

Password: ••••••

New Password:

Confirm Password:

Encryption Options

Encrypt Key:

[ ] Encryption Type [● AES] ○ SAS Proprietary ○ None

New Encrypt Key:

Confirm Encrypt Key:

Refer to the entries in the preceding example as you complete steps 3 through 8.

3. Select the application server that you want to use to unbind the target directory.

   Note: The application server must include a standard workspace server that has host access to the target directory.

4. Verify that the directory path of the target metadata-bound library is correct.

   Note: The directory path is pre-populated with the most recently referenced path. If any directories in the path have been renamed, be sure to modify the path.

5. Supply the current password of the target metadata-bound library.

   Note: If the target metadata-bound library has three distinct passwords, select the **Specify multiple passwords** check box, so there are three fields in the **Password** row. Supply all three passwords.

6. If you want to change or remove the current password, select the **Change password values** check box. Then take one of the following actions:

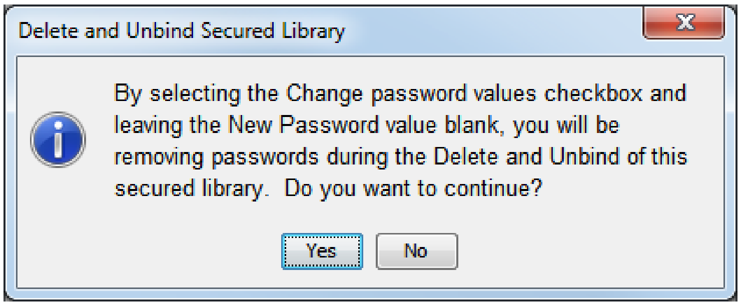
   • To specify a new password for accessing the unbound data, enter new values in the **New Password** and **Confirm Password** fields.

- To make the target accessible without passwords, leave the **New Password** and **Confirm Password** fields blank.

7. If the data sets are currently encrypted using AES encryption and the key is not stored in the library's metadata, enter the current key in the **Encrypt Key** field.
8. If you want to change the encryption options, select the **Encryption Type** check box. Then take one of the following actions:
   - To apply AES encryption (or to specify a new AES encryption key), select the **AES** radio button. Then enter an encryption key in the **New Encrypt Key** and **Confirm Encrypt Key** fields.

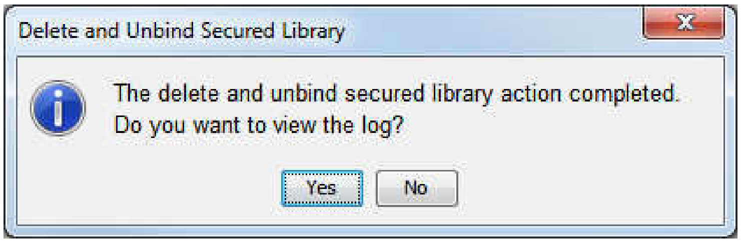
     TIP Make sure that you have a record of the encryption key, as users will need to supply it when accessing the unbound data.
   - To apply SAS Proprietary encryption, select the **SAS Proprietary** radio button. You cannot remove the library's password if you use this encryption type.
   - To remove all encryption from the library's data sets, select the **None** radio button.

   CAUTION:
   If you have to unbind a library that contains AES-encrypted data sets that are referentially related to other data sets, then either make sure that all related data sets are no longer AES-encrypted or make sure that all related data sets share the same encryption key. If you preserve AES encryption, the data will be available only to those users who supply the key and have host-layer access.
9. Click **OK**.
10. In the Delete and Unbind Secured Library window, click **Yes** to confirm your intent.


Delete and Unbind Secured Library

By selecting the Change password values checkbox and leaving the New Password value blank, you will be removing passwords during the Delete and Unbind of this secured library. Do you want to continue?

11. In the Delete and Unbind Secured Library window, click **Yes** to view the log.


Delete and Unbind Secured Library

The delete and unbind secured library action completed. Do you want to view the log?

It is strongly recommended that you always check the log for warnings after you perform an action on a secured library object.

Scope

In general, all tables within a metadata-bound library are protected by the library and share the library's password. The GUI method facilitates this simple, best practice approach by affecting the physical library and all of its eligible tables. A table is eligible if it is either unsecured or secured with the library password.

The following list identifies selective scope situations, in which you cannot use the GUI method:

- If you do not want to affect the library, use the code method and set the TABLESONLY option.
- If you want to affect only some of the tables, use the code method and add a TABLES statement after the MODIFY statement.

Code Method

As an alternative to using SAS Management Console, you can use the AUTHLIB procedure to unbind a library. See "REMOVE Statement" on page 86.

Results

After you complete the preceding steps, the physical library remains intact, but is no longer bound to or protected by metadata.

In the host layer, the security location information is deleted from the target directory and files.

In the metadata layer, the corresponding secured library and table objects are deleted.

TIP To update the display in SAS Management Console, right-click the **Secured Libraries** folder and select **Refresh**.

Changing a Metadata-Bound Library Password

Overview

Before you change a library password, it is essential to understand when and how library passwords are used. See "Passwords for Metadata-Bound Data" on page 43.

You might change a library password for any of the following reasons:

- You suspect the password has been compromised.
- Your security policy mandates periodic changes to library passwords.
- You want to bind tables that have been host-copied into the metadata-bound library (in order to maintain the best practice scenario where all tables within a metadata-bound library are bound to that library).

To change the password of a metadata-bound library, use either SAS Management Console or the AUTHLIB procedure.

Requirements

In order to change a library password, the following requirements must be met:

- You must know the current password for the metadata-bound library.

- The requesting workspace server (or SAS session) must run under an account that has host-layer control of the target physical library. For host-specific details, see "Requirement for Host-Layer Control" on page 42.
- The requesting workspace server (or SAS session) must connect to the metadata server as an identity that has the ReadMetadata and WriteMetadata permissions to the corresponding secured library object and secured table objects.

  Note: On a secured library object, the WriteMemberMetadata permission (from the parent secured data folder) is inherited as the WriteMetadata permission. See "WriteMetadata and WriteMemberMetadata" in *SAS Intelligence Platform: Security Administration Guide*.

GUI Method

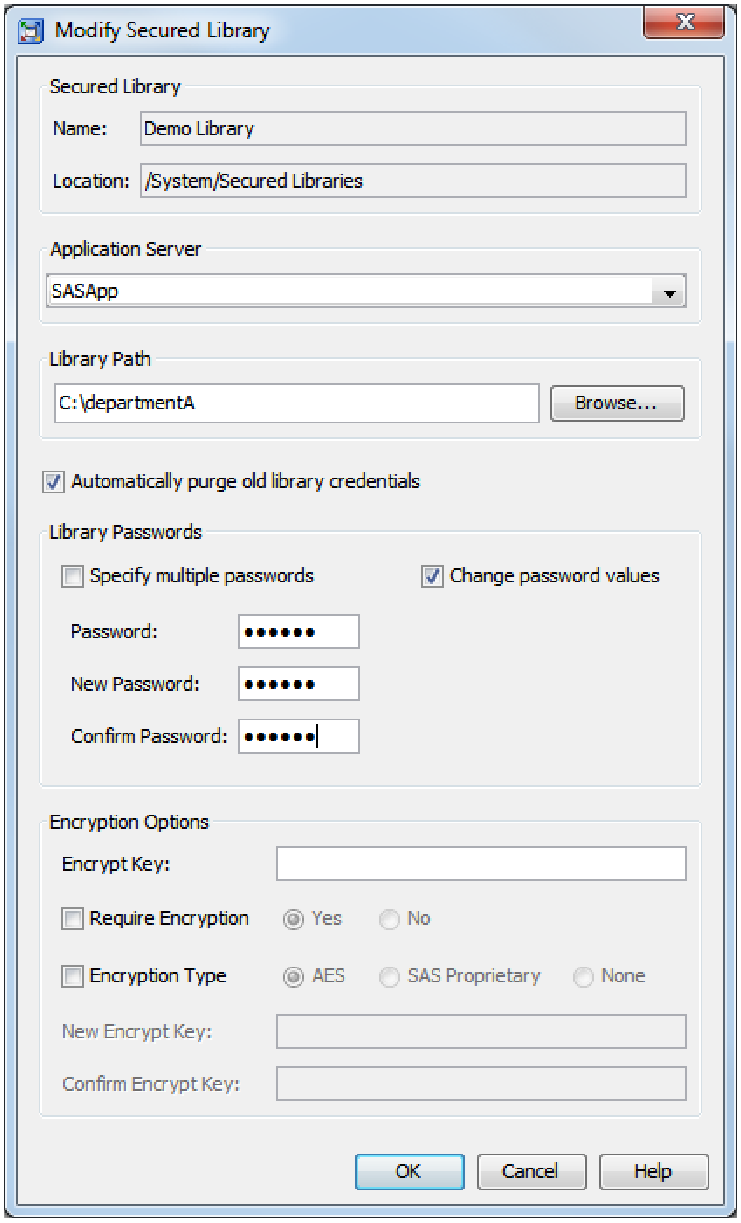
Introduction

In SAS Management Console, you change a library password by modifying its corresponding secured library object.

Instructions

1. On the **Folders** tab in SAS Management Console, beneath a **`/System/Secured Libraries`** branch, locate the secured library object that corresponds to the metadata-bound library whose password you want to change.
2. Right-click the object and select **Modify**.

   The Modify Secured Library dialog box appears:


Modify Secured Library

Secured Library

Name: Demo Library

Location: /System/Secured Libraries

Application Server: SASApp

Library Path: C:\departmentA

[✓] Automatically purge old library credentials

Library Passwords

[ ] Specify multiple passwords [✓] Change password values

Password: ••••••

New Password: ••••••

Confirm Password: ••••••

Encryption Options

Encrypt Key:

[ ] Require Encryption [✓ Yes] No

[ ] Encryption Type [✓ AES] SAS Proprietary None

New Encrypt Key:

Confirm Encrypt Key:

Refer to the entries in the preceding example as you complete steps 3 through 7.

3. Select the application server that you want to use to update the binding information in the target directory.

   Note: The application server must include a standard workspace server that has host access to the target directory.

4. Verify that the directory path of the target metadata-bound library is correct.

   Note: The directory path is pre-populated with the most recently referenced path. If any directories in the path have been renamed, be sure to modify the path.

5. The **Automatically purge old library credentials** check box is selected by default. This option automatically removes all retained metadata-bound library credentials (passwords or encryption keys) if all tables in the library are successfully modified to use the newer credentials.

   If you want the replaced credentials to be retained in metadata, then clear the check box. The passwords are retained until you use the PURGE statement to remove them, or until you later modify the library with the check box selected. The following are reasons that you might want to retain credentials:

- You created views, using the old passwords, to implement row and column level security on the library's tables. SAS does not know which view files might contain the old passwords and does not have the ability to modify them in the view file. The old passwords need to be retained until you redefine the views to use the new passwords.
- You want to be able to process data sets that are restored from backups taken before the passwords were modified.

6. In the **Password** field, supply the current password for the target metadata-bound library.

   Note: If the target metadata-bound library currently has three distinct passwords, select the **Specify multiple passwords** check box and supply all three passwords in the **Password** row.

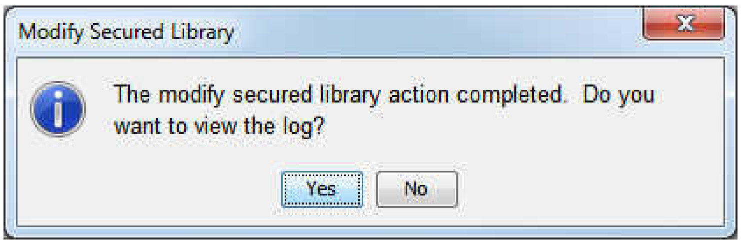
7. Select the **Change password values** check box.
8. In the **New Password** and **Confirm Password** fields, set and confirm a new password for the metadata-bound library. The password can be only eight characters long.

   CAUTION:
   If you lose the password for a metadata-bound library, you cannot unbind the library or change its password. Keep track of passwords that you assign.

   TIP To create a more complex password, select the **Specify multiple passwords** check box and supply three distinct passwords. If the target metadata-bound library currently has one password, supply the current password in all three of the fields in the **Password** row. Using three passwords for a metadata-bound library only increases security; the different passwords do not manage different types of access.

   TIP Users do not supply metadata-bound library passwords in order to access data, so they should neither know, nor have access to, the password values.

9. Click **OK**.
10. In the Modify Secured Library window, click **Yes** to view the log.


Modify Secured Library

The modify secured library action completed. Do you want to view the log?

It is strongly recommended that you always check the log for warnings after you perform an action on a secured library object. If the log indicates that some tables were not modified (perhaps because a user was accessing them), repeat the modification when the tables are not being used. When doing so, specify the new password in the **Password** field.

Scope

In general, all tables within a metadata-bound library are protected by the library and share the library's password. The GUI method facilitates this simple, best practice approach by affecting the physical library and all of its eligible tables. A table is eligible if it is either unsecured or secured with the library password.

In the following selective scope situations, you cannot use the GUI method:

- If you do not want to affect the library, use the code method and set the TABLESONLY option.
- If you want to affect only some of the tables, use the code method and add a TABLES statement after the MODIFY statement.

Code Method

As an alternative to using SAS Management Console, you can use the AUTHLIB procedure to modify a library password. See “MODIFY Statement” on page 81.

Results

After you complete the preceding steps, the new password is recorded in the physical tables, replacing each instance of the old password.

The new password is also recorded in the metadata and associated with the corresponding secured library object.

Changing a Metadata-Bound Library’s Encryption Options

Overview

To change the encryption options for a metadata-bound library, use either SAS Management Console or the AUTHLIB procedure.

Requirements

In order to change a library’s encryption options, the following requirements must be met:

- You must know the current password for the metadata-bound library.
- The requesting workspace server (or SAS session) must run under an account that has host-layer control of the target physical library. For host-specific details, see “Requirement for Host-Layer Control” on page 42.
- The requesting workspace server (or SAS session) must connect to the metadata server as an identity that has the ReadMetadata and WriteMetadata permissions to the corresponding secured library object and secured table objects.

  Note: On a secured library object, the WriteMemberMetadata permission (from the parent secured data folder) is inherited as the WriteMetadata permission. See “WriteMetadata and WriteMemberMetadata” in *SAS Intelligence Platform: Security Administration Guide*.

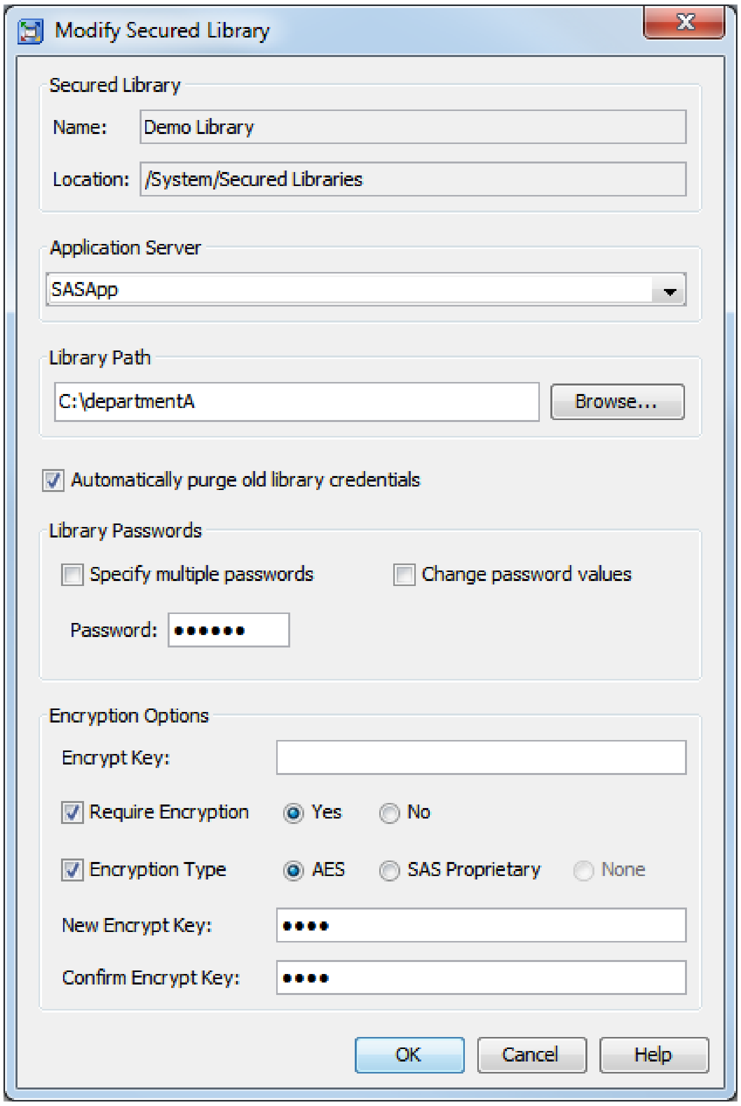
GUI Method

Introduction

In SAS Management Console, you change a library's encryption options by modifying its corresponding secured library object.

Instructions

1. On the **Folders** tab in SAS Management Console, beneath a **`/System/Secured Libraries`** branch, locate the secured library object that corresponds to the metadata-bound library whose encryption options you want to change.
2. Right-click the object and select **Modify**. The Modify Secured Library dialog box appears:



Secured Library
Name: Demo Library
Location: /System/Secured Libraries

Application Server
SASApp

Library Path
C:\departmentA

[✓] Automatically purge old library credentials

Library Passwords
[ ] Specify multiple passwords [ ] Change password values
Password: ••••••

Encryption Options
Encrypt Key:
[✓] Require Encryption [✓ Yes] No
[✓] Encryption Type [✓ AES] SAS Proprietary None
New Encrypt Key: ••••
Confirm Encrypt Key: ••••

Refer to the entries in the previous example as you complete steps 3 through 8.

3. Select the application server that you want to use to update the binding information in the target directory.

Note: The application server must include a standard workspace server that has host access to the target directory.

4. Verify that the directory path of the target metadata-bound library is correct.

   Note: The directory path is pre-populated with the most recently referenced path. If any directories in the path have been renamed, be sure to modify the path.

5. The **Automatically purge old library credentials** check box is selected by default. This option automatically removes all retained metadata-bound library credentials (passwords or encryption keys) if all tables in the library are successfully modified to use the newer credentials.

   If you want the replaced encryption key to be retained in metadata, then clear the check box. For example, you might want to retain the replaced encryption key so that you can process data sets that are restored from backups taken before the key was replaced. The old encryption key is retained until you use the PURGE statement to remove it, or until you later modify the library with the check box selected.

6. Supply the current password of the target metadata-bound library.

   Note: If the target metadata-bound library has three distinct passwords, select the **Specify multiple passwords** check box, so there are three fields in the Password row. Supply all three passwords.

7. If the library's data sets are already encrypted using AES encryption and the key is not stored in the library's metadata, enter the current key in the **Encrypt Key** field.

   TIP The value that you enter is a passphrase of up to 64 characters in length, from which the actual AES encryption key is derived. Most SAS documentation refers to the passphrase as the encryption key.

   TIP The key that you enter is placed in quotation marks when it is submitted to SAS and is therefore case sensitive. If the key was originally specified in SAS code without quotation marks, then be sure to use uppercase letters when entering it here.

8. If you want to require encryption for all tables that are bound to the library, select the **Require Encryption** check box and select **Yes**.

   If an AES encryption key was previously stored in the library's metadata, that key will be used to encrypt every data set that is bound to the library. If you want to use a different key, or if you did not previously store a key, specify the key as described in step 8.

   If you want to require SAS Proprietary encryption, select **Encryption Type** check box and select **SAS Proprietary**.

9. If you want to store an AES encryption key in the library's metadata or change the value of a previously stored key:

   a. Select the **Encryption Type** check box and select **AES**.

   b. Enter an encryption key in the **New Encrypt Key** and **Confirm Encrypt Key** fields.

   TIP The value that you enter is a passphrase of up to 64 characters in length.

   TIP The encryption key is placed in quotation marks when it is submitted to SAS and is therefore case sensitive.

   TIP Be sure to keep a record of the encryption key, even though it is stored in metadata.

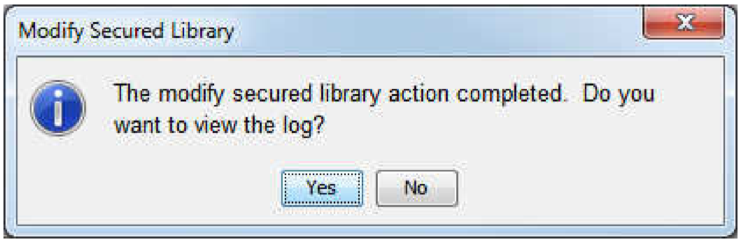
If encryption is required, the stored key will be used to encrypt every data set that is bound in the library.

If encryption is not required, the stored key will be used to re-encrypt every data set that was encrypted using a previously stored key. It will also be used to encrypt new tables when AES encryption is specified in SAS code but no key is supplied.

Note: If you choose not to require encryption, then you can use the TABLES statement with the code method to specify an encryption key for each table. However, SAS recommends that you store an encryption key in the library's metadata and use it for all of the library's metadata-bound data sets that are encrypted with AES.

CAUTION:
For AES-encrypted data sets that are referentially related to one another, follow these best practices to ensure that the data does not become inaccessible: Store the encryption key in the library's metadata. You can modify the stored key, but do not remove the key from metadata and do not unbind the library.

10. Click **OK**.
11. In the Modify Secured Library window, click **Yes** to view the log.


Modify Secured Library

The modify secured library action completed. Do you want to view the log?

It is strongly recommended that you always check the log for warnings after you perform an action on a secured library object. If the log indicates that some tables were not modified (perhaps because a user was accessing them), repeat the modification when the tables are not being used. When doing so, specify the new encryption key in the **Encrypt Key** field.

Note: You can remove encryption by selecting an encryption type of **None**. However, if encryption is currently required, you must use a two-step process. In the first step, select the **Require Encryption** check box, select **No**, and select the current encryption type. Click **OK** to save this change. Then, modify the library again, and select an encryption type of **None**.

Code Method

As an alternative to using SAS Management Console, you can use the AUTHLIB procedure to modify encryption options. See "MODIFY Statement" on page 81.

Results

After you complete the preceding steps, the tables are re-encrypted using the newly supplied options.

The new AES encryption key is recorded in the metadata and associated with the corresponding secured library object.

Verifying Access to Metadata-Bound Data

Who Can Read Metadata-Bound Data?

In order to read metadata-bound data, you must connect to the target metadata server as an identity that has the following metadata-layer effective access:

- the ReadMetadata permission (for the target secured table object and its parent secured library object)
- the Select permission (for the target secured table object)

If you are accessing the data from a client that uses metadata in order to locate data, you must also have the ReadMetadata permission for the corresponding traditional table object.

If the data is accessed through the MLE, you must also have the Read permission for the corresponding traditional table object.

Example

This example demonstrates one way to verify Read access to data.

In this example, you connect to a metadata server as a restricted user, set up a libref that points at a metadata-bound library, and then write a description of the contents of one of the metadata-bound tables within that library.

```
options
     metaserver="machine.company.com"
     metauser="sasdemo"
     metapass="********";

libname secdemo 'path';

proc datasets library=secdemo nolist;
   contents data=EMPINFO out=testout;
   title 'Contents of the Metadata-Bound Table EMPINFO';
run;
```

See Also

- "Facilitate End-User Access" on page 55
- "Limitations of Metadata-Bound Libraries" on page 5
- "DATASETS Procedure" in *Base SAS Procedures Guide*

Mutually Exclusive Access

Introduction

To establish several distinct levels of access, set up a metadata folder structure with appropriate permissions. Each secured library object inherits permissions from its metadata folder. Each secured table object inherits permissions from its parent secured library object.

This example demonstrates one way to set up mutually exclusive access for two user groups (GroupA and GroupB) to four libraries (LibraryA1, LibraryA2, LibraryB1, and LibraryB2).

Note: This example illustrates library-level access distinctions. You can also set up table-level access distinctions by setting metadata-layer permissions on individual secured table objects within a metadata-bound library.

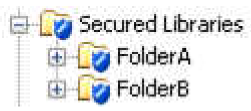
Preparation

The example assumes that the following prerequisites are met:

- The privilege and permission requirements are met. See “Requirements” on page 10.
- The data exists in the host.
- If you are using SAS code to perform the bind action, each physical library has been assigned a libref (liba1, liba2, libb1, and libb2) in your SAS session.
- GroupA and GroupB exist in the SAS metadata.

Instructions

1. On the **Folders** tab in SAS Management Console, beneath **`SAS Folders/System/Secured Libraries`**, create two sibling secured data folders named **`FolderA`** and **`FolderB`**.



2. Constrain access at the **`Secured Libraries`** folder. One way to do this is to explicitly deny all permissions to the PUBLIC group and explicitly grant all permissions to the SAS Administrators group. These protections flow throughout the Secured Libraries branch, except where modified by additional direct access controls.
3. Expand access to the new folders as follows:

Folder	Metadata Group	Explicit Grants*
`FolderA`	GroupA	ReadMetadata and Select

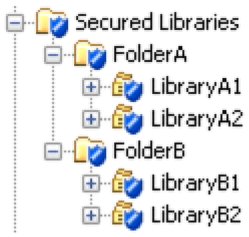
Folder	Metadata Group	Explicit Grants*
FolderB	GroupB	ReadMetadata and Select

* For conciseness, this example uses individual explicit controls (instead of ACTs) and provides only Read access (the Select permission). These settings do not allow members of GroupA and GroupB to update or delete data.

TIP To add GroupA and GroupB to the **Authorization** tab, click the **Add** button next to the **Users and Groups** list box. In SAS Management Console, an explicit setting has a white background color (not gray or green).

4. To bind the physical data to metadata, either use SAS Management Console or submit code. Be sure to specify **FolderA** as the metadata location for the first two libraries, and **FolderB** as the metadata location for the last two libraries.

5. In SAS Management Console, examine the contents of **FolderA** and **FolderB**.



```
Secured Libraries
  FolderA
    LibraryA1
    LibraryA2
  FolderB
    LibraryB1
    LibraryB2
```

TIP If the new secured library objects are not immediately visible, right-click the **Secured Libraries** folder and select **Refresh** from the pop-up menu. The new secured table objects are visible in the right panel when their respective secured library object is selected in the folder tree.

Examine the **Authorization** tab of several of the new objects to verify that metadata-layer access is as expected.

6. If you want to provide access through clients that use metadata to locate data, register the library and tables in metadata. For example, if the data is accessed from SAS Web Report Studio, you might register it beneath the **Shared Data** folder.

Results

Test access from various clients. Behavior should be as follows:

- A user who is unrestricted should have access to all of the tables.
- A user who is a direct or indirect member of both GroupA and GroupB should have access to all of the tables.
- A restricted user who is a member of only GroupA or only GroupB should have access to only the data beneath **FolderA** or **FolderB**.
- A restricted user who is not GroupA, GroupB, or the SAS Administrators group should not have access to any of the data.

See Also

"Binding Data to Metadata" on page 10

Providing Fine-Grained Access Using Views

Overview

You can create views that give users access to some but not all of the data within a physical table. Use the following approach:

1. If the physical table and its parent library are not already bound to metadata, bind them.
2. Create a view that excludes the rows or columns that you want to hide.
3. Apply the password of the underlying physical table to the view.
4. Set metadata-layer permissions to control who can access the view.

Column-Level Access

In this example, partial access to a customer data table is provided by creating a view and managing access to it. The view keeps the name and telephone number columns from the underlying table, but excludes the credit card number column.

```
options metauser="sasadm@saspw" metapass="********"
   metaserver="machine.company.com";

libname cust 'path';

proc authlib library=cust;
   create
      securedlibrary='cust'
      securedfolder='CustomerData'
      pw=secret;
quit;

proc sql;
   create view cust.PUBLIC as
      select Name, Phone
      from cust.PRIVATE(pw=secret);
quit;
```

The preceding code creates a new secured library object (CustomerData) that contains two objects: a table object (called PRIVATE) and a view object (called PUBLIC).

Note: The password that is supplied to bind the library is also supplied when the PUBLIC view is defined against the PRIVATE table. In order to create a view of a metadata-bound table, you must know the password of that physical table's parent library, and provide that password in the view definition. You can enable end users to access the view without giving them access to the underlying table. In effect, this provides selective access to the columns and rows within the underlying table.

Note: If you modify the password for the metadata-bound library, you must also update the view definition with the new password. Until you have time to redefine all of the views to use the new password, you can retain the old password in metadata by deselecting **Automatically purge old library credentials** in the Modify Secured

Library dialog box. The old password is retained until you use the PURGE statement to remove it, or until you later modify the library with the **Automatically purge old library credentials** check box selected.

To complete the protection, use SAS Management Console to set metadata-layer permissions so that restricted users can access the PUBLIC view but not the PRIVATE table. For example, if only unrestricted users should access the PRIVATE table, you might use the following approach:

- On the **Authorization** tab for the **`CustomerData`** folder, verify that the PUBLIC group is denied the ReadMetadata, WriteMetadata, WriteMemberMetadata, and Select permissions. Verify that the PRIVATE table inherits these denials.
- On the **Authorization** tab for the PUBLIC view object, explicitly grant the ReadMetadata and Select permissions to SASUSERS.

Identity-Driven, Row-Level Access

In this example, partial access to an employee information table (HR.EMPINFO) is provided by creating a view (HR.PERSONAL) that dynamically filters rows in the underlying table. The filtering is based on each requesting user’s authenticated user ID. The filtering relies on a security associations table, which maps each user’s authenticated user ID to a corresponding employee ID.

The following code creates the identity-driven view of the employee information table. When requesting users access the view, they retrieve only those rows that match the user ID with which they authenticated to the metadata server.

```
proc sql;
   create view hr.personal as
      select a.*
      from hr.empinfo(pw=secret) a,
           hr.security(where=(loginid=_METADATA_AUTHENTICATED_USERID_)) b
      where b.loginid ne '' and a.empid = b.empid;
quit;
```

Here are some details about the preceding code:

- The code assumes that the HR libref is already established and points to a metadata-bound library that has a single password value of **`secret`**.
- The reference to the EMPINFO table must supply the password (**`secret`**) in order to create the view, because the table is bound to metadata.
- SECURITY is a security associations table that maps all valid _METADATA_AUTHENTICATED_USERID_ values to the primary key of the target table (the EMPID column in the EMPINFO table).

  Note: As an alternative to creating a separate security associations table, you could directly add a column of _METADATA_AUTHENTICATED_USERID_ values to your target table.
- _METADATA_AUTHENTICATED_USERID_ is a substitution parameter that supplies a user-specific value in each request, based on the user ID with which the requesting user authenticated to the metadata server.
- The _METADATA_AUTHENTICATED_USERID_ substitution parameter is used in a WHERE clause that is expressed as a data set option.

If you want to provide broader access to certain users (for example, to enable department managers to see information about their employees), you can enhance the SECURITY table to include a column that maps employees to departments, create an additional view that exploits that mapping, and set metadata-layer permissions so that only department managers can use the new view.

See Also

- "SAS Views" in *SAS Language Reference: Concepts*
- "Connection Options " in *SAS Language Interfaces to Metadata*

Providing Fine-Grained Access Using Condition Permissions

Overview

Effective with the first maintenance release for SAS 9.4, you can use permission conditions to give users access to some but not all of the data within a physical table. Use the following approach:

1. If the physical table and its parent library are not already bound to metadata, bind them.
2. Set metadata-layer permissions to control who can access each table.
3. Use SAS Management Console to specify permission conditions.

Instructions

1. On the **Folders** tab in SAS Management Console, beneath a **`/System/Secured Libraries`** branch, select the secured library object that corresponds to the metadata-bound library whose data sets you want to protect.
2. In the right panel, right-click the table for which you are defining a permission condition. Select **Properties**, and then select the **Authorization** tab of the properties dialog box.
3. Select or add the identity whose access you want to limit.
4. In the permissions list, add an explicit ☑ grant of the Select permission for the selected identity. In SAS Management Console, an explicit setting has a white background color (not gray or green).
5. Click the **Add Condition** button.

   Note: If the **Edit Condition** button is displayed, a condition already exists for the selected user or group. You can click this button to modify the condition.

6. In the Permission Condition dialog box, enter the WHERE clause for an SQL query that filters the data as appropriate for the selected identity. Do not include the WHERE key word in your entry.

   TIP To make dynamic, per-person access distinctions, you can use identity-driven properties as the values against which target data values are compared. Use the following syntax when specifying one of these properties: **`SUB::`*`property-name`*** (for example, **`SUB::SAS.Userid`**). For a list of available identity-driven properties, see "Fine-Grained Controls for Data" in *SAS Intelligence Platform: Security Administration Guide*.

CAUTION:

The syntax that you enter and save in the Permission Condition dialog box is not checked for validity. Make sure that the syntax that you have entered is correct.

7. Click **OK**.

Best Practices

General Guidelines

The following list provides general guidelines for implementing metadata-bound libraries:

- Use SAS (not host commands) for management of physical data. Using host commands does not compromise security, but it can decrease clarity and create noise (warnings) in the audit logs. See Appendix 1, "Security Impact of Moving Tables," on page 59.

  Note: An exception to this guideline is that using a host copy command to back up or restore physical data to the same directory is not problematic.

- Within the `/System/Secured Libraries` folder in metadata, open up access only as necessary. In particular, grant the WriteMetadata and WriteMemberMetadata permissions to only administrators that should be able to change access to the metadata-bound data.
- After you bind libraries to metadata, review the metadata-layer permissions on the generated secured library objects and secured table objects, and adjust access if needed. For example, you can use SAS Management Console to add users or groups to a secured table's **Authorization** tab, and grant (or deny) the Select permission (to manage Read access to the data).
- If you use the metadata promotion tools (for example, to create separate deployments for development, test, and production environments), maintain a separate copy of your physical data for each environment. The alternative, pointing multiple metadata servers at the same physical data, is supported but introduces significantly more complexity. For more information about promoting secured library objects, secured table objects, and secured data folders, see "Promotion Details for Specific Object Types" in *SAS Intelligence Platform: System Administration Guide*.
- Create or modify a metadata-bound library at a time when the physical data is not being accessed by other users. If the physical data is in use, some actions on open tables might fail. Interactions with a libref that was established before a library is bound are as follows:
  - For an existing physical table, the pre-established libref is subject to security that is implemented in the subsequent statement. Access that occurs in these circumstances causes a message to be written to the log. The message explains that the physical table is in a library that does not have a secured library location.
  - For a new physical table, the pre-established libref is not subject to security that is implemented through the subsequent statement, and the new table is not bound to a secured table object.

- If you bind data that users are accustomed to accessing directly, inform those users that they must establish a connection to the metadata server before they can assign a libref against a metadata-bound library.
- In your metadata backup strategy, remember to consider your secured data folders, secured library objects, and secured table objects. See "Best Practices for Backing Up and Restoring Your SAS Content" in *SAS Intelligence Platform: System Administration Guide*.
- Use the LIBNAME option AUTHADMIN=YES when you are repairing any inconsistencies between physical data and its corresponding secured library and secured table objects in metadata. Do not use AUTHADMIN=YES in other circumstances.

Avoiding Mixed States

Binding data to metadata is a library-level feature. In general, all tables within a metadata-bound library are protected by that library and share the library's password.

To establish and maintain a best practice state that minimizes complexity:

- Ensure that all physical tables within a metadata-bound library are protected by that library. This standard, default state maximizes clarity. Special circumstances (for example, a table that has a different, pre-existing password) can result in a mixed state (for example, one of the tables within a metadata-bound library is not bound to that library). To verify that this guideline is met, validate the library.
- Ensure that all physical tables that are protected by a particular metadata-bound library remain within that library. This standard, default state maximizes clarity and is essential for unbind actions to be fully effective. Special circumstances (for example, a table that is host-copied to another directory) can prevent an unbind action from unbinding the relocated data set.

For example, if someone uses a host copy command to add unbound physical tables to a metadata-bound library, the added tables are not automatically bound. To bind the added physical tables, use the modify action to re-apply the password of the metadata-bound library. See "Changing a Metadata-Bound Library Password" on page 22.

Note: If the host-copied tables were already bound to another library, you must first use the REPAIR DELETE LOCATION statement of the AUTHLIB procedure to remove that binding. See "REPAIR Statement" on page 89. After you remove the original binding, you can use the modify action to re-apply the password of the library in which the tables are currently located.

Chapter 3
Reference for Metadata-Bound Libraries

Permissions for Metadata-Bound Data

Permissions on Secured Library and Table Objects

For secured library objects and secured table objects, SAS enforces the following special metadata-layer permissions:

Table 3.1 *Permissions for Metadata-Bound Data*

Permission	Abbreviation	Actions Affected
Delete	D	Delete rows in a physical table. For example, in order to use SAS to delete data from a metadata-bound table, you need the Delete permission on the corresponding secured table object. You also need the Select permission on that object.
Insert	I	Add rows to a physical table. For example, in order to use SAS to add data to a metadata-bound table, you need the Insert permission on the corresponding secured table object.
Update	U	Update rows in a physical table. For example, in order to use SAS to update data in a metadata-bound table, you need the Update permission on the corresponding secured table object. You also need the Select permission on that object.
Select	S	Read rows within a physical table. For example, in order to use SAS to read data from a metadata-bound table, you need the Select permission on the corresponding secured table object.
Create Table	CT	Create a new physical table. For example, in order to use SAS to add a table to a metadata-bound library, you need the Create Table permission on the corresponding secured library object. Rename a physical table (if that action creates a new table, rather than overwriting a preexisting table). For example, if you rename TableA to TableB in a metadata-bound library that does not already contain a TableB, you need the Create Table permission on the corresponding secured library object. You also need the Alter Table permission on TableA's corresponding secured table object.
Drop Table	DT	Delete a physical table. For example, in order to use SAS to delete a metadata-bound table, you need the Drop Table permission on the corresponding secured table object.*
Alter Table	AT	Replace a physical table. For example, in order to use SAS to replace a metadata-bound table, you need the Alter Table permission on the corresponding secured table object. Rename a physical table. For example, in order to use SAS to rename a metadata-bound table, you need the Alter Table permission on the corresponding secured table object. You also need the Create Table permission on the corresponding secured library object. Perform other administrative updates on a physical table, such as modifying variable names and labels. For example, in order to use SAS to change labels in a metadata-bound table, you need the Alter Table permission on the corresponding secured table object.

* A user who has Write access in the host layer can delete physical tables using operating system commands, regardless of whether the user has a grant of DT in the metadata layer. Any table replacements are detected through audit log entries. See "Auditing for Metadata-Bound Libraries" on page 45.

Permission Requirements

Introduction

This topic addresses tasks that are performed using SAS. Tasks that are instead performed using host commands are not subject to metadata-layer permission requirements. See Appendix 1, “Security Impact of Moving Tables,” on page 59.

Table-Level Tasks

The following table documents the effective metadata-layer grants on a secured table object that are required in order to perform certain tasks with that object.

TIP Each of the following tasks is initiated by SAS against physical data (SAS data set files or SAS views). For data that is bound to metadata, SAS always enforces metadata-layer permission requirements before allowing access.

Table 3.2 *Metadata-Layer Permission Requirements for Selected Tasks*

Task	**Effective Grants (on the secured table object)**
View data in a metadata-bound table	Select
Add rows to a metadata-bound table	Insert
Update rows in a metadata-bound table	Select, Update
Delete rows from a metadata-bound table	Select, Delete
Replace a metadata-bound table with a new version of data	Alter Table
Rename a metadata-bound table (for example, using PROC DATASETS with CHANGE)	Alter Table, Create Table*
Modify variable names or labels in a metadata-bound table (for example, using PROC DATASETS with MODIFY)	Alter Table
Copy a metadata-bound table out of a metadata-bound library (for example, using PROC COPY)	Select
Move a metadata-bound table out of a metadata-bound library (for example, using PROC COPY with MOVE)	Select, Drop Table
Delete a metadata-bound table from the file system (for example, using PROC DATASETS with DELETE)	Drop Table

* The Create Table permission is required on any target secured table object that will be overwritten. If there is no such object, then the Create Table permission is required on the parent library.

Library-Level Tasks

The following table documents which effective metadata-layer grants on which objects are required in order to perform certain library-level tasks.

TIP Each of the following tasks is initiated by SAS against physical data. With metadata-bound libraries, SAS always enforces metadata-layer permission requirements before allowing access.

Table 3.3 *Metadata-Layer Permission Requirements for Selected Tasks*

Task	Secured Data Folder	Secured Library Object	Secured Table Object
Bind a library to metadata*	WriteMemberMetadata	-	-
Unbind a metadata-bound library	WriteMemberMetadata	WriteMetadata	-
Add tables to a metadata-bound library (for example, using the COPY procedure)	-	Create Table	Alter Table**

* If the metadata identity that is used to bind a physical library to metadata doesn't have effective metadata-layer grants of the data permissions, explicit grants are added to the new secured library object when it is created.

** Alter Table is required on any target table object that is overwritten. If there is no such table object, then Create Table is required on the parent library object.

Additional Considerations

The following additional requirements apply:

- The metadata identity under which authorization decisions are requested must have the ReadMetadata permission for all applicable objects.
- The host identity under which physical data is accessed must meet the usual host-layer requirements, such as the following:
  - In order to view physical data, the host identity must have host-layer Read access to the data.
  - In order to add, update, or delete a physical table, the host identity must have host-layer Write access to the target directory or file (in accordance with the host system's requirements).
  - In order to create, modify, or delete a metadata-bound library, the host identity must have host-layer control of the physical directory. For host-specific details, see "Requirement for Host-Layer Control" on page 42.
- When accessing data from a client such as SAS Web Report Studio, users must also have appropriate permissions for the traditional library and table objects that are used to locate the data.

Requirement for Host-Layer Control

For actions other than validation, the requesting identity must have host-layer control of the target physical directory. This ensures that only users who have host control can bind, unbind, or modify passwords for a physical library.

The requesting SAS session (or workspace server) must run under a privileged host account as follows:

- On UNIX, the account must be the owner of the directory.
- On Windows, the account must have Full Control of the directory.

- On z/OS, for UNIX file system libraries, the account must be the owner of the directory.
- On z/OS, for direct-access bound libraries, the account must have RACF ALTER access authority to the library data set.

Identity in Authorization Evaluations

In general, metadata-layer access is evaluated against the metadata identity of the requesting user (the client), and host-layer access is evaluated against the server process identity (or, for a Base SAS session, the identity under which the session was initiated).

The following list documents some exceptions:

- If access is through a SAS/CONNECT server that did not receive the requesting user’s metadata identity from the client session, metadata-layer authorization checks are made against the SAS/CONNECT server’s metadata identity. This unusual circumstance occurs if the server runs with the NOCONNECTMETACONNECTION option and is not a trusted peer of the metadata server.
- If access is through a SAS/SHARE server that cannot impersonate the requesting user on a connection to the metadata server, and the target data is a remote view, then metadata-layer authorization checks are made against the SAS/SHARE server’s metadata identity. For example, metadata-layer authorization checks are made against the SAS/SHARE server’s metadata identity if that server runs with the AUTHENTICATE=OPTIONAL option and no client identity is established.

For access through a SAS/SHARE server that does impersonate the requesting user, an additional consideration is which client identity matters—the one that connects to the SAS/SHARE server, or the one that issues the LIBNAME statement. The following list provides details:

- In both of the following cases, checks are under the metadata identity that corresponds to the user ID with which the client host authenticates to the SAS/SHARE server:
  - Access is through view files and the REMOTEVIEW option is set to YES (the default value).
  - Access is from a third-party client (such as ODBC, OLEDB, or JDBC).
- Otherwise, checks are under the metadata identity in the client at the time that the LIBNAME statement is issued.

Passwords for Metadata-Bound Data

In addition to being tied to a particular metadata object, a metadata-bound library also has a set of associated passwords. These passwords serve a secondary role, enabling administrators to recover metadata (for example, in the event that they accidentally delete a secured library object from the metadata) and ensuring that authorization decisions come from only valid sources.

Here are some details about these passwords:

- The passwords are recorded both in the physical data and in metadata.

- The passwords are always stored and transmitted in encrypted formats. Even if an encrypted password is captured, it can't be submitted as a password value in SAS code.
- The passwords do not create access distinctions. For simplicity, we recommend that you use PW= to set a single password value, rather than specifying different password values using READ=, WRITE=, and ALTER=.

  However, each plain text password value can be only eight characters long. You might choose to set different password values (using READ=, WRITE=, and ALTER=) for greater security. In effect, setting different values can create a 24-character password.
- Passwords that you supply in SAS Management Console are encoded or encrypted in transit, in accordance with your configuration.
- You can use the PWENCODE procedure to encode passwords for use in the AUTHLIB procedure. If you supply an encoded password, enclose it in quotation marks. All other encryption of the password (both in-transit and on-disk) occurs automatically. An encrypted password that is captured in transmission cannot be used.
- End users never have to supply these passwords, so they should neither know, nor have access to, the password values.
- Use of metadata-bound libraries doesn't involve prompting end users for secured library passwords.
- When it communicates authorization decisions, the metadata server supplies passwords that match passwords that are stored with the physical data, in order to prove that it is the valid source for those decisions.
- In order to use SAS to copy a metadata-bound table, you must have Read access (the Select permission) for the source table. The source table's password is not applied to the new (output) table. If the new table is added to a metadata-bound library, that library's password is applied to it. If the new physical table is added to a traditional library, the new table is not protected as a secured table or with passwords retained from the source table.
- In general, all metadata-bound tables within a particular metadata-bound library share the same set of passwords. Each library's passwords are automatically applied to the tables within that library. However, the following exceptions exist:
  - Physical tables that existed in the operating system directory, with passwords, at the time that their parent metadata-bound library was created retain their pre-existing passwords. Such physical tables are not secured by metadata unless you modify their passwords to match the parent library's passwords.
  - Physical tables that you copy into a metadata-bound library using operating system commands yield the following results:
    - If the original tables are metadata-bound tables, the copied tables are protected by the same metadata-bound library that protected the original tables. The act of copying the physical tables into another metadata-bound library doesn't cause a change to the protections.
    - If the original tables are not metadata-bound tables, the copied tables are not secured by metadata unless you explicitly apply the library passwords to them.

See Also

"Avoiding Mixed States" on page 37

Auditing for Metadata-Bound Libraries

Which Events Can Be Logged?

For metadata-bound libraries, certain events can be logged to a system-wide logging facility.

The following table summarizes the events that can be logged:

Table 3.4 *Logged Events for Metadata-Bound Data*

Category (Logger)	**Logged Events**
Authorization failure records (Audit.Data.MetaBoundLib.PermDenied)	A user attempts to access a metadata-bound table to which the user has insufficient effective permissions in the metadata layer. Access is not allowed.
Misalignment issue records* (Audit.Data.MetaBoundLib.AuthAudit)	A user accesses a metadata-bound table that is located within a traditional (unbound) library.
	A user accesses a traditional (unbound) table that is located within a metadata-bound library.**
	A user accesses a metadata-bound table whose security location reference doesn't match the security location reference of its parent library.
	A user accesses a metadata-bound table whose security name reference doesn't match the corresponding secured table object. In other words, there is a mismatch of names (the correspondence is determined by another identifier).
	A user attempts to access a metadata-bound table whose passwords do not match the passwords of the corresponding secured library object. In other words, there is a mismatch of passwords. Even if the user's metadata-layer permissions are sufficient, access is not allowed.

* The misalignment issue records do not specify who created the issue; these records just indicate that the issue exists at the time that access is requested.

** This is the most important event to audit, because it might indicate an earlier circumvention of security. See "Detecting a Circumvention of Update Security" on page 45.

Detecting a Circumvention of Update Security

A user who has only Read access to a metadata-bound table might be able to update the table by using the following process:

1. Use SAS to copy the metadata-bound table to an unsecured library.
2. Update the data.

3. Use a host command to copy the table back to its original metadata-bound library.

Because the user used a host command to copy the table back to its original secured library location, the updated table is no longer bound to its corresponding secured table object.

Note: The preceding scenario can occur when you enable a user to update some, but not all, of the tables within a metadata-bound library. In order to perform that task, the user needs host-layer Write access to the entire library. This enables the user to perform step 3 in the preceding list, regardless of whether the user has metadata-layer Update access to the target metadata-bound table.

The audit record that is written when a user accesses a traditional (unbound) table that is located within a metadata-bound library provides an indication that someone might have used the preceding process to circumvent security.

Note: These audit records can also indicate that you have not followed the recommended practice of ensuring that all tables within a metadata-bound library are bound to that library.

Audit Record Content and Layout

Here is an example of an authorization failure record:

DateTime=2012-02-15T17:48:28,671, Userid=JOE@COMPANY, StepName=DATASTEP, Action=Read, LoginId=JOE@COMPANY, IdentityName=Joe, Libref=REVENUE , OSLibraryPath=\\machine.company.com\Data\Revenue, MemberName=CSV, MemberType=VIEW , DataSetInfoSecuredLibrary=/System/Secured Libraries/Data/, DataSetInfoSecuredLibraryGuid=5200B831-50A1-4E66-92CD-AD86ACDB43B7, DataSetInfoSecuredTableName=CSV.VIEW, DataSetInfoSecuredTableGuid=5BE37390-986F-45B4-8227-F3653C79768A, LibraryInfoSecuredLibrary=/System/Secured Libraries/Data, LibraryInfoSecuredLibraryGuid=5200B831-50A1-4E66-92CD-AD86ACDB43B7, RequiredPermission=Select, UserEffectivePermissions=None, Message=ERROR: JOE@COMPANY as Joe is not authorized to read data set REVENUE.CSV.VIEW. Select permission is required.

Here is an example of a misalignment issue record that indicates a possible security concern:

DateTime=2012-02-15T17:48:21,201, Userid=JOE@COMPANY, StepName=DATASTEP, RecType=201, LoginId=JOE@COMPANY, IdentityName=omitest, Libref=METAOMI , OSLibraryPath=\\machine.company.com\Data, MemberName=D, MemberType=DATA , DataSetInfoSecuredLibrary=, DataSetInfoSecuredLibraryGuid=, DataSetInfoSecuredTableName=, DataSetInfoSecuredTableGuid=, LibraryInfoSecuredLibrary=/System/Secured Libraries/Data, LibraryInfoSecuredLibraryGuid=ACFAF468-B77E-4DF2-BB64-D7342F2CB1CE, PasswordDifferences=, UserEffectivePermissions=, Message=WARNING: Data set METAOMI.D.DATA is not bound to a secured table object, but it resides in a directory that is bound to a secured library object. The data set might have existed in this directory before the library was bound, or the data set might have been copied to this directory with a host copy utility.

TIP The layout of an audit record is determined by conversion patterns within your logging configuration file.

See Also

- "Administering Logging for SAS Servers" in *SAS Intelligence Platform: System Administration Guide*
- "Audit Messages for Metadata-Bound Libraries" in *SAS Logging: Configuration and Programming Reference*

Considerations for Data File Encryption

Overview

In general, the process for encrypting metadata-bound tables is the same as for encrypting traditional tables, except for the following:

- The Read password is obtained from the secured library object (in metadata), instead of being supplied in SAS code.
- Beginning in the first maintenance release for SAS 9.4, AES encryption keys can be obtained from the secured library object (in metadata), instead of being supplied in SAS code.
- Beginning in the first maintenance release for SAS 9.4, you can specify that encryption is required for the tables in a library. When tables are created or modified, the data is automatically encrypted using the specified method.

The following topics document aspects of data encryption that are specific to metadata-bound libraries.

Using AES Encryption with Metadata-Bound Libraries

When you create or modify a metadata-bound library and specify AES encryption, you can specify an encryption key to be stored in the library's metadata. The stored key is used to attempt to open the library's AES-encrypted data sets when no key is supplied by the user. If a table has a different encryption key, it cannot be opened unless the user supplies the correct key.

You can specify that AES encryption is required for the tables in a library. In this case, the stored encryption key is used to encrypt every data set that is bound in the library. You cannot specify different encryption keys for individual tables if encryption is required.

If encryption is not required, the stored key is used to encrypt new tables when AES encryption is specified in SAS code but no key is supplied.

To simplify administration, we strongly recommend that all AES-encrypted tables within a metadata-bound library be encrypted with the same key.

Making Security-Related Changes to an Encrypted Table

When you modify the encryption options for a metadata-bound library or table, a copy-in-place approach is used to re-encrypt any tables that were originally encrypted. For example, this approach is used to apply a new encryption method or a new AES encryption key, to automatically encrypt tables when encryption is required, and to incorporate a new READ password into keys for default SAS encryption.

The following actions occur in the copy-in-place process:

1. The original table is renamed to the reserved data set name __TEMP_ENCRYPT_FILE_NAME__.
2. The temporary table is copied back to the original table name (which causes the data to be re-encrypted).

   Note: In order for this step to be completed, the metadata-layer Create Table and Alter Table permissions are required.
3. The temporary table (__TEMP_ENCRYPT_FILE_NAME__) is deleted.

The preceding process occurs automatically when it is needed. No action on your part is required.

See Also

- "Example 10: Binding a Library When Existing Data Sets Are SAS Proprietary Encrypted" on page 114
- "Example 11: Binding a Library When Existing Data Sets Are AES-Encrypted" on page 116
- "ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*
- "ENCRYPT= Data Set Option" in *SAS Data Set Options: Reference*
- "SAS Data File Encryption" in *SAS Language Reference: Concepts*

Considerations for Renaming Physical Tables

If you need to rename a physical table within a metadata-bound library, use the DATASETS procedure or the Explorer window. If a secured table object with the new name already exists within the secured library object, access to the physical table is governed by permissions for that secured table object. If no secured table object exists with the new name, a new secured table object is created. Any direct access controls (explicit permissions and directly applied access control templates) on the secured table object with the old name are set on the new secured table object.

Object Creation, Location, and Inheritance

About This Topic

This topic highlights key differences between the following sets of metadata objects:

- secured library and table objects
- traditional library and table objects

Each set of objects serves a distinct purpose, so each has distinct characteristics.

Object Creation

You create secured library and table objects by binding physical data to metadata. This adds security information to the physical data and creates corresponding metadata objects at the same time. You use either SAS Management Console or the AUTHLIB procedure to perform this task.

You create traditional library and table objects by registering physical data (using SAS Management Console). You can also register traditional tables through SAS code (using the METALIB procedure). Registering data in metadata has no impact on the physical content of the data.

Metadata Location

Traditional library and table objects can be located in any regular metadata folder. For greater security, metadata locations for secured library and table objects are constrained as follows:

- These objects can be located only within a **`/System/Secured Libraries`** branch in the SAS Folders tree. You can put secured library objects in subfolders within a **`/System/Secured Libraries`** branch. You cannot bind physical libraries to secured library objects in other locations.
- Each secured library object is located within a secured data folder, which is a special type of metadata folder that exists only under a **`/System/Secured Libraries`** branch.
- Each secured table object is located within a secured library object.

Access Control Inheritance

For secured library and table objects, metadata-layer access control inheritance is as follows:

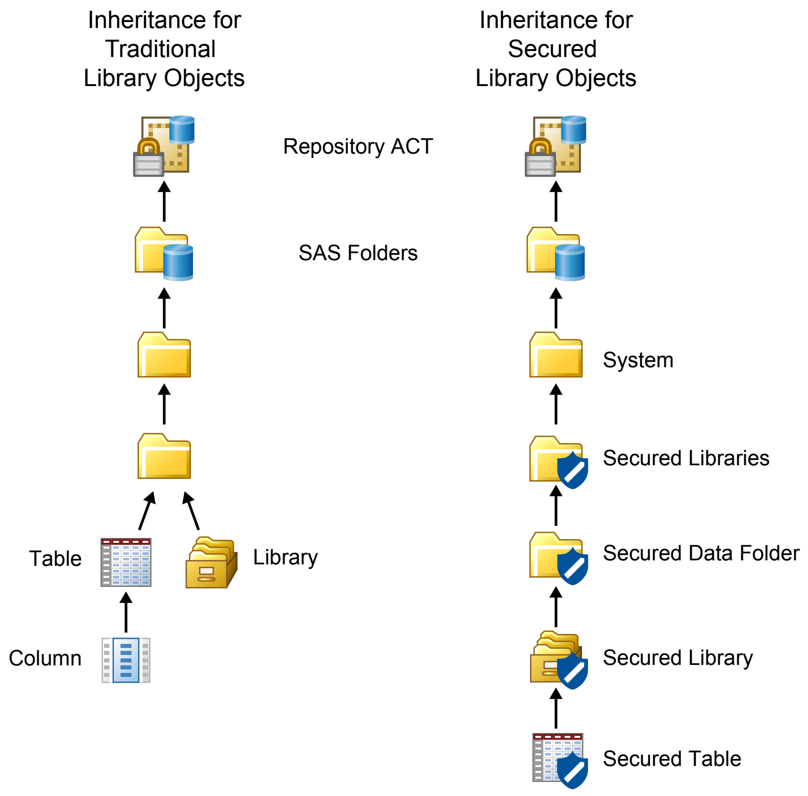
- Each secured library object inherits from its parent secured data folder.
- Each secured table object inherits from its parent secured library.

This inheritance pattern differs from that of traditional library and table objects, which both inherit directly from their parent folders. The difference in inheritance pattern reflects the distinct purpose of each set of objects:

- Secured library and table objects serve as bind targets for physical data (providing access control for the data), so their inheritance pattern follows the inheritance pattern of the physical data.
- Traditional library and table objects serve as pointers to physical data (enabling clients to locate data through metadata), so their inheritance pattern is folder oriented.

The following figure depicts examples of inheritance for both types of libraries:

Figure 3.1 *Examples of Inheritance for Traditional and Secured Library Objects*



Security Information in Metadata-Bound Data

In addition to the usual directory and file content, additional security-related information is stored in the host for metadata-bound data.

- For a physical library, the security information consists of a subdirectory and file. The corresponding metadata object is called a secured library object.

  z/OS Specifics
  On z/OS, the security information for a UNIX file system (UFS) library is stored as described above. However, the security information for a z/OS direct-access bound library is instead stored within the bound library data set itself. For this reason, z/OS sites that choose to use metadata-bound libraries might prefer the z/OS direct-access bound library implementation to the UFS library implementation. z/OS sequential-access bound libraries cannot be bound to metadata.

- For a physical table, the security information consists of information in the header. The corresponding metadata object is called a secured table object.

The security information includes the following:

- an indication that access to the physical data must be authorized by the metadata server
- a record of the metadata location of the metadata objects that correspond to the physical data

The security-related information itself is host protected.

SAS Language Reference for Metadata-Bound Libraries

About This Topic

Administrators who use the SAS code method to set up and maintain metadata-bound libraries need to know the SAS language reference information for metadata-bound libraries.

Note: For data consumers, interaction with metadata-bound libraries is mostly transparent: each access request succeeds or fails based on the requesting user's permissions. However, a connection to the metadata server is required in order to access metadata-bound data, so users who make direct requests (for example, through a LIBNAME statement) do have to facilitate that connection.

Metadata Server Connection Options

An administrator connects from a SAS session to a target metadata server in order to set up or maintain metadata-bound data using SAS code.

Metadata server connection information can be provided in the following ways:

- Specify options in a configuration file. For example, you might add the following lines to a configuration file:

  ```
  -METAPORT  8561
  -METAREPOSITORY "foundation"
  -METASERVER "a123.us.company.com"
  -METAUSER  "myuserid"
  -METAPASS  "Pwd1"
  ```

- Specify options in an OPTIONS statement. For example, you might add the following OPTIONS statement to your SAS program or autoexec.sas file:

  ```
  options
       metaport=8561
       metarepository="foundation"
       metaserver="a123.us.company.com"
       metauser="myuserid"
       metapass="Pwd1";
  ```

- Reference a stored connection profile (using the METACONNECT= and METAPROFILE options).
- Provide connection information interactively. If sufficient information is not otherwise available, you can be prompted for connection information.

TIP If a port is not specified, the default metadata server port (8561) is used. If a repository is not specified, the foundation repository is used. If Integrated Windows authentication (IWA) is configured, you don't have to supply a user ID or password.

For additional information, see "Connection Options " in *SAS Language Interfaces to Metadata*.

LIBNAME Statement Options

AUTHADMIN

An administrator specifies the AUTHADMIN=YES option in a LIBNAME statement in order to access a metadata-bound library for which corresponding metadata is corrupted, misconfigured, or missing. If the administrator specifies AUTHADMIN=YES in a LIBNAME statement, the administrator must somehow supply the metadata-bound library password (or passwords) in order to access the files. In this specialized context, metadata-layer permissions are not used to determine access.

Note: Since some access requests do not have a way to specify passwords, the administrator can use the AUTHPW option (or related options) in the LIBNAME statement to provide the metadata-bound library password (or passwords). See "AUTHPW (and Related Options)" on page 52.

Note: The AUTHADMIN=YES option is accepted only if the account under which your SAS session runs has host-layer control of the target physical library. This ensures that only users who have host control for a particular directory can use this option against that directory. The host-specific details for this requirement are the same as for the AUTHLIB procedure statements (except the REPORT statement). See "Requirements" on page 10.

We recommend that you use AUTHADMIN=YES when you are repairing any inconsistencies between physical data and its corresponding secured library and secured table objects in metadata. We do not recommend that you use AUTHADMIN=YES in other circumstances. The purpose of this option is to enable an administrator to establish a libref for a library that is in need of repair.

We also recommend that you reassign the library (without AUTHADMIN=YES) after the repairs are made.

Note: In the current release, the REPAIR DELETE LOCATION statement is a production feature. Other actions in the REPAIR statement are preproduction features.

For additional information, see "LIBNAME Statement" in *SAS Statements: Reference*.

AUTHPW (and Related Options)

An administrator can choose to specify AUTHPW=*password-value* in a LIBNAME statement as an alternate method for making the metadata-bound library password available to later requests.

A password that is supplied by the AUTHPW option is used only if both of the following circumstances exist:

- AUTHADMIN=YES is specified in the LIBNAME statement.

  Note: Use of AUTHADMIN=YES does not necessitate use of AUTHPW. You are not required to specify metadata-bound library passwords in a LIBNAME statement. However, you should not specify metadata-bound library passwords in a LIBNAME statement that doesn't also specify AUTHADMIN=YES.

- The correct password for the target metadata-bound library is not otherwise available (either no password is supplied or the supplied password is invalid).

In such requests, the value from the AUTHPW option is validated against the password within the physical table. An error is returned if the passwords do not match.

Requests to access metadata-bound tables within a library that was assigned with AUTHADMIN=YES must meet at least one of the following criteria:

- The request comes from the AUTHLIB procedure, which has a supplied password.
- The request explicitly supplies the password.
- The library was also assigned with the AUTHPW option, which supplies the password.
- The request is interactive and the user can supply the password when prompted.

Note: For conciseness, the preceding discussion assumes that there is only one password for the target metadata-bound library. For a metadata-bound library that has two or three distinct passwords, you must specify each password (using the AUTHREAD, AUTHWRITE, and AUTHALTER options as appropriate) instead of using the AUTHPW option on its own.

For additional information, see "LIBNAME Statement" in *SAS Statements: Reference*.

See Also

"Passwords for Metadata-Bound Data" on page 43

AUTHLIB Procedure

An administrator can use the AUTHLIB procedure to set up and maintain metadata-bound libraries. The AUTHLIB procedure is documented in the *Base SAS Procedures Guide*. As a convenience for the reader, documentation for the AUTHLIB procedure is also reproduced in this document. (See Appendix 2.)

Chapter 4
Troubleshooting for Metadata-Bound Libraries

Facilitate End-User Access

This topic provides guidance for enabling end users to access metadata-bound data.

Issue: User can't access the metadata server.

Resolution:

Complete the following steps:

- Explain to the user how to connect to the target metadata server from his or her SAS session. See "Metadata Server Connection Options" on page 51.
- In SAS Management Console, make sure the user is correctly registered in the metadata repository. See the topic "Introduction to User Administration" in the *SAS Management Console: Guide to Users and Permissions*.

Issue: User is not authorized to access the data.

Resolution:

Complete the following steps:

- Verify that the user is using the second maintenance release of SAS 9.3 (or a later release). Users cannot access metadata-bound data from earlier releases of SAS.
- In SAS Management Console, make sure the user is correctly registered in the metadata repository. See the topic "Introduction to User Administration" in the *SAS Management Console: Guide to Users and Permissions*.
- In SAS Management Console, verify that the user has the necessary permissions on the secured table object that corresponds to the target metadata-bound table. If the user is attempting to access data from a client that uses metadata in order to locate data, verify that the user has sufficient permissions on the traditional table object. See "Verifying Access to Metadata-Bound Data" on page 30.
- Make sure that the host account under which the data is retrieved has host-layer access to the data. See "Identity in Authorization Evaluations" on page 43.

- In the metadata server log, make sure that the user is connecting to the metadata server under the expected metadata identity. See "Default Locations for Server Logs" in *SAS Intelligence Platform: System Administration Guide*.

Replace Missing Metadata Objects

This topic provides guidance for replacing missing secured data folders, secured library objects, and secured table objects.

Note: Some of the resolutions in this section use the REPAIR statement, most of which is a preproduction feature in the current release. These resolutions should be used only under advice and direction from SAS Technical Support. As an alternative to using the REPAIR statement, you can create a new physical library, bind that library to metadata, and use the SAS COPY procedure to copy your data into the new library. See "REPAIR Statement" on page 91.

Issue: A secured data folder is missing.

Resolution:

Use one of the following approaches:

- Use the metadata promotion tools to import the folder (and its contents) from an existing package. See "Promotion Tools Overview" in *SAS Intelligence Platform: System Administration Guide*.
- Re-create the folder in SAS Management Console. On the **Folders** tab, navigate to the appropriate location (under a repository's `/System/Secured Libraries` branch), right-click, and select **New** ⇨ **Folder**.
- Recover metadata from a metadata server backup. See "Backing Up and Recovering the SAS Metadata Server" in *SAS Intelligence Platform: System Administration Guide*.

Issue: A secured library object is missing.

Resolution:

Use one of the following approaches:

- Use the metadata promotion tools to import the secured library object (and its secured table objects) from an existing package. See "Promotion Tools Overview" in *SAS Intelligence Platform: System Administration Guide*.
- Use the REPAIR statement of the AUTHLIB procedure, with the ADD action and the METADATA option. This approach should be used only under advice and direction from SAS Technical Support.
- Recover metadata from a metadata server backup. See "Backing Up and Recovering the SAS Metadata Server" in *SAS Intelligence Platform: System Administration Guide*.

Issue: A secured table object is missing.

Resolution:

Use one of the following approaches:

- Use the metadata promotion tools to import the secured table object's parent library from an existing package. The import overwrites the current secured library object and its child objects, and it re-creates any missing secured table objects. See "Promotion Tools Overview" in *SAS Intelligence Platform: System Administration Guide*.
- Use the REPAIR statement of the AUTHLIB procedure, with the ADD action, the METADATA option, and TABLESONLY=YES. This approach should be used only under advice and direction from SAS Technical Support.
- Recover metadata from a metadata server backup. See "Backing Up and Recovering the SAS Metadata Server" in *SAS Intelligence Platform: System Administration Guide*.

Realign Security Location Information

This topic provides guidance for realigning missing or corrupted security location information (the security binding information that is stored with the physical data).

Note: The resolutions in this section use the REPAIR statement, most of which is a preproduction feature in the current release. These resolutions should be used only under advice and direction from SAS Technical Support. As an alternative to using the REPAIR statement, you can create a new physical library, bind that library to metadata, and use the SAS COPY procedure to copy your data into the new library. See "REPAIR Statement" on page 91.

Issue: Physical library (or table) security location information is corrupted.

Resolution:

Use the REPAIR statement with the UPDATE action and the LOCATION option.

Issue: Physical library (or table) security location information is missing.

Resolution:

Use the REPAIR statement with the ADD action and the LOCATION option.

Appendix 1
Security Impact of Moving Tables

About This Appendix

If you need to copy or move metadata-bound libraries in the file system, we recommend that you use SAS, not operating system commands.

Note: An exception to this guideline is that using a host copy command to back up or restore physical data to the same directory is not problematic.

For example, if you use SAS (the COPY procedure) to copy a table, the new table takes on the nature of its parent library as follows:

- Copying a table into a metadata-bound library yields a metadata-bound table.
- Copying a table into an unbound library yields an unbound table.

The following topics depict the security impact of copying or moving metadata-bound libraries and tables.

Adding Physical Tables to a Metadata-Bound Library

The examples in this topic use the copy action. The same results occur if physical tables are moved, except that the original physical tables are deleted.

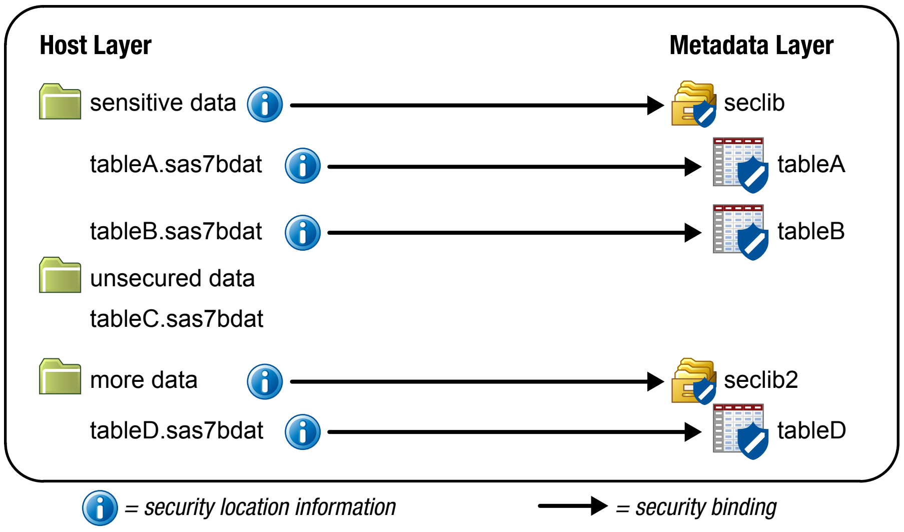
Introduction

When you copy or move physical tables into a metadata-bound library, the result varies depending on the following factors:

- whether you use SAS or host commands to perform the action
- whether the original tables are protected with passwords that differ from the password of the target library

The following figure depicts an initial state for the examples in this topic.

Figure A1.1 *Example: Initial State*

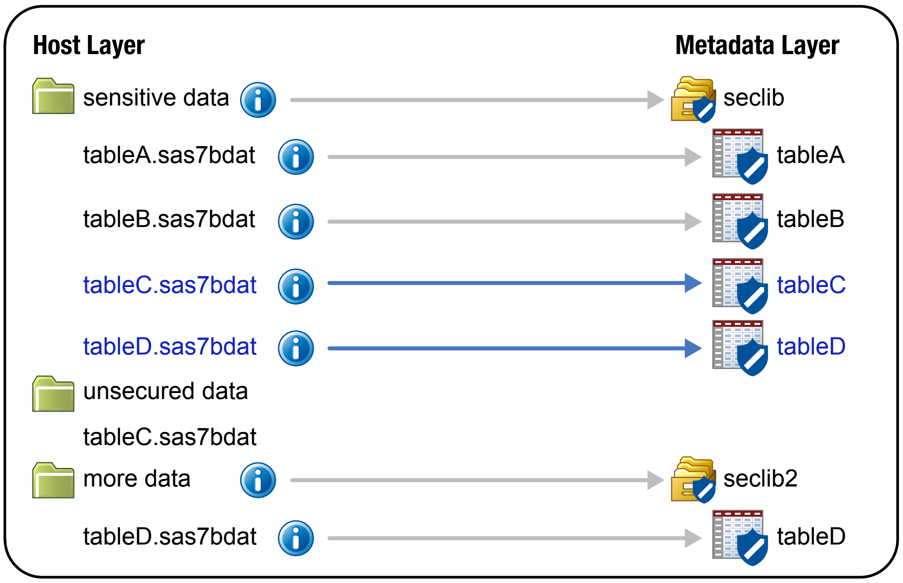


Using SAS

If you use SAS to add physical tables to a metadata-bound library, the added tables are automatically secured. The password of the target library is applied to the added tables, and corresponding secured table objects are created in metadata.

The following example depicts the impact of using the COPY procedure to copy tableC and tableD into the sensitive data folder.

Figure A1.2 *Example: After a SAS Copy*



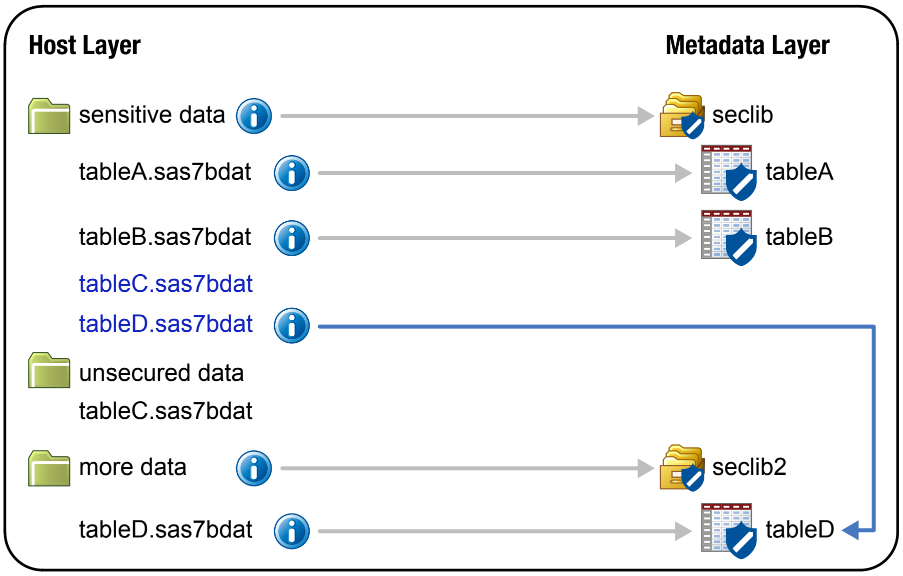
Notice that security information and bindings are generated for the added tables and that corresponding secured table objects are automatically created in metadata. With a SAS copy, both of the added tables are automatically secured by their parent library.

Using Host Commands

If you use a host copy command to add physical tables to a secured library, the added tables are not automatically secured. If you create a host copy of an unsecured table, the copy is unsecured. If you create a host copy of a secured table, the copy retains the security information and binding of the original table.

The following example depicts the impact of using host commands to copy two physical tables (tableC and tableD) into the sensitive data folder.

Figure A1.3 *Example: After a Host Copy*



Notice that the copied tableC is not secured, and that the copied tableD has the same security binding as the original tableD.

Copying Metadata-Bound Tables to a Traditional Library

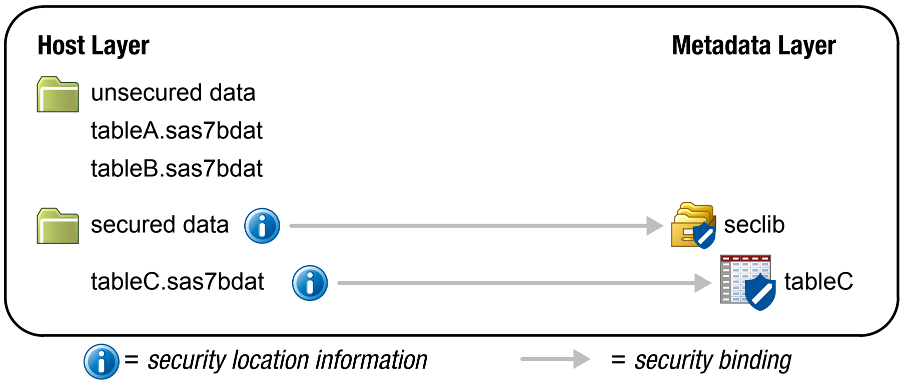
The examples in this topic use the copy action. The same results occur if tables are moved, except that the original physical tables are deleted.

Introduction

When you copy or move metadata-bound tables into a traditional physical library, the result varies depending on whether you use SAS or host commands to perform the action.

The following figure depicts an initial state for the examples in this topic.

Figure A1.4 *Example: Initial State*

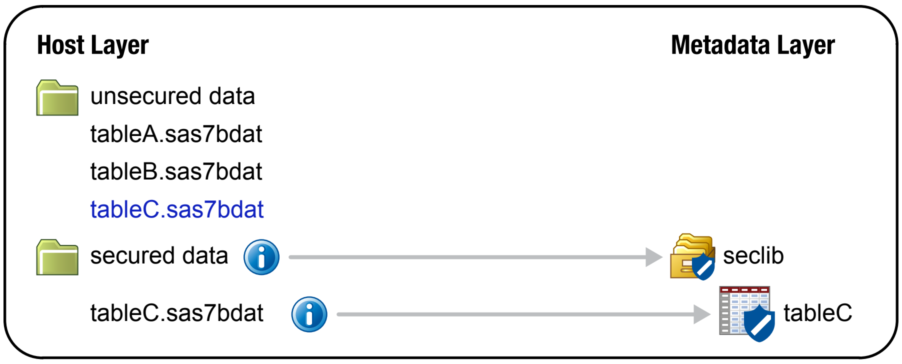


Using SAS

If you use SAS to add a metadata-bound table to a traditional physical library, the added table is not secured. It takes on the unsecured nature of its new parent library. For this reason, SAS requires that you have adequate metadata-layer permissions to the original table in order to copy (or move) it. See "Permissions for Metadata-Bound Data" on page 39.

The following example depicts the impact of using the COPY procedure to copy a physical table (tableC) into the unsecured data folder.

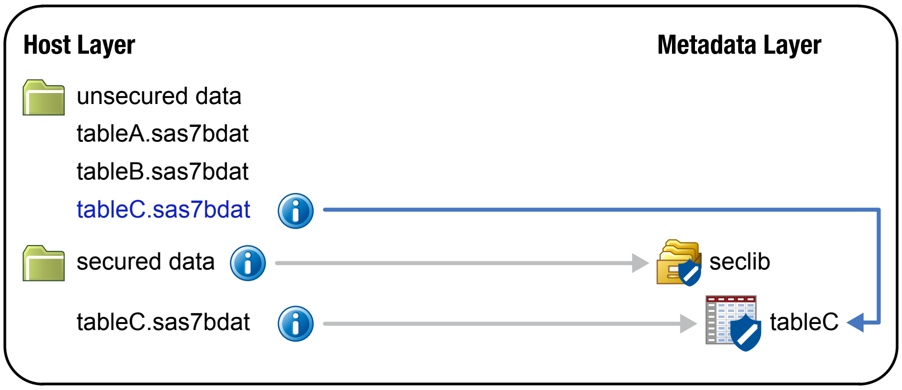
Figure A1.5 Example: After a SAS Copy



Using Host Commands

If you use a host command to add a metadata-bound table to a traditional physical library, the added table is secured. It is bound to the same metadata object that the original table is bound to. With a host command, SAS isn't involved, so metadata-layer permissions can't be checked. Thus, the original security information and binding is preserved.

Figure A1.6 Example: After a Host Copy



Appendix 2
AUTHLIB Procedure

Overview: AUTHLIB Procedure

The AUTHLIB procedure is a utility procedure that manages metadata-bound libraries. With PROC AUTHLIB, you can do the following:

- create a metadata-bound library by binding a physical library to metadata within a SAS Metadata Repository
- modify password and encryption key values for a metadata-bound library
- purge replaced password and encryption key values that are also known as metadata-bound library credentials
- repair metadata-bound libraries by recovering security information, secured library objects, and secured table objects
- remove the physical security information and metadata objects that protect a metadata-bound library
- report inconsistencies between physical library contents and corresponding metadata objects within a specified metadata-bound library

Users cannot access metadata-bound data sets from any release of SAS prior to the second maintenance release of 9.3.

Note: For a z/OS direct-access bound library that has been bound to metadata, the constraint is slightly broader. Neither the library nor any of its members can be accessed by earlier releases of SAS.

Concepts: AUTHLIB Procedure

Metadata-Bound Library

A metadata-bound library is a physical library that is tied to a corresponding metadata secured table object. Each physical table within a metadata-bound library has information in its header that points to a specific metadata object. The pointer creates a security binding between the physical table and the metadata object. The binding ensures that SAS universally enforces metadata-layer access requirements for the physical table —regardless of how a user requests access from SAS. See Chapter 1, "Overview of Metadata-Bound Libraries," on page 1.

Using Metadata-Bound Library Passwords

A metadata-bound library contains a single set of passwords that are stored in the secured library object. This set of passwords is added to all data sets that are created in the metadata-bound library. These passwords are not used to authorize user access to the data. They are used to authorize administrator access to repair the binding of physical data to the secured library or table metadata objects. The passwords are also validated in the process of authorizing a user's access to a data set. They do not determine the permissions that any user is authorized to have.

The metadata-bound library passwords are intended to be known only by the administrators of the metadata-bound library. Knowledge of these passwords is required to restore or re-create secured library and secured table objects in a SAS Metadata Server for data sets in a data library that have lost their previously recorded metadata objects and permissions.

The metadata-bound library passwords also prevent a user from exporting the secured library and secured table objects from a SAS Metadata Server and then importing them to a SAS Metadata Server that an unauthorized user created and controls. This prevents the unauthorized user from using such objects where the user has modified the permissions.

The metadata-bound library passwords are always stored and transmitted in encrypted formats. The encrypted password is not usable to access the data if it is captured from a transmission and presented to SAS as a password value in the SAS language. Administrators might choose to use the PWENCODE procedure to encode the passwords for use in a PROC AUTHLIB statement. Using an encoded password prevents a casual observer from seeing the clear-text password in the PROC AUTHLIB statements that the administrator types.

There are three passwords in the metadata-bound library set that correspond to the READ=, WRITE=, and ALTER= passwords of SAS data sets. For greater simplicity in administration of metadata-bound libraries, it is recommended that you use the PW= option in PROC AUTHLIB statements to specify a single password value. In the context of metadata-bound libraries, the READ=, WRITE=, and ALTER= options do not Create access distinctions. If you are concerned that a single eight character password does not meet your security requirements, then you can choose to set three different password values (using READ=, WRITE=, and ALTER=). Setting different values for these three options can create a 24-character password. However, you must keep track of all password values that you have assigned to a metadata-bound library. You must specify the passwords to do the following:

- unbind the library
- modify the passwords
- repair any inconsistencies in the binding information between what is recorded in the physical files and the actual metadata objects

For more information, see "Setting and Modifying Metadata-Bound Library Passwords" on page 68.

TIP All password values must be valid SAS names with a maximum length of 8 characters.

CAUTION:

If you lose the password (or passwords) for a metadata-bound library, then you cannot unbind the library or change its passwords. Be sure to keep track of passwords that you assign in the CREATE and MODIFY statements.

Setting and Modifying Metadata-Bound Library Passwords

The metadata-bound library passwords are set in the CREATE statement and can be changed with the MODIFY statement. The passwords stored in data sets in the operating system library can be changed by those statements and subordinate TABLES statements. The passwords stored in the data sets can also be changed if the library is unbound from the metadata with a REMOVE statement.

All of the password options in the CREATE, MODIFY, TABLES, and REMOVE statements accept a syntax where two values can be specified separated by a slash (/) (for example, **`PW=password-value/new-password-value`**). For CREATE and MODIFY statements, a password value to set in the metadata or data sets is obtained from the password value before the slash (/) if no new password value is specified after the slash (/). The same is true for the REMOVE statement with the additional possibility of specifying the slash (/) and no new password value to indicate that the password should be removed from the data sets during the unbind process. However, note that if the CREATE, MODIFY, or REMOVE statement also specifies TABLESONLY=YES, then any new password values on those statements are ignored.

In general, you do not specify a new password value in a TABLES statement following a CREATE or MODIFY statement. The new value is obtained from the metadata to which the data set is bound or being bound. You can specify a new password value in TABLES statements following a REMOVE statement if you want different data sets to have unique passwords. In that case, you follow these steps:

1. Change the password for the data sets using a REMOVE statement with TABLESONLY=YES and an individual TABLES statement for each unique password.
2. Remove the metadata-bound library with a REMOVE statement without TABLESONLY=YES.

See Also

- "Example 1: Binding a Physical Library That Contains Unprotected Data Sets" on page 100
- "Example 2: Binding a Physical Library That Contains Password-Protected Data Sets" on page 102
- "Example 3: Binding a Library When Existing Data Sets Are Protected with the Same Passwords" on page 104
- "Example 4: Binding a Library When Existing Data Sets Are Protected with Different Passwords" on page 105
- "Example 5: Changing Passwords on Data Sets" on page 107
- "Example 6: Changing Metadata-Bound Library Passwords" on page 109

Encrypted Data Set Considerations

Some data sets in metadata-bound libraries might be encrypted either with the SAS Proprietary Encryption or Advanced Encryption Standard (AES) algorithm. SAS Proprietary Encryption is specified as ENCRYPT=YES when the data set is created. AES encryption is specified as ENCRYPT=AES and an ENCRYPTKEY= key value when the data set is created. Special considerations apply for these encrypted data sets when processed by the AUTHLIB procedure.

CAUTION:

AES encryption is supported only in SAS 9.4 and later releases. Do not use AES encryption if the data sets need to be accessible by the second maintenance release of SAS 9.3.

SAS Proprietary Encrypted Data Sets

SAS Proprietary Encryption uses the READ password of the data set as part of the encryption key. Since all metadata-bound data sets in the library share the same set of passwords, it is not necessary to specify the READ password when accessing the file. However, when the READ password is modified on the data set in a CREATE, MODIFY, or REMOVE statement, the data must be re-encrypted with the new password value. This process is done automatically for you in the 9.4 release with a copy-in-place operation. For more information about the copy-in-place operation, see "Copy-In-Place Operation" on page 99.

AES-Encrypted Data Sets

There are two ways to access an AES-encrypted data set:

- the user must provide the ENCRYPTKEY= key value to open the data set
- the administrator must have recorded an optional or required encryption key for the metadata-bound library with the ENCRYPTKEY= option in the CREATE or MODIFY statement

Note: The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.

By recording an optional or required ENCRYPTKEY= key value for the metadata-bound library, the metadata becomes a key store for the encryption key value. Like password values, the key value is always stored and transmitted in encrypted formats. The encrypted key value is not usable to access the data if it is captured from a transmission and presented to SAS as an encryption key value in the SAS language. For more information, see "Setting and Modifying Metadata-Bound Library Encryption Options" on page 70. If there is no recorded encryption key for the library or the data set is encrypted with a different key, then you can specify the encryption key value by specifying the ENCRYPTKEY= option in a TABLES statement. For more information, see "TABLES Statement" on page 95.

Note: If an encryption key is recorded in the metadata with the AUTHLIB procedure, then it is honored by the SAS 9.4 release when creating and replacing SAS data sets, whether the first maintenance release for SAS 9.4 has been applied or not. The SAS 9.4 release of the AUTHLIB procedure cannot be used to administer the metadata-bound library if the REQUIRE_ENCRYPTION=YES attribute has been set.

CAUTION:

Even if you record the encryption key in metadata for the library, you should also record the key elsewhere when using ENCRYPT=AES. If you lose the metadata and forget the ENCRYPTKEY= key value, then you lose your data.

SAS cannot assist you in recovering the ENCRYPTKEY= key value. The following note is written to the log:

```
NOTE: If you lose or forget the ENCRYPTKEY= value,
there will be no way  to open the file or
recover the data.
```

For more information, see "Setting and Modifying Metadata-Bound Library Encryption Options" on page 70.

CAUTION:

If data sets using AES encryption have referential integrity constraints, then the encryption key for all data sets must be available when they are opened for Update access. Normally, SAS requires that all data sets share the same encryption key. With a recorded optional or required encryption key in metadata, related data sets can have different keys. However, issues can arise if you change the encryption key on one library that has data sets related to data sets in a different library.

TIP If a metadata-bound library contains AES-encrypted data sets, then SAS recommends that you record an encryption key and use it for all metadata-bound data sets in the library that are encrypted with AES. The best way to ensure that the encryption key is used for all data sets is to require encryption. For more information, see "Requiring Encryption for Metadata-Bound Data Sets" on page 72.

See Also

- "Example 10: Binding a Library When Existing Data Sets Are SAS Proprietary Encrypted" on page 114
- "Example 11: Binding a Library When Existing Data Sets Are AES-Encrypted" on page 116
- "Example 12: Binding a Library with an Optional Recorded Encryption Key When Existing AES-Encrypted Data Sets Have Different Encryption Keys " on page 118
- "Example 13: Binding a Library with Required AES Encryption When Existing Data Sets Are Encrypted with the Same Encryption Key" on page 121
- "Example 14: Changing the Encryption Key on a Metadata-Bound Library That Requires AES Encryption" on page 123

Setting and Modifying Metadata-Bound Library Encryption Options

There are three options that affect metadata-bound library encryption:

- REQUIRE_ENCRYPTION=
- ENCRYPT=
- ENCRYPTKEY=

The metadata-bound library encryption options are set in the CREATE statement and can be changed with the MODIFY statement. The encryption of data sets in the operating system library can be changed by the CREATE and MODIFY statements and subordinate TABLES statements. The encryption of data sets can also be changed if the library is unbound from the metadata by using a REMOVE statement. However, note that if the CREATE, MODIFY, or REMOVE statement also specifies TABLESONLY=YES, then any new encryption options on those statements are ignored. Also note that when encryption options are changed for a data set, the copy-in-place operation is automatically executed to re-encrypt the data with the new options. For more information about the copy-in-place operation, see "Copy-In-Place Operation" on page 99.

The default for the REQUIRE_ENCRYPTION= option is NO when it is used in the CREATE statement. The REQUIRE_ENCRYPTION= option can be changed in the MODIFY statement to YES or NO.

The ENCRYPT= option specifies the encryption type to use: AES, YES, or NO. ENCRYPT=NO is not valid if encryption is required. To record or change a metadata-bound library encryption key, ENCRYPT=AES must be specified. If you want to switch from required encryption with a recorded AES encryption key to required encryption with the SAS Proprietary algorithm, then specify ENCRYPT=YES in the MODIFY statement. This process also removes the recorded encryption key. To remove the recorded encryption key when encryption is not required, specify ENCRYPT=NO in the MODIFY statement. To change the encryption of data sets when unbinding with the REMOVE statement, perform one of the following tasks:

- specify different encryption options for data sets that are unbound by using TABLESONLY=YES and the encryption options on different TABLES statements
- change to a common encryption for all data sets that are unbound with the ENCRYPT=option if TABLESONLY is not YES

Similar to password options, the ENCRYPTKEY= option on statements accepts a syntax where two values that are separated by a slash (/) can be specified. Here is an example:

```
ENCRYPTKEY=key-value/new-key-value
```

For CREATE and MODIFY statements, the encryption key value to record in the metadata or data sets is obtained from the encryption key value before the slash (/) if

- ENCRYPT=AES
- there is no new key value specified after the slash (/)

If you do not specify ENCRYPT=AES, then the encryption key value is used to open data sets but is not recorded in metadata. Unlike password options, you do not remove an encryption key value by specifying a slash (/) after it and leaving it blank. Instead, you use ENCRYPT=YES or ENCRYPT=NO, as discussed in the previous paragraph.

If encryption is required, then you do not specify a new key value in a TABLES statement following a CREATE or MODIFY statement. The new value is obtained from the metadata to which the data set is bound or being bound. If encryption is not required or if you are following a REMOVE statement with TABLESONLY=YES, then you can specify ENCRYPT=AES and a new key value in TABLES statements to have the data set re-encrypted with the new key value.

See Also

- "Example 15: Binding a Library with Existing Data Sets That Are AES-Encrypted with Different Encryption Keys" on page 126
- "Example 16: Changing a Metadata-Bound Library to Require AES Encryption When Existing Data Sets Are Encrypted with Different Encryption Keys" on page 129

Retaining and Purging Metadata-Bound Library Credentials

Passwords and encryption keys for a metadata-bound library are collectively referred to as *metadata-bound library credentials*. Prior to the third maintenance release of SAS

9.4, when any of these credentials were modified, the replaced values were immediately removed from the metadata. Sometimes tables were not processed because another user was accessing the table.

Beginning with the third maintenance release of SAS 9.4, the credentials are retained in metadata and can be used by the system to open data sets that were not modified. This retention enables the user to continue processing tables and the administrator to complete the modification of credentials. The retained credentials are purged if a MODIFY statement that is processing all of the tables in the library determines that all the tables have been successfully changed with the credentials.

An administrator might want to retain the credentials even after all the existing tables have been processed successfully. The following are reasons for retaining the credentials:

- It enables processing of view files that implemented row and column level security on underlying tables by using the old passwords in the view definition. SAS does not know which view files might contain the passwords and does not have the ability to modify them in the view file. The administrator must redefine the views with the new passwords.
- It enables processing of data sets restored from backups prior to the modification.

An administrator who wants to retain older credentials and not purge them can specify the PURGE=NO option in the MODIFY statement.

Note: The administrator must specify the PURGE=NO option in each MODIFY statement that processes all tables until the administrator is ready for the replaced credentials to be purged.

If a library contains tables that do not follow our best practices, automatic deletion of old credentials might not occur when issuing a MODIFY statement for all tables. For example, a MODIFY statement that changes the stored encryption key for a library with optional encryption would not modify the keys of data sets whose keys do not match the stored key. Because some data sets were not modified, the old encryption key is not removed. In this case, the PURGE statement must be used to remove the old credentials.

Note: Notes are written to the SAS log whenever a metadata-bound table is accessed and the replaced credentials are used to successfully open the data set. The Note identifies the date and time that these credentials were replaced.

For more information, see “PURGE Statement” on page 85.

Requiring Encryption for Metadata-Bound Data Sets

Beginning in the first maintenance release for SAS 9.4, an administrator can require that all data sets in a metadata-bound library be automatically encrypted when created. This is specified by using the REQUIRE_ENCRYPTION=YES option in the CREATE or MODIFY statements. The type of encryption required depends on whether there is a recorded AES encryption key or not. If there is a recorded encryption key, then all data sets that are bound to the secured library object are automatically AES-encrypted with the recorded encryption key. If there is no recorded AES encryption key, then all data sets are automatically encrypted with the SAS Proprietary algorithm.

In order to automatically encrypt the data sets, a copy-in-place operation is used. For an explanation of the copy-in-place operation, see “Copy-In-Place Operation” on page 99. If the data set is currently encrypted with a different key value, then that key value must be either the current recorded encryption key value or specified with the ENCRYPTKEY= option in the TABLES statement.

Note: If the REQUIRE_ENCRYPTION=YES attribute of a metadata-bound library is set in the metadata with the AUTHLIB procedure, then it is honored by SAS 9.4 when creating and replacing SAS data sets whether the first maintenance release for SAS 9.4 has been applied or not. The pre-maintenance version of the AUTHLIB procedure cannot be used to administer the metadata-bound library if the REQUIRE_ENCRYPTION=YES attribute has been set. The second maintenance release of SAS 9.3 does not honor the REQUIRE_ENCRYPTION=YES attribute, and its AUTHLIB procedure should not be used to administer the library if the REQUIRE_ENCRYPTION=YES attribute is set.

See Also

- "Example 15: Binding a Library with Existing Data Sets That Are AES-Encrypted with Different Encryption Keys" on page 126
- "Example 16: Changing a Metadata-Bound Library to Require AES Encryption When Existing Data Sets Are Encrypted with Different Encryption Keys" on page 129

Data Sets in a Metadata-Bound Library That Are Not Bound to Secured Table Objects

It is possible that some data sets in a metadata-bound library do not have the metadata-bound library passwords. These data sets are not considered to be part of the bound library for authorization purposes. This can occur with either of the following scenarios:

- the data sets existed in the library before it was bound and their passwords differed from the metadata library passwords
- the data set is AES-encrypted and the encryption key was not available to open the data set in a CREATE or MODIFY statement

See the following examples:

- "Example 2: Binding a Physical Library That Contains Password-Protected Data Sets" on page 102
- "Example 12: Binding a Library with an Optional Recorded Encryption Key When Existing AES-Encrypted Data Sets Have Different Encryption Keys " on page 118

This can also occur if data sets were to be copied into the library by an operating system copy utility.

If a data set was bound before being copied, then the data set is still protected by the permissions that the users have in the secured table object to which it is bound in the original secured library.

If a data set was not bound before being copied, then it is also not bound in the new library or protected by the metadata permissions. If the data set has passwords, then you must supply the appropriate passwords to access the data.

You can use the MODIFY statement to modify the passwords if necessary and to bind the data set to a secured table object in the secured library object to which the library is bound. For more information, see "Example 5: Changing Passwords on Data Sets" on page 107.

Syntax: AUTHLIB Procedure

Restrictions:	Users cannot access metadata-bound data sets from any release of SAS prior to the second maintenance release of SAS 9.3.
	The AUTHLIB procedure is intended for use by SAS administrators. Users who lack sufficient privileges in either the metadata layer or the host layer cannot use this statement.
	The AUTHLIB procedure cannot operate on libraries that are assigned for access through a SAS/SHARE server.
	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.
Requirement:	The AUTHLIB procedure requires a connection to the target metadata server.
Tip:	Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.
See:	*SAS Guide to Metadata-Bound Libraries*

```
PROC AUTHLIB <option(s)>;
  CREATE
    SECUREDLIBRARY='secured-library-name'
     <SECUREDFOLDER='secured-folder-path'>
     <LIBRARY=libref>
     PW=all-password-value </ new-all-password-value> |
     ALTER=alter-password-value </ new-alter-password-value>
     READ=read-password-value </ new-read-password-value>
     WRITE=write-password-value </ new-write-password-value>
     <REQUIRE_ENCRYPTION=YES | NO>
     <ENCRYPT=YES | NO | AES>
     <ENCRYPTKEY=key-value </ new-key-value>>;
  MODIFY <LIBRARY=libref>
    PW=all-password </ new-all-password> |
     ALTER=alter-password </ new-alter-password>
     READ=read-password </ new-read-password>
     WRITE=write-password </ new-write-password>
     <TABLESONLY=YES | NO>
     <REQUIRE_ENCRYPTION=YES | NO>
     <ENCRYPT=YES | NO | AES>
     <ENCRYPTKEY=key-value </ new-key-value>>
     <PURGE=YES | NO>;
  PURGE CREDENTIALS | CREDS <LIBRARY=libref>
    PW=all-password |
     ALTER=alter-password
     READ=read-password
     WRITE=write-password
     BEFORE=datetime;
  REMOVE<LIBRARY=libref>
    PW=all-password </ <new-all-password>> |
     ALTER=alter-password </ <new-alter-password>>
     READ=read-password </ <new-read-password>>
     WRITE=write-password </ <new-write-password>>
     <TABLESONLY=YES | NO>
     <ENCRYPT=YES | NO | AES>
     <ENCRYPTKEY=key-value </ new-key-value>>;
  REPAIR ADD | UPDATE | DELETE
    LOCATION | METADATA
     SECUREDLIBRARY='secured-library-name'
     SECUREDFOLDER='secured-folder-path'
     <LIBRARY=libref>
     PW=all-password |
      ALTER=alter-password
        READ=read-password
        WRITE=write-password
     <TABLESONLY=YES | NO>
     <ENCRYPTKEY=key-value>;
  REPORT <LIBRARY=libref>
    <ENCRYPTKEY=key-value>;
  TABLES SAS-dataset(s) | _ALL_ | _NONE_
    </>
     <PW=all-password > </ <new-all-password>> |
     <ALTER=alter-password> </ <new-alter-password>>
     <READ=read-password> </ <new-read-password>>
     <WRITE=write-password> </ <new-write-password>>;
     <MEMTYPE= DATA | VIEW>
     <ENCRYPT=YES | NO | AES>
```

Statement	Task	Example
PROC AUTHLIB	Create and manage metadata-bound libraries	
CREATE	Create the secured library object in the SAS Metadata Server and record the physical security information in the directory or bound files	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 11, Ex. 10, Ex. 12, Ex. 13, Ex. 15
MODIFY	Modify password values and encryption key values for a metadata-bound library	Ex. 5, Ex. 6, Ex. 16
PURGE	Removes any retained metadata-bound library credentials older than a given date of replacement.	
REMOVE	Remove the physical security information and metadata objects that protect a metadata-bound library	Ex. 7
REPAIR	Recover security information (in physical data) or secured library and table objects (in metadata)	
REPORT	For a specified metadata-bound library, compare physical library contents with corresponding metadata objects (in order to identify any inconsistencies)	Ex. 8
TABLES	Specify which tables within a specified metadata-bound library are affected by certain AUTHLIB statements	Ex. 4, Ex. 9, Ex. 11, Ex. 10, Ex. 12, Ex. 15, Ex. 16

PROC AUTHLIB Statement

Manages metadata-bound libraries.

Syntax

PROC AUTHLIB *<option(s)>*;

Summary of Optional Arguments

LIBRARY=*libref*
: is the name of the physical library for which the secured library object is created and the security information is stored.

NOWARN
: suppresses error processing.

PWREQ=YES | NO
: controls the pop up of a dialog box.

Optional Arguments

LIBRARY=*libref*
is the name of the physical library for which the secured library object is created and the security information is stored.

If the LIBRARY= option is not specified, then the LIBRARY=*libref* (physical library) from the CREATE, MODIFY, REMOVE, REPORT, or REPAIR statement is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

NOWARN
suppresses the `file not found` error message when a data set in a TABLES statement does not exist.

PWREQ=YES | NO
controls the pop up of a dialog box for a data set password in interactive mode.

YES
specifies that a dialog box appear if a missing or invalid password is entered when required.

NO
prevents a dialog box from appearing. If a missing or invalid password is entered, then the data set is not opened, and an error message is written to the SAS log.

Default	PWREQ=NO

CREATE Statement

Binds a physical library and data sets in the library to metadata by generating corresponding metadata objects in the SAS Metadata Repository and creating a record of the metadata objects in the physical directory and data sets.

Requirement:	The AUTHLIB CREATE statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the AUTHLIB Statements" on page 99.
Tip:	Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.

Syntax

CREATE

> SECUREDLIBRARY='*secured-library-name*'
> <SECUREDFOLDER='*secured-folder-path*'>
> <LIBRARY=*libref*>
> PW=*all-password-value* </ *new-all-password-value*> |
> ALTER=*alter-password-value* </ *new-alter-password-value*>
> READ=*read-password-value* </ *new-read-password-value*>
> WRITE=*write-password-value* </ *new-write-password-value*>
> <REQUIRE_ENCRYPTION=YES | NO>
> <ENCRYPT=YES | NO | AES>
> <ENCRYPTKEY=*key-value* </ *new-key-value*>>;

Required Arguments

SECUREDLIBRARY='*secured-library-name*'
names the secured library object in the SAS Metadata Server.

Alias	SECLIB=
Restriction	The total length of the secured library object pathname including the fully qualified secured folder path cannot exceed 256 characters.

PW=*all-password-value* </ *new-all-password-value*>
sets a single password for a metadata-bound library.

ALTER=*alter-password-value* </ *new-alter-password-value*>
sets one of a maximum of three password values for a metadata-bound library.

READ=*read-password-value* </ *new-read-password-value*>
sets one of a maximum of three password values for a metadata-bound library.

WRITE=*write-password-value* </ *new-write-password-value*>
sets one of a maximum of three password values for a metadata-bound library.

TIP All password values must be valid SAS names with a maximum length of 8 characters.

Optional Arguments

SECUREDFOLDER='*secured-folder-path*'
is the name of the metadata folder within the `/System/Secured Libraries` folder tree where the secured library object is created.

If the SECUREDFOLDER= option is not specified, then the metadata-bound library is created directly in the `/System/Secured Libraries` folder of the Foundation repository. If the SECUREDFOLDER= option does not begin with a slash (/), then it is a relative path and the value is appended to `/System/Secured Libraries/` to find the folder. If the SECUREDFOLDER= option begins with a slash (/), then it is an absolute path and the value must begin with `/System/Secured Libraries` or `/<repository_name>/System/Secured Libraries`.

Alias	SECFLDR=
Restriction	The total length of the secured library object pathname including the fully qualified secured folder path cannot exceed 256 characters.

ENCRYPT=YES | NO | AES
specifies the encryption type.

YES
specifies the SAS Proprietary algorithm.

NO
specifies no encryption.

AES
specifies Advanced Encryption Standard (AES) encryption and to record the key in metadata.

Requirement	ENCRYPTKEY= option is required if the library has AES encryption.

See	"Encrypted Data Set Considerations" on page 69

ENCRYPTKEY=*key-value* <*/ key-value*>
specifies a key value for AES encryption.

Requirement	ENCRYPTKEY= option is required if the library or a data file has AES encryption.
Note	The encryption key value for all the data sets in a library can be stored in a metadata-bound library so that an authorized user does not have to supply the encryption key value every time a data set is opened. See "Considerations for Data File Encryption" on page 47.
Tip	The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.
See	"Encrypted Data Set Considerations" on page 69
	"ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

LIBRARY=*libref*
name of the physical library for which the secured library object is created and the security information is stored.

If the LIBRARY= option is not specified, then the physical library from the AUTHLIB procedure is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

REQUIRE_ENCRYPTION=YES | NO

YES
specifies that all data sets in a metadata-bound library are automatically encrypted.

NO
specifies that data sets in a metadata-bound library are not automatically encrypted.

See "Requiring Encryption for Metadata-Bound Data Sets" on page 72

Details

Specifying Passwords

If your physical library does not contain password-protected data sets, then you need to specify the new metadata-bound library password(s) with either the PW= option or READ=,WRITE=, and ALTER= options in the CREATE statement. This is the most common case. For an example, see "Example 1: Binding a Physical Library That Contains Unprotected Data Sets" on page 100.

If your physical library contains some password-protected data sets that all share the same current set of passwords, then you can specify the most restrictive password on the data sets before a slash (/) in the CREATE statement password option(s) and the new password(s) after the slash (/). For an example, see "Example 3: Binding a Library When Existing Data Sets Are Protected with the Same Passwords" on page 104.

If your physical library contains password-protected data sets with different sets of passwords, then you can specify the data sets with each set of passwords on separate TABLES statements (see "Example 4: Binding a Library When Existing Data Sets Are Protected with Different Passwords" on page 105) or you can subsequently use MODIFY and TABLES statements to change the passwords after the library has been bound with the CREATE statement (see "Example 5: Changing Passwords on Data Sets" on page 107).

Specifying Encryption Keys

To create or access a metadata-bound library that is protected using AES algorithm requires an encryption key value. You must use ENCRYPT=AES and ENCRYPTKEY=*key-value* data set options.

If your physical library contains some AES-encrypted data sets that all share the same AES encryption key, then you can specify the key value following ENCRYPTKEY= in the CREATE statement. If you want to record the key in metadata, then specify ENCRYPT=AES. For an example, see "Example 13: Binding a Library with Required AES Encryption When Existing Data Sets Are Encrypted with the Same Encryption Key" on page 121.

If your physical library contains AES-encrypted data sets with different encryption keys, then you can specify the data sets with each encryption key on separate TABLES statements. For an example, see "Example 15: Binding a Library with Existing Data Sets That Are AES-Encrypted with Different Encryption Keys" on page 126.

TIP See "Considerations for Data File Encryption" on page 47.

For more information, see "ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference* and "ENCRYPT= Data Set Option" in *SAS Data Set Options: Reference*.

CAUTION:
If data sets using AES encryption have referential integrity constraints, then the encryption key for all data sets must be available when they are opened for Update access. Normally, SAS requires that all data sets share the same encryption key. With a recorded optional or required encryption key in metadata, related data

sets can have different keys. However, issues can arise if you change the encryption key on one library that has data sets related to data sets in a different library.

CAUTION:

For AES-encrypted data sets that are referentially related to one another, follow these best practices to ensure that the data does not become inaccessible: Store the encryption key in the library's metadata. You can modify the stored key, but do not remove the key from metadata and do not unbind the library.

CAUTION:

Even if you record the encryption key in metadata for the library, then you should also record the key elsewhere when using ENCRYPT=AES. If you lose the metadata and forget the ENCRYPTKEY= key value, then you lose your data. SAS cannot assist you in recovering the ENCRYPTKEY= key value. The following note is written to the log:

```
NOTE: If you lose or forget the ENCRYPTKEY= value,
there will be no way  to open the file or
recover the data.
```

MODIFY Statement

Modifies password and encryption key values for a metadata-bound library.

Requirement: The AUTHLIB MODIFY statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the AUTHLIB Statements" on page 99.

Tip: Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.

Syntax

MODIFY

<LIBRARY=*libref*>
PW=*all-password* </ *new-all-password*> |
ALTER=*alter-password* </ *new-alter-password*>
READ=*read-password* </ *new-read-password*>
WRITE=*write-password* </ *new-write-password*>
<TABLESONLY=YES | NO>
<REQUIRE_ENCRYPTION=YES | NO>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value* </ *new-key-value*>>
<PURGE=YES | NO>;

Required Arguments

PW=*all-password* </ *new-all-password*>
modifies a single password for a metadata-bound library.

ALTER=*alter-password* </ *new-alter-password*>
modifies one of a maximum of three password values for a metadata-bound library.

READ=*read-password* </ *new-read-password*>
modifies one of a maximum of three password values for a metadata-bound library.

WRITE=*write-password* </ *new-write-password*>
modifies one of a maximum of three password values for a metadata-bound library.

TIP All password values must be valid SAS names with a maximum length of 8 characters.

Optional Arguments

ENCRYPT=YES | NO | AES
specifies the encryption type.

YES
specifies the SAS Proprietary algorithm.

NO
specifies no encryption.

AES
specifies Advanced Encryption Standard (AES) encryption and to record the key in metadata.

Requirement	ENCRYPTKEY= option is required if the library has AES encryption.

See	"Encrypted Data Set Considerations" on page 69

ENCRYPTKEY=*key-value* </ *key-value*>
specifies a key value for AES encryption.

Requirement	ENCRYPTKEY= option is required if the library or a data file has AES encryption.
Note	The encryption key value for all the data sets in a library can be stored in a metadata-bound library so that an authorized user does not have to supply the encryption key value every time a data set is opened. See "Considerations for Data File Encryption" on page 47.
Tip	The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.
See	"Encrypted Data Set Considerations" on page 69
	"ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

LIBRARY=*libref*
name of the physical library that is metadata-bound.

If the LIBRARY= option is not specified, then the physical library from the AUTHLIB procedure is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

PURGE=YES | NO

YES
removes all retained metadata-bound library credentials if all tables in the library are successfully modified to the newer credentials.

Default	YES

NO
does not remove replaced metadata-bound library credentials even if all tables in the library were successfully modified.

See	"Retaining and Purging Metadata-Bound Library Credentials" on page 71

REQUIRE_ENCRYPTION=YES | NO

YES
specifies that all data sets in a metadata-bound library are automatically encrypted.

NO
specifies that data sets in a metadata-bound library are not automatically encrypted.

See	"Requiring Encryption for Metadata-Bound Data Sets" on page 72

TABLESONLY=YES | NO
specifies whether the MODIFY statement action is applied at the library level or just to the tables. If TABLESONLY=NO, then the action is applied to the library and data sets. If TABLESONLY=YES, then the action is applied only to the data sets.

Default	NO
Tip	If you specify TABLESONLY=YES and a new password or encryption key value in the CREATE, MODIFY, or REMOVE statement, then the new password value or encryption key value is ignored. The current password or encryption key value is still required if the library is metadata-bound.

Details

Using the MODIFY Statement

The MODIFY statement can modify the value of the required metadata-bound library passwords and encryption options. This statement can also modify passwords on data sets (tables) that do not have the required metadata-bound library password values. The TABLES statement follows the MODIFY statement to specify current passwords and encryption keys in the data sets.

If your physical library is currently bound to a metadata library with one set of passwords and you want to change the metadata-bound library passwords to another set, then specify the current and new values for the metadata-bound library passwords separated by a / in the MODIFY statement. For an example, see "Example 6: Changing Metadata-Bound Library Passwords" on page 109.

If your physical library contains password-protected data sets with different sets of passwords from the metadata-bound library passwords, then you can modify the data set passwords to match the metadata-bound library required passwords using the MODIFY and TABLES statements. Specify the metadata-bound library passwords in the MODIFY statement. Specify the data sets with each set of passwords in separate TABLES statements. For more information, see "Example 5: Changing Passwords on Data Sets" on page 107.

If you want to change encryption options for the library, then specify the new options in the MODIFY statement. If your physical library contains AES-encrypted data sets, then you must specify the ENCRYPTKEY= key value in the MODIFY or TABLES statements or have a recorded encryption key for the library to make any modifications to the encrypted data sets. For and example, see "Example 16: Changing a Metadata-Bound Library to Require AES Encryption When Existing Data Sets Are Encrypted with Different Encryption Keys" on page 129.

For more information, see "TABLES Statement" on page 95.

CAUTION:

For AES-encrypted data sets that are referentially related to one another, follow these best practices to ensure that the data does not become inaccessible: Store the encryption key in the library's metadata. You can modify the stored key, but do not remove the key from metadata and do not unbind the library.

CAUTION:

Even if you record the encryption key in metadata for the library, you should also record the key elsewhere when using ENCRYPT=AES. If you lose the metadata and forget the ENCRYPTKEY= key value, then you lose your data. SAS cannot assist you in recovering the ENCRYPTKEY= key value.

You might have a need to import a SecuredLibrary object from a backup package for one of the following reasons:

- the SecuredLibrary object was inadvertently deleted
- you are promoting the metadata-bound library to a new metadata server

Password values and encryption key values are not exported with the SecuredLibrary object. This prevents them from being imported to a rogue Metadata Server. In this case, the passwords and any recorded encryption key values need to be reset in the imported SecuredLibrary object. Until you do this, libname assignments that refers to the imported SecuredLibrary object will fail with the following messages:

```
ERROR: The secured library object information for library library-name
could not be obtained from the metadata server or has invalid data.
ERROR: Association not found.
ERROR: Error in the LIBNAME statement.
```

For an example, see "Example 18: Resetting Credentials on Imported SecuredLibrary Objects" on page 133.

Using the LIBRARY= Option

If you want to override the default library from the AUTHLIB procedure, then use LIBRARY=.

```
MODIFY <LIBRARY=library-name>
```

If you want to modify the passwords or encryption options for a secured library object that is no longer bound to a physical library, then specify LIBRARY=_NONE_ with the SECUREDLIBRARY= and SECUREDFOLDER= options to locate the secured library object.

```
MODIFY <LIBRARY=_NONE_ SECUREDLIBRARY=secured-library-name>
                <SECUREDFOLDER=secured-folder-name>
```

CAUTION:

Do not use LIB=_none_ when the secured library object is bound to a physical library. LIB=_none_ causes the action to operate only on the secured library object and has no effect on the physical data.

Using the PURGE Option

Passwords and encryption keys for a metadata-bound library are collectively referred to as *metadata-bound library credentials*. For information about retaining and purging credentials, see "Retaining and Purging Metadata-Bound Library Credentials" on page 71.

PURGE Statement

Removes any retained metadata-bound library credentials older than a given date of replacement.

Requirement: The AUTHLIB PURGE statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the AUTHLIB Statements" on page 99.

Tip: Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.

Syntax

PURGE CREDENTIALS | CREDS <LIBRARY=*libref*>

PW=*all-password* |
  ALTER=*alter-password*
  READ=*read-password*
  WRITE=*write-password*
  BEFORE=*datetime*;

Required Arguments

PW=*all-password*
specifies a single password for a metadata-bound library.

ALTER=*alter-password*
specifies one of a maximum of three password values for a metadata-bound library.

READ=*read-password*
specifies one of a maximum of three password values for a metadata-bound library.

WRITE=*write-password*
specifies one of a maximum of three password values for a metadata-bound library.

TIP All password values must be valid SAS names with a maximum length of 8 characters.

BEFORE=*datetime*
specifies a datetime constant before any replaced, but retained, credentials are removed.

Optional Argument

LIBRARY=*libref*

name of the physical library for which the metadata-bound library is created and the security information is stored.

If the LIBRARY= option is not specified, then the physical library from the AUTHLIB procedure is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

Details

Using the PURGE Statement

Passwords and encryption keys for a metadata-bound library are collectively referred to as *metadata-bound library credentials*. For more information about purging metadata-bound library credentials, see "Retaining and Purging Metadata-Bound Library Credentials" on page 71.

Using the LIBRARY= Option

If you want to override the default library from the AUTHLIB procedure, then use LIBRARY= option.

```
PURGE CREDENTIALS <LIBRARY=library-name>
```

If you want to purge the credentials for a secured library object that is no longer bound to a physical library, then specify LIBRARY=_NONE_ with the SECUREDLIBRARY= and SECUREDFOLDER= options to locate the secured library object.

```
PURGE CREDENTIALS <LIBRARY=_NONE_ SECUREDLIBRARY=secured-library-name>
<SECUREDFOLDER=secured-folder-name>
```

REMOVE Statement

Removes the physical security information and metadata objects that protect a metadata-bound library so that it is no longer a metadata-bound library.

Requirement: The AUTHLIB REMOVE statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the AUTHLIB Statements" on page 99.

Note: If any data set uses SAS Proprietary Encryption, then you cannot remove passwords unless you also specify ENCRYPT=NO to remove encryption.

Tips: Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.

If you do not want the non-secured data sets altered, then move all non-secured data sets from the physical library before performing a REMOVE statement.

Before you use the REMOVE statement, consider running the REPORT statement. The output from the REPORT statement identifies any physical tables that do not have corresponding secured table objects in metadata. In the unusual circumstance that such physical tables exist, their security location information is unaffected by the

REMOVE statement unless you specify AUTHADMIN=YES in the LIBNAME statement. You should use the AUTHADMIN=YES option in the LIBNAME statement in this circumstance.

Examples: "Example 7: Using the REMOVE Statement" on page 110

"Example 17: Using the REMOVE Statement on a Metadata-Bound Library with Required AES Encryption" on page 131

Syntax

REMOVE<LIBRARY=*libref*>

PW=*all-password* </ <*new-all-password*>> |
  ALTER=*alter-password* </ <*new-alter-password*>>
  READ=*read-password* </ <*new-read-password*>>
  WRITE=*write-password* </ <*new-write-password*>>
  <TABLESONLY=YES | NO>
  <ENCRYPT=YES | NO | AES>
  <ENCRYPTKEY=*key-value* </ *new-key-value*>>;

Required Arguments

PW=*all-password* </ <*new-all-password*>>
specifies a single password for a metadata-bound library.

ALTER=*alter-password* </ <*new-alter-password*>>
specifies one of a maximum of three password values for a metadata-bound library.

READ=*read-password* </ <*new-read-password*>>
specifies one of a maximum of three password values for a metadata-bound library.

WRITE=*write-password* </ <*new-write-password*>>
specifies one of a maximum of three password values for a metadata-bound library.

Optional Arguments

ENCRYPT=YES | NO | AES
specifies the encryption type.

YES
specifies the SAS Proprietary algorithm.

NO
specifies no encryption.

AES
specifies Advanced Encryption Standard (AES) encryption and is required if specifying that data sets be encrypted with a new key value.

See "Encrypted Data Set Considerations" on page 69

ENCRYPTKEY=*key-value* </ *key-value*>
specifies a key value for AES encryption.

Tip The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.

See "ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

LIBRARY=*libref*

name of the physical library that is metadata-bound.

If the LIBRARY= option is not specified, then the physical library from the PROC AUTHLIB statement is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

TABLESONLY=YES | NO

specifies whether the REMOVE statement action is applied at the library level or just to the tables. If TABLESONLY=NO, then the action is applied to the library and data sets. If TABLESONLY=YES, then the action is applied only to the individual data sets listed.

Default	NO
Tip	If you specify TABLESONLY=YES and a new password or encryption options, then the new password or encryption options are ignored. The current password is still required if the library is metadata-bound.

Details

The REMOVE statement is used to unbind the metadata-bound library feature from a SAS library and the data sets within it. This statement also removes the secured library and secured table objects from the SAS Metadata Server. The data sets remain in the physical library protected by the metadata-bound library passwords unless the administrator specifies password modifications in the REMOVE statement. Since the metadata-bound library feature is being removed and there is no longer a requirement that the data set passwords match the metadata-bound library passwords, the data set passwords can be removed by using a slash (/) after the current password but not specifying a new password. If you choose to do this, then you are warned in the SAS log that the data sets no longer have any SAS protection. You can also modify the encryption key of data sets by specifying the new key following a slash (/) in ENCRYPTKEY= and specifying ENCRYPT=AES. You can change to SAS Proprietary Encryption by specifying ENCRYPT=YES. You can remove all encryption by specifying ENCRYPT=NO.

The REMOVE statement removes the location information from any data set if the passwords specified match the metadata-bound library passwords stored in the data set. Note also that if the data set is AES-encrypted, the encryption key must either be recorded in metadata or specified in the REMOVE or TABLES statements. However, it does not delete the referenced secured table object unless that secured table object is under the secured library object to which the operating system library is bound. If a data set has been copied into the bound library by a utility not written in SAS from another metadata-bound library, then this process prevents a REMOVE from deleting the secured table object that belongs to the other metadata-bound library.

Note: Ensure that all physical tables that are protected by a particular metadata-bound library remain within that library (directory). This best practice maximizes clarity and is essential in order for REMOVE statements to be fully effective. Special circumstances (for example, a table that is host copied to another directory) can prevent a REMOVE statement from unbinding the relocated data set.

CAUTION:

If you have to unbind a library that contains AES-encrypted data sets that are referentially related to other data sets, then either make sure that all related data sets are no longer AES-encrypted or make sure that all related data sets share the same encryption key. If you preserve AES encryption, the data will be available only to those users who supply the key and have host-layer access.

REPAIR Statement

Recovers security information (in physical data) or secured library and table objects (in metadata).

Requirement: The AUTHLIB REPAIR statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the AUTHLIB Statements" on page 99.

Tip: Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.

Syntax

REPAIR ADD | UPDATE | DELETE
  LOCATION | METADATA
    SECUREDLIBRARY='*secured-library-name*'
    SECUREDFOLDER='*secured-folder-path*'
    <LIBRARY=*libref*>
    PW=*all-password* |
    ALTER=*alter-password*
    READ=*read-password*
    WRITE=*write-password*
    <TABLESONLY=YES | NO>
    <ENCRYPTKEY=*key-value*>;

Required Arguments

ADD | UPDATE | DELETE
one of these actions must be specified.

LOCATION | METADATA
clarifies whether the action is to apply to the physical security information in the file system, to the metadata objects in the SAS Metadata Server, or to both.

PW=*all-password*
specifies a single password for a metadata-bound library.

ALTER=*alter-password*
assigns, changes, or removes an Alter password from the secured library object and from the data sets in the physical library.

READ=*read-password*
assigns, changes, or removes a Read password from the secured library object and from the data sets in the physical library.

WRITE=*write-password*
assigns, changes, or removes a Write password from the secured library object and from the data sets in the physical library.

Optional Arguments

ENCRYPTKEY=*key-value*
specifies a key value for AES encryption.

Requirement	ENCRYPTKEY= data set option is required if the library or a data file has AES encryption and if the key is not recorded in the library metadata.
Note	The encryption key value for all the data sets in a library can be stored in a metadata-bound library so that an authorized user does not have to supply the encryption key value every time a data set is opened. See "Considerations for Data File Encryption" on page 47.
Tip	The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.
See	"ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

LIBRARY=*libref*
name of the physical library where the security information is stored.

If the LIBRARY= option is not specified, then the physical library from the PROC AUTHLIB statement is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

SECUREDLIBRARY='*secured-library-name*'
names the secured library object in the SAS Metadata Server.

Alias	SECLIB=
Restriction	The total length of the secured library object pathname including the fully qualified secured folder path cannot exceed 256 characters.

SECUREDFOLDER='*secured-folder-path*'
name of the metadata folder within a `/System/Secured Libraries` folder tree where the secured library is repaired or re-created.

Alias	SECFLDR=
Restriction	The total length of the secured library object pathname including the fully qualified secured folder path cannot exceed 256 characters.

TABLESONLY=YES | NO
specifies whether the REPAIR statement action is applied at the library level or just to the tables. If TABLESONLY=NO, then the action is applied to the library and the tables. If TABLESONLY=YES, then the action is applied only to the tables. This is especially important for REPAIR because it gives the administrator a way to delete specific secured table objects without deleting the secured library and all secured tables.

Default NO

Details

The REPAIR statement feature that has been fully tested is REPAIR DELETE LOCATION. Use this combination of options when you need to delete the security information in a metadata-bound library and or data sets within the library without deleting the metadata objects.

It is possible for a system administrator to get in situations where a data set still has location information pointing to a secured table object that no longer exists. REPAIR DELETE LOCATION is required to remove that location information before the data set can be accessed in any other way.

When using the REPAIR statement, one of the ADD, UPDATE, or DELETE actions must be specified. LOCATION, METADATA, or both are used to clarify if the action is to apply to the metadata security information in the file system, to the metadata objects in the SAS Metadata Server, or to both. Other than DELETE LOCATION, these other actions have not been fully tested and are considered pre-production implementations. They are documented here but should be used only under advise and direction from Technical Support.

One or more TABLES statements can follow the REPAIR statement to perform the same action on the specified data sets. An implicit TABLES _ALL_ is used if no TABLES statement follows the REPAIR statement.

Inconsistencies between the metadata security information stored in the operating system files and the secured library object in the SAS Metadata Server that need repair can prevent the assignment of a LIBNAME statement to the physical library. The administrator that owns the physical library and knows the metadata-bound library passwords can perform a library assignment and repair the data by adding the AUTHADMIN=YES option to the LIBNAME statement. Best practice is to use the AUTHADMIN=YES option when performing any REPAIR actions.

CAUTION:

Repairing a metadata-bound library is an advanced task. Make sure you have a current backup (of both metadata and physical data) before you use this statement.

Use the REPAIR statement to restore metadata-bound library security information or metadata objects that are inadvertently deleted. The administrator can carefully use the REPAIR statement to make some repairs to inconsistencies reported by the REPORT statement. If there are a significant number of groupings in the REPORT listing, then it might be more advisable to do the following:

1. Create a new operating system directory and metadata-bound library, and then use SAS Management Console to set appropriate default library permissions for the new secured library object.
2. Access the current library with the AUTHADMIN=YES, AUTHPW= or AUTHALTER=, AUTHWRITE=, and AUTHREAD= options in the LIBNAME statement.
3. Use the SAS COPY procedure to copy the SAS data sets to the new library. Use CONSTRAINT=YES if any data sets have referential integrity constraints. Use SAS Management Console to set any permissions on the secured table objects that differ from those inherited from the secured library object. The following is an example of using the COPY procedure.

Metadata-bound library ABCDE also has data sets Employees, EmpInfo, and Product. The REPORT statement has shown some inconsistencies between the physical library contents and the corresponding metadata objects. This is an example of a way to resolve these differences.

```
libname klmno "SAS-library-2";

proc authlib lib=klmno;
 create securedfolder="Department XYZZY"
        securedlibrary="KLMNOEmps"
        pw=password;
run;
quit;

libname abcde "SAS-library"
   AUTHADMIN=yes
   AUTHPW=password;

proc copy in=abcde out=klmno ;run;
```

Log A2.1 *Using PROC COPY to Resolve Differences*

```
88   proc copy in=abcde out=klmno ;run;

NOTE: Copying ABCDE.EMPINFO to KLMNO.EMPINFO (memtype=DATA).
NOTE: Data set ABCDE.EMPINFO.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPINFO.DATA.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object
      at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.EMPINFO.DATA.
NOTE: There were 5 observations read from the data set ABCDE.EMPINFO.
NOTE: The data set KLMNO.EMPINFO has 5 observations and 6 variables.
NOTE: Copying ABCDE.EMPLOYEES to KLMNO.EMPLOYEES (memtype=DATA).
NOTE: Data set ABCDE.EMPLOYEES.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPLOYEES.DATA.
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library
      object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.EMPLOYEES.DATA.
```

```
NOTE: There were 5 observations read from the data set ABCDE.EMPLOYEES.
NOTE: The data set KLMNO.EMPLOYEES has 5 observations and 6 variables.
NOTE: Copying ABCDE.PRODUCT to KLMNO.PRODUCT (memtype=DATA).
NOTE: Data set ABCDE.PRODUCT.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.PRODUCT.DATA.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object
      at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.PRODUCT.DATA.
NOTE: There were 5 observations read from the data set ABCDE.PRODUCT.
NOTE: The data set KLMNO.PRODUCT has 5 observations and 2 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time           0.14 seconds
      cpu time            0.04 seconds
```

The following REPAIR statement combination of options are *preproduction* and have not been fully tested. Preproduction means that this feature is a preliminary release of software that has not completed full development and testing. Because it has not been fully tested, preproduction software should be used with care. After final testing is completed, preproduction software is likely to be offered in a future release as a production-quality component or product.

REPAIR ADD LOCATION
: Use this combination of options when metadata-bound library and secured table security information is missing in the metadata-bound library or data sets within the metadata-bound library. The secured library and secured tables objects must exist in the SAS Metadata Server.

REPAIR UPDATE LOCATION
: Use this combination of options when metadata-bound library and secured table security information exists in the metadata-bound library or data sets within the metadata-bound library but points to incorrect or non-existent metadata objects. The secured library and secured tables objects to which you update the location information must exist in the SAS Metadata Server.

REPAIR ADD METADATA LOCATION
: Use this combination of options when secured library and secured table objects have been deleted from the SAS Metadata Server and their security information is no longer registered in the metadata-bound library and data sets within the metadata-bound library. The metadata objects are created in the SAS Metadata Server, and the security information for these objects are registered in the metadata-bound library and data sets.

REPAIR DELETE METADATA
: Use this combination of options when you need to delete the secured library, the secured table metadata objects, or both without deleting the security information in a metadata-bound library or in the data sets within that library.

REPAIR DELETE METADATA LOCATION
: Use this combination of options when you need to delete the secured library, the secured table metadata objects, or both and the security information in a metadata-bound library or in the data sets within that library.

REPAIR UPDATE LOCATION
: Use this combination of options when you need to update the security information in a metadata-bound library, in the data sets, or both to point to different existing secured library and secured table metadata objects.

  Note: The METADATA option is not supported with a REPAIR UPDATE action.

REPORT Statement

For a specified metadata-bound library, compares physical library contents with corresponding metadata objects (in order to identify any inconsistencies).

Requirement:	The AUTHLIB REPORT statement requires a connection to the target metadata server. For more requirements, see "Requirements for Using the REPORT Statement" on page 94.
Tip:	Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.
Example:	"Example 8: Using the REPORT Statement" on page 112

Syntax

REPORT
 <LIBRARY=*libref*>
  <ENCRYPTKEY=*key-value*>;

Optional Arguments

LIBRARY=*libref*
name of the physical library on which to report binding information.

If the LIBRARY= option is not specified, then the physical library from the PROC AUTHLIB statement is used.

Alias	LIB=, DDNAME=, DD=
Restriction	The physical library specified cannot be a concatenated library, temporary library, or accessed through a SAS/SHARE server and must be processed by an engine that supports metadata-bound libraries.

ENCRYPTKEY=*key-value*
specifies a key value for an AES encryption.

Tip	The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.
See	"ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

Details

Requirements for Using the REPORT Statement

An administrator uses the REPORT statement to identify any inconsistencies between a physical metadata-bound library and its corresponding metadata objects.

In order to use the REPORT statement, you must meet the following criteria:

- The SAS session runs under an account that has host-layer Read access to the target physical library. This is necessary in order to assign the libref.
- The SAS session connects to the metadata server as an identity that has the ReadMetadata permission for the target secured library object and secured table objects.
- If the library has secured library object location information and the secured library object cannot be obtained, then you will need to use the AUTHADMIN=YES option in the LIBNAME= statement in order to assign the library.

Reporting Inconsistencies

The REPORT statement is used to report any inconsistencies between the physical library contents and the corresponding metadata objects.

Inconsistencies between the metadata security information in the physical directory, data sets, the secured library, and secured table objects might occur if the metadata or the operating system files are manipulated using nonstandard SAS processing. For example, an operating system data set copied from one directory into a metadata-bound library directory using an operating system copy utility will not have the appropriate security information for that metadata-bound library. Another example is that an administrator might mistakenly delete a secured library or secured table object using SAS Management Console.

The REPORT statement reports the secured table and metadata-bound library security information for each data set in the operating system directory of the library. This data set information is grouped by the metadata-bound library attributes that all the data sets share. If any data sets in the physical library are correctly registered to the secured library object for the library and have the required passwords, then those data sets and attributes will be listed as the first grouping in the report. Subsequent groupings are for data sets with either passwords that differ from the metadata-bound library passwords or whose metadata-bound library security information does not match the metadata-bound library location registered for the operating system directory.

TABLES Statement

Used after a CREATE, MODIFY, REMOVE, REPAIR, and REPORT statement to specify the tables to process a statement action. Also, you can specify the current passwords or encryption key value of the data sets in the TABLES statement, if different from the metadata-bound library passwords or recorded encryption key.

Default:	When no TABLES statement is specified, the TABLES _ALL_ statement is the default behavior.
Requirement:	The TABLES statement must be preceded by a CREATE, MODIFY, REMOVE, REPAIR, REPORT, or another TABLES statement.
Tip:	Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log.
Example:	"Example 9: Using the TABLES Statement" on page 113

Syntax

TABLES *SAS-dataset(s)* | _ALL_ | _NONE_

</>
<PW=*all-password* > </ <*new-all-password*>> |
<ALTER=*alter-password*> </ <*new-alter-password*>>
<READ=*read-password*> </ <*new-read-password*>>
<WRITE=*write-password*> </ <*new-write-password*>>;
<MEMTYPE= DATA | VIEW>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value*< / *new-key-value*>>;

Required Argument

***SAS-dataset(s)* | _ALL_ | _NONE_**

SAS-dataset(s)
name of one or more SAS data sets

ALL
specifies password options to apply to all data sets.

NONE
limits the action of the previous CREATE, MODIFY, or REPAIR statements to the library level and does not apply the action to any table.

Optional Arguments

/
is required if any options are included, such as passwords or MEMTYPE=. Here is an example:

```
tables table-name / pw=password;
```

ENCRYPT=YES | NO | AES
specifies the encryption type.

YES
specifies the SAS Proprietary algorithm.

NO
specifies no encryption.

AES
specifies Advanced Encryption Standard (AES) encryption and is required if changing and encrypting with a new key value and TABLESONLY=YES in the action statement.

See “Encrypted Data Set Considerations” on page 69

ENCRYPTKEY=*key-value* </ *key-value*>
specifies a key value for AES encryption.

Requirement	ENCRYPTKEY= data set option is required if the data file has AES encryption and the key is not recorded for the library.
Tip	The ENCRYPTKEY= value is a passphrase that can be up to 64 characters long from which the actual AES encryption key is later derived, but it is referred to as the encryption key in most SAS documentation.

See	"Encrypted Data Set Considerations" on page 69
	"ENCRYPTKEY= Data Set Option" in *SAS Data Set Options: Reference*

MEMTYPE= DATA | VIEW
restricts processing to a single member type of DATA or VIEW. If not specified, then the default is both types.

DATA
specifies SAS data file member type.

VIEW
specifies SAS view member type.

Alias	MTYPE=, MT=
Default	ALL

PW=*all-password* </ <*new-all-password*>>
specifies the current password of the data set.

ALTER=*alter-password* </ <*new-alter-password*>>
specifies the current ALTER= password of the data set.

READ=*read-password* </ <*new-read-password*>>
specifies the current READ= password of the data set.

WRITE=*write-password* </ <*new-write-password*>>
specifies the current WRITE= password of the data set.

TIP All password values must be valid SAS names with a maximum length of 8 characters.

Details

Using the TABLES Statement

The TABLES statement is primarily used to specify the current password(s) and encryption key(s) on data sets when different from the current metadata-bound library required password(s) or encryption key(s). A TABLES statement usually follows a CREATE or MODIFY statement to make the data set passwords and encryption keys change to the metadata-bound library passwords and encryption keys. For an example, see "Example 4: Binding a Library When Existing Data Sets Are Protected with Different Passwords" on page 105.

TABLES _NONE_ can be used to limit the action of the previous CREATE, MODIFY, or REPAIR statements to the library level and not apply the action to any table. TABLES _ALL_ is the default behavior if no TABLES statement is specified. You might wish to write an explicit TABLES _ALL_ if you want to specify passwords or encryption key values to use when opening all data sets.

Using the TABLES Statement with the CREATE Statement

The CREATE statement can be followed by one or more TABLES statements to specify current passwords or encryption key values for data sets when different from the metadata-bound library passwords and encryption keys. If the TABLES statement is not used, then only two groups of data sets are bound:

- data sets without passwords or encryption keys

- data sets with passwords or encryption key values matching the metadata-bound library

In effect, omitting TABLES statements is equivalent to specifying one TABLES _ALL_ statement. For more information, see "CREATE Statement" on page 77.

Using the TABLES Statement with the MODIFY Statement

The MODIFY statement can be followed by one or more TABLES statements to specify modifications to passwords or an encryption key value in the data sets. If no TABLES statement follows the MODIFY statement, then there is an implicit TABLES _ALL_ statement. A separate TABLES statement is required for sets of data sets (tables) that might have different current passwords or encryption keys. For more information, see "MODIFY Statement" on page 81.

Using the TABLES Statement with the REPAIR Statement

When using the REPAIR statement, one of the ADD, UPDATE, or DELETE actions must be specified. LOCATION, METADATA, or both are used to clarify if the action is to apply to the physical security information in file system, to the metadata objects in the SAS Metadata Server, or to both. The REPAIR statement can be followed by one or more TABLES statements to perform the same action on the specified data sets. However, you cannot specify a new password or encryption key value in a TABLES statement that follows a REPAIR statement. For more information, see "REPAIR Statement" on page 89.

Using the TABLES Statement with the REMOVE Statement

You can use a TABLES statement with TABLESONLY=YES in the REMOVE statement to only remove the location information and secured table objects for specific tables in the metadata-bound library. If you do not use TABLESONLY=YES with a TABLES statement, then the secured library object and all secured table objects are deleted by the REMOVE statement.

When you use the TABLES statement after the REMOVE statement, an ENCRYPT=NO option removes the encryption on the data set as the table is being removed. For more information, see "Encrypted Data Set Considerations" on page 69. This process is necessary only if the administrator is trying to remove the passwords or encryption of a data set.

If you are removing the binding of the physical library to metadata or the physical library is not bound to a secured library, then you might want to modify the data set passwords or encryption to some other value. You are not restricted to changing to a common metadata-bound library password or encryption. You might choose to specify both a current and new password or current and new encryption key separated by a slash (/) in the REMOVE statement. If you want the different data sets to have unique passwords or encryption, then use the following two steps:

1. Change the PW= option for the data sets using a REMOVE statement with TABLESONLY=YES and an individual TABLES statement for each unique password and encryption.
2. Remove the metadata-bound library using a REMOVE statement **without** TABLESONLY=YES.

Using the TABLES Statement with the REPORT Statement

The TABLES statement is syntactically accepted with the REPORT statement but has little use. Specifying TABLES limits the report to the tables listed if used. For more information, see "REPORT Statement" on page 94.

Using the AUTHLIB Procedure

Requirements for Using the AUTHLIB Statements

Except for the REPORT statement, all statements within the AUTHLIB procedure require that you must meet the following criteria:

- The SAS session runs under an account that has host-layer control of the target physical library. To ensure that only users who have host control can bind a physical library to metadata, the SAS session must run under a privileged host account as follows:
    - On UNIX, the account must be the owner of the directory.
    - On Windows, the account must have full control of the directory.
    - On z/OS, for UNIX file system libraries, the account must be the owner of the directory.
    - On z/OS, for direct-access bound libraries, the account must have RACF ALTER access authority to the library data set.
- The SAS session connects to the SAS Metadata Server as an identity that has ReadMetadata and WriteMemberMetadata permissions to the target secured data folder.
- You must supply the password(s) in CREATE, MODIFY, REPAIR, and REMOVE statements.

The REPORT statement requirements are less restrictive and are documented with that statement.

Copy-In-Place Operation

In the SAS 9.4 release, the copy-in-place operation is used to re-encrypt data sets.

Prior to the second maintenance release of SAS 9.4, metadata-bound data sets in different representations other than the host environment executing the AUTHLIB code fails in CREATE, MODIFY, REPAIR, and REMOVE actions. In the second maintenance release of SAS 9.4, the copy-in-place operation is used to bind or alter bindings of most metadata-bound data files and view files that are accessed through CEDA (Cross-Environment Data Access). However, metadata-bound data sets accessed through CEDA that contain indexes, extended attributes, and integrity constraints are detected and the copy-in-place operation is not attempted as it would still fail.

The following steps are performed in the copy-in-place operation:

1. The data set is renamed to _TEMP_ENCRYPT_FILE_NAME_.
2. The data set is copied back to the original data set name, which re-encrypts the data in the process.
3. The _TEMP_ENCRYPT_FILE_NAME_ file is deleted.

See the following SAS log examples of the copy-in-place operation:

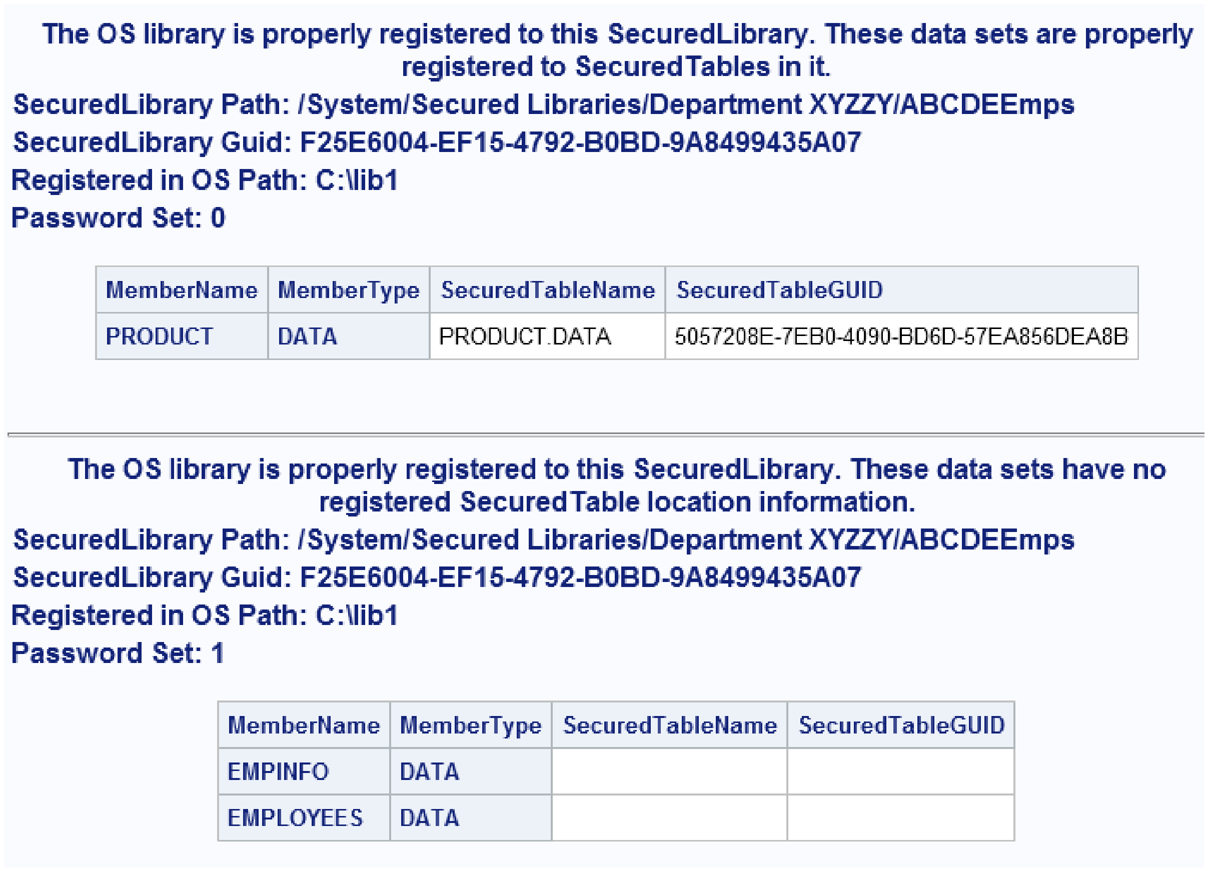
- “Example 12: Binding a Library with an Optional Recorded Encryption Key When Existing AES-Encrypted Data Sets Have Different Encryption Keys ” on page 118

- “Example 16: Changing a Metadata-Bound Library to Require AES Encryption When Existing Data Sets Are Encrypted with Different Encryption Keys” on page 129

Results: AUTHLIB Procedure

The REPORT statement produces the following output.

Output A2.1 *Using the REPORT Statement*



The OS library is properly registered to this SecuredLibrary. These data sets are properly registered to SecuredTables in it.

SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps

SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07

Registered in OS Path: C:\lib1

Password Set: 0

MemberName	MemberType	SecuredTableName	SecuredTableGUID
PRODUCT	DATA	PRODUCT.DATA	5057208E-7EB0-4090-BD6D-57EA856DEA8B

The OS library is properly registered to this SecuredLibrary. These data sets have no registered SecuredTable location information.

SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps

SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07

Registered in OS Path: C:\lib1

Password Set: 1

MemberName	MemberType	SecuredTableName	SecuredTableGUID
EMPINFO	DATA		
EMPLOYEES	DATA		

Examples: AUTHLIB Procedure

Example 1: Binding a Physical Library That Contains Unprotected Data Sets

Features: PROC AUTHLIB statement options
CREATE statement options:
PW=
SECUREDLIBRARY=

SECUREDFOLDER=

Details

This example demonstrates binding a physical library that contains data sets that do not have passwords or AES encryption.

Program

```
proc authlib lib=zyxwvut;

    create securedfolder="Department XYZZY"
        securedlibrary="ZYXWVUTEmps"
        pw=secretpw;
run;
quit;
```

Program Description

Library ZYXWVUT contains three data sets that do not have passwords: Employees, EmpInfo, Product.

```
proc authlib lib=zyxwvut;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Specify metadata-bound library passwords with the PW= option.

```
    create securedfolder="Department XYZZY"
        securedlibrary="ZYXWVUTEmps"
        pw=secretpw;
run;
quit;
```

Results: The library and data sets are bound with the password *secretpw*. The binding is straightforward, as PROC AUTHLIB has unhindered access to the data.

Log Examples

Log A2.2 *Unprotected Data Sets*

```
79   proc authlib lib=zyxwvut;
80
81    create securedfolder="Department XYZZY"
82           securedlibrary="ZYXWVUTEmps"
83           pw=XXXXXXXX;
84
85   run;

NOTE: Successfully created a secured library object for the physical library ZYXWVUT and recorded its
      location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ZYXWVUTEmps
           SecuredLibraryGUID: 1A323C03-A3D8-4A83-9615-2BC2CB9FAAE2
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ZYXWVUTEmps" for data set ZYXWVUT.EMPINFO.DATA.
NOTE: The passwords on ZYXWVUT.EMPINFO.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ZYXWVUTEmps" for data set
      ZYXWVUT.EMPLOYEES.DATA.
NOTE: The passwords on ZYXWVUT.EMPLOYEES.DATA were successfully modified.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ZYXWVUTEmps" for data set ZYXWVUT.PRODUCT.DATA.
NOTE: The passwords on ZYXWVUT.PRODUCT.DATA were successfully modified.
86   quit;
```

Example 2: Binding a Physical Library That Contains Password-Protected Data Sets

Features: PROC AUTHLIB statement options
CREATE statement options:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

Details

This example demonstrates what happens if you use a similar CREATE statement as Example 1 when the physical library contains two data sets that have the same READ=, WRITE=, and ALTER= passwords and one data set that does not have any passwords. None of the data sets are AES-encrypted.

Program

```
proc authlib lib=abcde;

    create securedfolder="Department XYZZY"
        securedlibrary="ABCDEEmps"
        pw=secretpw;
run;
quit;
```

Program Description

Library ABCDE has Employees, EmpInfo, and Product data sets. However, in library ABCDE, the Employees and EmpInfo data sets are protected with a READ= password **`abcd`**, WRITE= password **`efgh`**, and an ALTER= password **`ijkl`** before the library is secured by the statements. The third data set, Product, is not protected with passwords.

```
proc authlib lib=abcde;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Specify metadata-bound library passwords with the PW= option.

```
       create securedfolder="Department XYZZY"
           securedlibrary="ABCDEEmps"
           pw=secretpw;
run;
quit;
```

Results: The ABCDE library is bound and the unprotected Product data set is bound and the password was set. The protected data sets are not bound and their passwords did not change because their current passwords were not specified.

Log Examples

Log A2.3 *Password-Protected Data Sets*

```
179  proc authlib lib=abcde;
180
181   create securedfolder="Department XYZZY"
182          securedlibrary="ABCDEEmps"
183          pw=XXXXXXXX;
184
185  run;

NOTE: Successfully created a secured library object for the physical library ABCDE and recorded its
      location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 4881263D-C346-41F7-AC49-BF9181AF13D2
ERROR: The ALTER password is the most restrictive on ABCDE.EMPINFO.DATA. You must supply its value in
       order to alter or add any passwords.
ERROR: The ALTER password is the most restrictive on ABCDE.EMPLOYEES.DATA. You must supply its value
       in order to alter or add any passwords.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.PRODUCT.DATA.
NOTE: The passwords on ABCDE.PRODUCT.DATA were successfully modified.
NOTE: Some statement actions not processed because of errors noted above.
186  quit;

NOTE: The SAS System stopped processing this step because of errors.
```

Example 3: Binding a Library When Existing Data Sets Are Protected with the Same Passwords

Features: PROC AUTHLIB statement options
CREATE statement options:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

Details

This example demonstrates how to specify the passwords for the Employees and EmpInfo data sets from the preceding example in the PROC AUTHLIB CREATE statement. None of the data sets are AES-encrypted.

Program

```
proc authlib lib=abcde;

    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=ijkl/secretpw;
run;
quit;
```

Program Description

Library ABCDE also has Employees, EmpInfo, and Product data sets. However, in library ABCDE, the Employees and EmpInfo data sets are protected with a READ= password **`abcd`**, WRITE= password **`efgh`**, and ALTER= password **`ijkl`** before the library is secured by the statements. The third data set, Product, is not protected with any passwords.

```
proc authlib lib=abcde;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Specify the ALTER= password **`ijkl`** for the data sets in the PW= argument before the new password *secretpw*, separated by a slash (/).

```
    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=ijkl/secretpw;
run;
quit;
```

Results: The library ABCDE is bound. All three data sets are bound with the same password *secretpw*.

Log Examples

Log A2.4 *Securing a Library with Data Sets That Are Protected with the Same Passwords*

```
39   proc authlib lib=abcde;
40   create securedlibrary="ABCDEEmps"
41   securedfolder="Department XYZZY"
42   pw=XXXX/XXXXXXXX;
43   run;

NOTE: Successfully created a secured library object for the physical library ABCDE and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 9F746F86-2336-4E2F-A67E-BFB77DEC27F0
NOTE: Successfully added new secured table object "DEPTNAME.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.
NOTE: The passwords on ABCDE.DEPTNAME.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPINFO.DATA.
NOTE: The passwords on ABCDE.EMPINFO.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPLOYEE.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEE.DATA.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA were successfully modified.
44   quit;
```

Example 4: Binding a Library When Existing Data Sets Are Protected with Different Passwords

Features: PROC AUTHLIB statement options
- CREATE statement options:
- ALTER=
- READ=
- SECUREDLIBRARY=
- SECUREDFOLDER=
- WRITE=
- TABLE statement options:
- ALTER=
- PW=
- READ=
- WRITE=

Details

This example demonstrates how to bind the library KLMNO, which contains three data sets with different passwords. None of the data sets are AES-encrypted. It also demonstrates creating a longer metadata-bound library password by specifying the READ=, WRITE=, and ALTER= password options.

Program

```
proc authlib lib=klmno;
```

```
       create securedlibrary="KLMNOEmps"
           securedfolder="Department XYZZY"
           read=abcdefgh
           write=ijklmno
           alter=pqrstuvw;

       tables employees /
           pw=lmno;
       tables empinfo /
           read=abcd
           write=efgh
           alter=ijkl;
       tables product;
   run;
   quit;
```

Program Description

Library KLMNO has Employees, EmpInfo, and Product data sets. The Employees data set is protected with the PW= password **`lmno`**. The EmpInfo data set is protected with a READ= password **`abcd`**, a WRITE= password **`efgh`**, and an ALTER= password **`ijkl`**. The Product data set is not protected.

```
   proc authlib lib=klmno;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Specify the values for READ= password **`abcdefgh`**, WRITE= password **`ijklmno`**, and ALTER= password **`pqrstuvw`** to create a longer metadata-bound library password.

```
       create securedlibrary="KLMNOEmps"
           securedfolder="Department XYZZY"
           read=abcdefgh
           write=ijklmno
           alter=pqrstuvw;
```

Use the TABLES statement to specify the current password for each data set. When using TABLES statements, a TABLES statement must be specified for all data sets.

```
       tables employees /
           pw=lmno;
       tables empinfo /
           read=abcd
           write=efgh
           alter=ijkl;
       tables product;
   run;
   quit;
```

Results: The library KLMNO is bound, and all three data sets are bound with the same passwords. The passwords are READ= password **`abcdefgh`**, WRITE= password **`ijklmno`**, and ALTER= password **`pqrstuvw`**.

Log Examples

Log A2.5 *Securing a Library with Existing Data Sets That Are Protected with Different Passwords*

```
177  libname klmno "c:\lib2";
NOTE: Libref KLMNO was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\lib2
178
179  proc authlib lib=klmno;
180  create securedlibrary="KLMNOEmps"
181  securedfolder="Department XYZZY"
182  read=XXXXXXXX
183  write=XXXXXXX
184  alter=XXXXXXXX;
185  tables employees /
186  pw=XXXX;
187  tables empinfo /
188  read=XXXX
189  write=XXXX
190  alter=XXXX;
191  tables product;
192  run;

NOTE: Successfully created a secured library object for the physical library KLMNO and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     KLMNOEmps
           SecuredLibraryGUID: BC74E81F-E86B-402E-8C16-F9A94A078F81
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPLOYEES.DATA.
NOTE: The passwords on KLMNO.EMPLOYEES.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPINFO.DATA.
NOTE: The passwords on KLMNO.EMPINFO.DATA were successfully modified.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.PRODUCT.DATA.
NOTE: The passwords on KLMNO.PRODUCT.DATA were successfully modified.
193  quit;
```

Example 5: Changing Passwords on Data Sets

Features: PROC AUTHLIB statement options
- MODIFY statement options:
- PW=
- TABLESONLY=
- TABLES statement options:
- PW=

Details

This example shows a different approach for modifying the passwords of existing data sets to match the metadata-bound library passwords. It uses the MODIFY statement. Here, the MODIFY statement is used to modify the data set passwords of the Employees and EmpInfo data sets from Example 2 on page 102 to match the metadata-bound library password. Neither of these data sets are AES-encrypted.

The MODIFY statement can also be used to modify the data set passwords of data sets that are copied into a metadata-bound library by operating system commands after the library has been bound.

Program

```
proc authlib lib=abcde;

    modify tablesonly=yes
        pw=secretpw;

    tables _all_ /
        pw=ijkl/secretpw;
run;
quit;
```

Program Description

Library ABCDE has Employees, EmpInfo, and Product data sets. The library is bound with metadata-bound library password *secretpw*. However, in library ABCDE, the Employees and EmpInfo data sets are not bound to the library and are protected with an ALTER= password **ijkl**. The third data set, Product, is already bound.

```
proc authlib lib=abcde;
```

The MODIFY statement is used to modify the data set passwords of the Employees and EmpInfo data sets to match the metadata-bound library password. The TABLESONLY= statement specifies to modify table passwords only.

```
    modify tablesonly=yes
        pw=secretpw;
```

A TABLES statement must be specified. The existing data sets' ALTER password is specified in the PW= argument before the metadata-bound password, separated by a slash (/) in the TABLES statement.

```
    tables _all_ /
        pw=ijkl/secretpw;
run;
quit;
```

Results: All three data sets are now bound with the *secretpw* password.

Log Examples

Log A2.6 *Changing Data Set Passwords*

```
76   proc authlib lib=abcde;
77   modify tablesonly=yes
78   pw=XXXXXXXX;
79   tables _all_ /
80   pw=XXXX/XXXXXXXX;
81   run;

NOTE: The passwords on ABCDE.DEPTNAME.DATA do not require modification.
NOTE: The passwords on ABCDE.EMPINFO.DATA do not require modification.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA do not require modification.
82   quit;
```

Example 6: Changing Metadata-Bound Library Passwords

Features: PROC AUTHLIB statement options
  MODIFY statement options:
  PW=
  SECUREDLIBRARY=
  SECUREDFOLDER=

Details

This example demonstrates how to use the MODIFY statement to change the library passwords if you believe that the metadata-bound library passwords have been compromised. The following code changes the library passwords and the data set passwords of all data sets in the library that use the specified passwords or do not have a password. In this example, no data sets are AES-encrypted. See later examples if your library has AES-encrypted data.

Program

```
proc authlib lib=abcde;

    modify securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secretpw/new-password;
run;
quit;
```

Program Description

Library ABCDE requires a password change.

```
proc authlib lib=abcde;
```

Use the MODIFY statement to change the library passwords and the data set passwords. Note that the name of the secured library object and the name of the metadata folder are optional, but can be specified to ensure that the library is bound to that secured library object before making the change. This is used when the SAS Management Console submits the code from the Modify action to ensure that the correct operation system library path was specified.

```
       modify securedlibrary="ABCDEEmps"
           securedfolder="Department XYZZY"
           pw=secretpw/new-password;
   run;
   quit;
```

Results: The library ABCDE remains bound and the library password is modified to the *new-password*. All three data sets remain bound, and their passwords are modified with *new-password*. An error message would be displayed in the SAS log for any data set that had a password other than *secretpw*.

Log Examples

***Log A2.7** Changing Metadata-bound Library Passwords*

```
217  proc authlib lib=abcde;
218    modify securedlibrary="ABCDEEmps"
219        securedfolder="Department XYZZY"
220        pw=XXXXXXXX/XXXXXXXX;
221
222  run;

NOTE: The passwords for the secured library object with path "/System/Secured Libraries/Department
      XYZZY/ABCDEEmps" were successfully modified."
NOTE: The passwords on ABCDE.EMPINFO.DATA were successfully modified.
NOTE: The passwords on ABCDE.EMPLOYEES.DATA were successfully modified.
NOTE: The passwords on ABCDE.PRODUCT.DATA were successfully modified.
223  quit;
```

Example 7: Using the REMOVE Statement

Features: PROC AUTHLIB statement options
REMOVE statement options:
PW=

Details

This example demonstrates how to unbind a metadata-bound library. The code does the following:

- deletes metadata that describes the library and its tables from the SAS Metadata Repository
- removes security bindings from the physical library and data sets
- removes the assigned password from the data sets, leaving them unprotected

The slash (/) after the password is optional and is used to remove or replace the password from the data sets. If a library is bound with READ=, WRITE=, and ALTER=

passwords, as in Example 4 on page 105, then you must specify all of the passwords, and they must each have a slash (/). None of the data sets are AES-encrypted.

Program

```
proc authlib lib=abcde;

    remove
        pw=currntpw/;
run;
quit;
```

Program Description

Unbinding the metadata-bound library ABCDE.

```
proc authlib lib=abcde;
```

Use the REMOVE statement to unbind the metadata-bound library. The slash (/) after the password is used to remove the password from the data sets.

```
    remove
        pw=currntpw/;
run;
quit;
```

Results: The library ABCDE and all the data sets that are bound to it are no longer bound. All passwords are removed from the unbound data sets making them unprotected.

Log Examples

***Log A2.8** Unbinding a Metadata-Bound Library*

```
195  proc authlib lib=abcde;
196  remove
197  pw=XXXXXXXX/;
198  run;

WARNING: Some or all the passwords on ABCDE.DEPTNAME.DATA were removed along with the secured library
object location,
         leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.DEPTNAME.DATA was successfully removed.
WARNING: Some or all the passwords on ABCDE.EMPINFO.DATA were removed along with the secured library
object location, leaving
         the data set unprotected.
NOTE: The secured table object location for ABCDE.EMPINFO.DATA was successfully removed.
WARNING: Some or all the passwords on ABCDE.EMPLOYEE.DATA were removed along with the secured library
object location,
         leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.EMPLOYEE.DATA was successfully removed.
NOTE: Successfully deleted the secured library object that was located at:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 9F746F86-2336-4E2F-A67E-BFB77DEC27F0
NOTE: Successfully deleted the recorded location of the secured library object for the physical
library ABCDE.
199  quit;
```

Example 8: Using the REPORT Statement

Features: PROC AUTHLIB statement options
Report statement

Details

This example demonstrates how to check a library's bindings.

Program

```
proc authlib lib=abcde;

    report;
run;
quit;
```

Program Description

Check the bindings of the metadata-bound library ABCDE.

```
proc authlib lib=abcde;
```

Use the REPORT statement.

```
      report;
run;
quit;
```

Results: For the REPORT statement results, see“Output Example” on page 113.

Log Examples

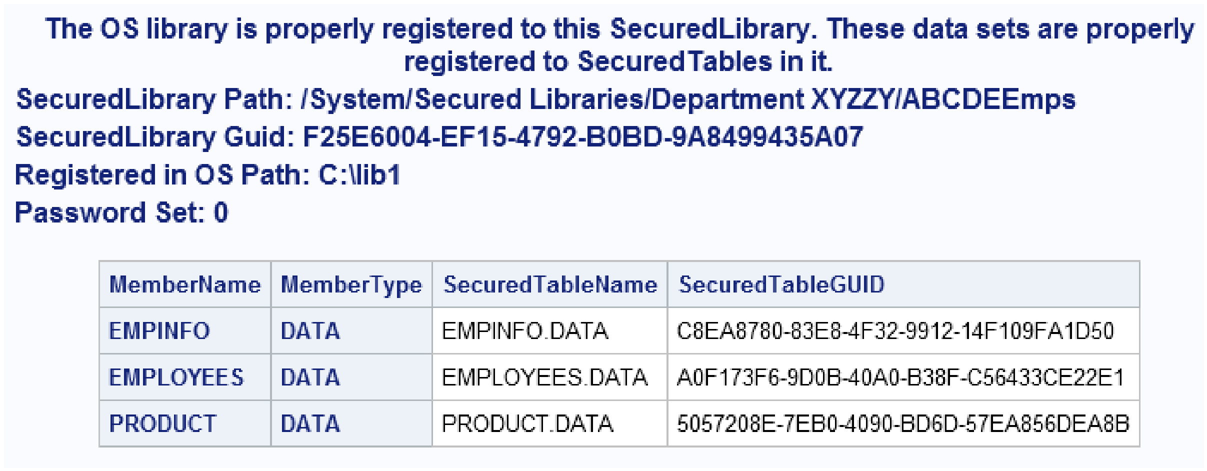
Log A2.9 *Creating a Report*

```
49   proc authlib lib=abcde;
50    report;
51   run;

52   quit;
```

Output Example

Output A2.2 *REPORT Statement Results for the ABCDE Library*



The OS library is properly registered to this SecuredLibrary. These data sets are properly registered to SecuredTables in it.

SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps

SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07

Registered in OS Path: C:\lib1

Password Set: 0

MemberName	MemberType	SecuredTableName	SecuredTableGUID
EMPINFO	DATA	EMPINFO.DATA	C8EA8780-83E8-4F32-9912-14F109FA1D50
EMPLOYEES	DATA	EMPLOYEES.DATA	A0F173F6-9D0B-40A0-B38F-C56433CE22E1
PRODUCT	DATA	PRODUCT.DATA	5057208E-7EB0-4090-BD6D-57EA856DEA8B

Example 9: Using the TABLES Statement

Features: PROC AUTHLIB statement options
- CREATE statement options:
- ALTER=
- READ=
- SECUREDLIBRARY=
- SECUREDFOLDER=
- WRITE=
- TABLE statements options:
- ALTER=
- PW=
- READ=
- WRITE=

Details

Example 4 on page 105 demonstrates how to use the TABLES statement.

Example 10: Binding a Library When Existing Data Sets Are SAS Proprietary Encrypted

Features: PROC AUTHLIB statement options
CREATE statement options:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=
TABLES statement options:
PW=
READ=

Details

The following example demonstrates how to bind and change passwords on SAS Proprietary encrypted data sets.

Program

```
proc authlib lib=klmno;

    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;

    tables employees /
        pw=lmno;
    tables empinfo /
        read=abcd;
    tables product;
run;
quit;
```

Program Description

Library KLMNO has three data sets: Employees, EmpInfo, and Product. In this library, the Employees data set is protected with the PW= password **`lmno`**. The EmpInfo data set is protected with a READ= password **`abcd`**. Both Employees and EmpInfo data sets are SAS Proprietary encrypted. The Product data set is not protected.

```
proc authlib lib=klmno;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Set the library password to **`pqrstuvw`**.

```
    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;
```

Because these data sets have different passwords, a TABLES statement must be specified for all data sets in order to change their passwords.

```
       tables employees /
           pw=lmno;
       tables empinfo /
           read=abcd;
       tables product;
   run;
   quit;
```

Results: The library KLMNO is bound. All three data sets are bound and use the same PW= password **`pqrstuvw`**. Data sets Employees and EmpInfo are copied-in-place to encrypt with the password **`pqrstuvw`**. Data set Product is bound, but not encrypted.

Log Examples

Log A2.10 *TABLES Statement for the KLMNO Library Containing a SAS Proprietary Data Set*

```
265  proc authlib lib=klmno;
266  create securedlibrary="KLMNOEmps"
267  securedfolder="Department XYZZY"
268  pw=XXXXXXXX;
269  tables employees /
270  pw=XXXX;
271  tables empinfo /
272  read=XXXX;
273  tables product;
274  run;

NOTE: Successfully created a secured library object for the physical library KLMNO and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     KLMNOEmps
           SecuredLibraryGUID: E71881CD-8C54-4E21-A8B5-FD7D4FBDAA7D
NOTE: Copying data set KLMNO.EMPLOYEES in place to encrypt with the new secured library passwords or
encryption options.
NOTE: Renaming the data set KLMNO.EMPLOYEES to KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__ to KLMNO.EMPLOYEES.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPLOYEES.DATA.
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPLOYEES.DATA.
NOTE: There were 5 observations read from the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set KLMNO.EMPLOYEES has 5 observations and 6 variables.
NOTE: Deleting the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on KLMNO.EMPLOYEES.DATA were successfully modified.
NOTE: Copying data set KLMNO.EMPINFO in place to encrypt with the new secured library passwords or
encryption options.
NOTE: Renaming the data set KLMNO.EMPINFO to KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__ to KLMNO.EMPINFO.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPINFO.DATA.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPINFO.DATA.
NOTE: There were 5 observations read from the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set KLMNO.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on KLMNO.EMPINFO.DATA were successfully modified.
NOTE: The passwords on KLMNO.PRODUCT.DATA do not require modification.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.PRODUCT.DATA.
275  quit;
```

Example 11: Binding a Library When Existing Data Sets Are AES-Encrypted

Features: PROC AUTHLIB statement options
- CREATE statement options:
- PW=
- SECUREDLIBRARY=
- SECUREDFOLDER=
- TABLES statement option:
- ENCRYPTKEY=

Details

This example demonstrates how to bind data sets that are AES-encrypted. None of the data sets have passwords.

CAUTION:

SAS strongly recommends that you not have AES-encrypted data sets with different encryption keys in metadata-bound libraries, like this example creates. Instead, SAS recommends that you record a default encryption key in metadata and convert all AES-encrypted data sets to use that key. Doing this, your users, and programs do not have to specify the key when opening the data sets. The examples following this example show you how to do this process.

Program

```
proc authlib lib=klmno;

    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;

    tables employees /
        encryptkey=lmno;
    tables empinfo /
        encryptkey=abcd;
    tables product;
run;
quit;
```

Program Description

Library KLMNO has three data sets: Employees, EmpInfo, and Product. In this library, the Employees data set is AES-encrypted and has the ENCRYPTKEY= value **`lmno`**. The EmpInfo data set is AES-encrypted and has the ENCRYPTKEY= value **`abcd`**. The Product data set is not protected.

```
proc authlib lib=klmno;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. Set the library password to **`pqrstuvw`**.

```
    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;
```

Using the TABLES statements, specify the encrypt key for each data set. A TABLES statement must be specified for all data sets.

```
    tables employees /
        encryptkey=lmno;
    tables empinfo /
        encryptkey=abcd;
    tables product;
run;
quit;
```

Results: The library KLMNO is bound. All three data sets are bound. The Employees and EmpInfo data sets remain AES-encrypted. The Product data set is not encrypted. The encrypt key values for the Employees and Empinfo data sets are different. SAS strongly recommends that you not have AES-encrypted data sets with different encryption keys in metadata-bound libraries, like this example created.

Log Examples

Log A2.11 *TABLES Statement for the KLMNO Library Containing AES-Encrypted Data Sets*

```
351  proc authlib lib=klmno;
352  create securedlibrary="KLMNOEmps"
353  securedfolder="Department XYZZY"
354  pw=XXXXXXXX;
355  tables employees /
356  encryptkey=XXXX;
357  tables empinfo /
358  encryptkey=XXXX;
359  tables product;
360  run;

NOTE: Successfully created a secured library object for the physical library KLMNO and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     KLMNOEmps
           SecuredLibraryGUID: 48E2C4C7-ADE1-49D2-BBFE-14E5EAAB8961
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPLOYEES.DATA.
NOTE: The passwords on KLMNO.EMPLOYEES.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPINFO.DATA.
NOTE: The passwords on KLMNO.EMPINFO.DATA were successfully modified.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.PRODUCT.DATA.
NOTE: The passwords on KLMNO.PRODUCT.DATA were successfully modified.
361  quit;
```

Example 12: Binding a Library with an Optional Recorded Encryption Key When Existing AES-Encrypted Data Sets Have Different Encryption Keys

Features: PROC AUTHLIB statement options
- CREATE statement options:
- ENCRYPT=
- ENCRYPTKEY=
- PW=
- SECUREDLIBRARY=
- SECUREDFOLDER=

TABLES statement options:
ENCRYPT=
ENCRYPTKEY=

Details

This example demonstrates how to bind a library with an optional recorded encryption key. None of the data sets have passwords.

Since some SAS code existed that created and references the EmpInfo data set with ENCRYPTKEY=DEF and since the recorded library key is not required, the specification of the ENCRYPTKEY=DEF should be removed from the code. Any code that re-creates the data must keep the ENCRYPT=AES option so that the optional recorded key is used when the data set is re-created.

Program

```
proc authlib lib=abcde;

    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        encrypt=aes
        encryptkey=optionalkey;


    tables employee;
    tables empinfo /
        encryptkey=def/optionalkey
        encrypt=aes;
    tables deptname;
run;
quit;
```

Program Description

Library ABCDE has Employees, EmpInfo, and DeptName data sets. In this library, the EmpInfo data set is AES-encrypted and has the ENCRYPTKEY= value **`def`**.

```
proc authlib lib=abcde;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. The optional encrypt key is specified for the metadata-bound library.

```
    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        encrypt=aes
        encryptkey=optionalkey;
```

A TABLES statement is required for each data set.

```
    tables employee;
```

```
        tables empinfo /
            encryptkey=def/optionalkey
            encrypt=aes;
        tables deptname;
    run;
    quit;
```

Results: The ABCDE library is bound and the optional encrypt key is stored. When the statements are executed, the following happens to the three data sets. The Employee data set is updated with the new metadata-bound library password but is not encrypted. The DeptName data set is updated with the metadata-bound library password but is not encrypted. The EmpInfo data set is copied to re-encrypt with the optional recorded key and gets the new metadata-bound library password. Note that it is necessary to supply both the current and new optional key in the TABLES statement for EmpInfo in the following program. Without the new key specification, the data set would remain encrypted with the **`def`** key.

Log Examples

Log A2.12 *Changing an Encryption Key Value to the Recorded Encryption Key*

```
467  libname abcde "c:\lib1";
NOTE: Libref ABCDE was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\lib1
468
469  proc authlib lib=abcde;
470  create securedlibrary="ABCDEEmps"
471  securedfolder="Department XYZZY"
472  pw=XXXXXX
473  encrypt=aes
474  encryptkey=XXXXXXXXXXX;
475  tables employee;
476  tables empinfo /
477  encryptkey=XXX/XXXXXXXXXXX
478  encrypt=aes;
479  tables deptname;
480  run;

NOTE: Successfully created a secured library object for the physical library ABCDE and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 8E683650-B306-4871-A92D-16D481EC6456
NOTE: Successfully added new secured table object "EMPLOYEE.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEE.DATA.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA were successfully modified.
NOTE: Copying data set ABCDE.EMPINFO in place to encrypt with the new secured library passwords or
encryption options.
NOTE: Renaming the data set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.
NOTE: Metadata-bound library permissions are used for ABCDE.EMPINFO.DATA.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPINFO.DATA.
NOTE: There were 5 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPINFO.DATA were successfully modified.
NOTE: Successfully added new secured table object "DEPTNAME.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.
NOTE: The passwords on ABCDE.DEPTNAME.DATA were successfully modified.
480  quit;
```

Example 13: Binding a Library with Required AES Encryption When Existing Data Sets Are Encrypted with the Same Encryption Key

Features: PROC AUTHLIB statement options
- CREATE statement options:
- ENCRYPT=
- ENCRYPTKEY=
- PW=
- REQUIRE_ENCRYPTION
- SECUREDLIBRARY=

SECUREDFOLDER=

Details

This example demonstrates how to bind a library with requiring that all of the data sets in this metadata-bound library have AES encryption and have the same encryption key.

Program

```
proc authlib lib=abcde;

    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=abc ;
run;
quit;
```

Program Description

Library ABCDE has three data sets: Employees, EmpInfo, and DeptName. Data set EmpInfo has encryption key value of **`abc`**. The other two data sets are not AES-encrypted. None of the data sets have passwords.

```
proc authlib lib=abcde;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. REQUIRE_ENCRYPTION=YES specifies that all data sets in the metadata-bound library are automatically AES-encrypted.

```
    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=abc ;
run;
quit;
```

Results: The library ABCDE is bound, and all of the data sets are bound and AES-encrypted with the same encryption key.

Log Examples

Log A2.13 *Library ABCDE Requiring AES Encryption When the Data Sets Are Already Encrypted with the Same Encryption Key*

```
40   proc authlib lib=abcde;
41   create  seclib="ABCDEEmps"
42         securedfolder="Department XYZZY"
43         pw=XXXXXX
44         require_encryption=yes
45         encrypt=aes
46         encryptkey=XXX ;
47   run;

NOTE: Setting library to require encryption.
NOTE: Required encryption will use AES encryption with the recorded key.


NOTE: Successfully created a secured library object for the physical library ABCDE and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 9FD6C5D9-EF00-4CDC-8D0A-348D08BB329E
NOTE: Copying data set ABCDE.DEPTNAME in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.
NOTE: Metadata-bound library permissions are used for ABCDE.DEPTNAME.DATA.
NOTE: Successfully added new secured table object "DEPTNAME.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.
NOTE: There were 10 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.DEPTNAME has 10 observations and 2 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.DEPTNAME.DATA were successfully modified.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPINFO.DATA.
NOTE: The passwords on ABCDE.EMPINFO.DATA were successfully modified.
NOTE: Copying data set ABCDE.EMPLOYEE in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: Metadata-bound library permissions are used for ABCDE.EMPLOYEE.DATA.
NOTE: Successfully added new secured table object "EMPLOYEE.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEE.DATA.
NOTE: There were 22 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 22 observations and 11 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA were successfully modified.
48   quit;
```

Example 14: Changing the Encryption Key on a Metadata-Bound Library That Requires AES Encryption

Features: PROC AUTHLIB statement options
MODIFY statement options:

ENCRYPT=
ENCRYPTKEY=
PW=

Details

This example demonstrates how to use the MODIFY statement to change the stored library encryption key if you believe that the metadata-bound library encryption keys might have been compromised.

Program

```
proc authlib lib=abcde;

    modify
       pw=secret
       encrypt=aes
       encryptkey=/new;

run;
quit;
```

Program Description

Library ABCDE has three data sets: Employees, EmpInfo, and DeptName. In this library, all data sets are AES-encrypted with encryption key value **`abc`** since AES encryption is required for the metadata bound library.

```
proc authlib lib=abcde;
```

Use the MODIFY statement to change the library encryption key and the data set encryption key. You must specify ENCRYPT=AES.

```
    modify
       pw=secret
       encrypt=aes
       encryptkey=/new;

run;
quit;
```

Results: The library ABCDE remains bound with the same password and a new encryption key. All three data sets remain bound with the same password and a new encryption key. Note that the data sets were copied-in-place to be encrypted with the new key value.

Log A2.14 *Changing the Encryption Key ABCDE Library*

```
502  proc authlib lib=abcde;
503  modify
504  pw=XXXXXX
505  encrypt=aes
506  encryptkey=/XXX;
507  run;

NOTE: Changing the required encryption key.


NOTE: The passwords on ABCDE.DEPTNAME.DATA do not require modification.
NOTE: Copying data set ABCDE.DEPTNAME in place to do required encryption with
the library's required encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.
NOTE: There were 4 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.DEPTNAME has 4 observations and 2 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPINFO.DATA do not require modification.
NOTE: Copying data set ABCDE.EMPINFO in place to do required encryption with the
library's required encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.
NOTE: There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA do not require modification.
NOTE: Copying data set ABCDE.EMPLOYEE in place to do required encryption with
the library's required encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords and/or encryption options for the secured library object
with path "/System/Secured Libraries/Department
      XYZZY/ABCDEEmps" were successfully modified."
NOTE: All data sets in library ABCDE are properly protected with the metadata-
bound library passwords and encryption options.
      Replaced Passwords and encryption keys were purged.
NOTE: Purged 1 versions of the replaced passwords and encryption keys older than
2015-05-04T15:40:57-05:00.
508  quit;
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: There were 22 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 22 observations and 11 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords and/or encryption options for the secured library object
with path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" were successfully modified.
```

Example 15: Binding a Library with Existing Data Sets That Are AES-Encrypted with Different Encryption Keys

Features: PROC AUTHLIB statement options
CREATE statement options:
ENCRYPT=
ENCRYPTKEY=
PW=
REQUIRE_ENCRYPTION
SECUREDLIBRARY=
SECUREDFOLDER=
TABLES statement option:
ENCRYPTKEY=

Details

This example demonstrates how to change all data sets in the metadata-bound library that contain different encryption keys to have the required AES encryption and have the same encryption key. None of the data sets have passwords.

Program

```
proc authlib lib=abcde;


    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new ;

    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
quit;
```

Program Description

Library ABCDE has three data sets: Employee, EmpInfo, and DeptName. The Employee and EmpInfo data sets are already AES-encrypted with different keys. The DeptName data set is not encrypted.

```
proc authlib lib=abcde;
```

Using the CREATE statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. REQUIRE_ENCRYPTION=YES specifies that all data sets in the metadata-bound library are automatically AES-encrypted.

```
       create  seclib="ABCDEEmps"
           securedfolder="Department XYZZY"
           pw=secret
           require_encryption=yes
           encrypt=aes
           encryptkey=new ;
```

Using the TABLES statement, specify the encrypt key for each data set. TABLES statements are required for each data set.

```
       tables employee /
           encryptkey=abc;
       tables empinfo /
           encryptkey=def;
       tables deptname ;
   run;
   quit;
```

Results: The library ABCDE is bound. All data sets in the metadata-bound library ABCDE have been copied-in-place to be encrypted with the required key.

Log Examples

Log A2.15 *Library ABCDE Requiring AES Encryption When Each Data Set Has Different Encryption Key Values*

```
554  proc authlib lib=abcde;
555  create seclib="ABCDEEmps"
556  securedfolder="Department XYZZY"
557  pw=XXXXXX
558  require_encryption=yes
559  encrypt=aes
560  encryptkey=XXX ;
561  tables employee /
562  encryptkey=XXX;
563  tables empinfo /
564  encryptkey=XXX;
565  tables deptname ;
566  run;

NOTE: Setting library to require encryption.
NOTE: Required encryption will use AES encryption with the recorded key.


NOTE: Successfully created a secured library object for the physical library ABCDE and recorded its
location as:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 097E9A84-D6E8-488E-B779-1E2AB0670036
NOTE: Copying data set ABCDE.EMPLOYEE in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: Metadata-bound library permissions are used for ABCDE.EMPLOYEE.DATA.
NOTE: Successfully added new secured table object "EMPLOYEE.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEE.DATA.
NOTE: There were 5 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA were successfully modified.
NOTE: Copying data set ABCDE.EMPINFO in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.
NOTE: Metadata-bound library permissions are used for ABCDE.EMPINFO.DATA.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object at path
"/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPINFO.DATA.
NOTE: There were 5 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPINFO.DATA were successfully modified.
NOTE: Copying data set ABCDE.DEPTNAME in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.
NOTE: Metadata-bound library permissions are used for ABCDE.DEPTNAME.DATA.
NOTE: Successfully added new secured table object "DEPTNAME.DATA" to the secured library object at
path "/System/Secured
      Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.
NOTE: There were 4 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.DEPTNAME has 4 observations and 2 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.DEPTNAME.DATA were successfully modified.
567  quit;
```

Example 16: Changing a Metadata-Bound Library to Require AES Encryption When Existing Data Sets Are Encrypted with Different Encryption Keys

Features: PROC AUTHLIB statement options
MODIFY statement options:
ENCRYPT=
ENCRYPTKEY=
PW=
REQUIRE_ENCRYPTION
SECUREDLIBRARY=
SECUREDFOLDER=
TABLES statement option:
ENCRYPTKEY=

Details

This example is similar to the previous example. The difference is that the library is already bound to metadata, so the MODIFY statement is used to change the binding to require AES encryption.

Program

```
proc authlib lib=abcde;

    modify  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new;

    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
quit;
```

Program Description

Library ABCDE has three data sets: Employees, EmpInfo, and DeptName. In this library, the Employees data set has the encryption key value **`abc`**. The EmpInfo data set has the encryption key value **`def`**. The DeptName data set is not AES-encrypted.

```
proc authlib lib=abcde;
```

Using the MODIFY statement, enter the name of the metadata folder and name the secured library object in the SAS Metadata Server. You use the REQUIRE_ENCRYPTION=YES option to require that all data sets in the metadata-bound library have AES encryption. Note that the name of the secured library object and the name of the metadata folder are optional, but can be specified to ensure that the library is bound to that secured library object before making the change.

```
       modify  seclib="ABCDEEmps"
           securedfolder="Department XYZZY"
           pw=secret
           require_encryption=yes
           encrypt=aes
           encryptkey=new;
```

Using the TABLES statement, specify the encrypt key for each data set. TABLES statements are required for each data set.

```
       tables employee /
           encryptkey=abc;
       tables empinfo /
           encryptkey=def;
       tables deptname ;
   run;
   quit;
```

Results: The library ABCDE remains bound. The MODIFY statement changed the binding to require AES encryption. All three data sets are copied-in-place to encrypt the data sets with the required encrypt key..

Log Examples

Log A2.16 *Library ABCDE Requiring AES Encryption and Changing the Encryption Key Values of Each Data Set to a Recorded Encryption Key Value*

```
628  proc authlib lib=abcde;
629  modify seclib="ABCDEEmps"
630  securedfolder="Department XYZZY"
631  pw=XXXXXX
632  require_encryption=yes
633  encrypt=aes
634  encryptkey=XXX;
635  tables employee /
636  encryptkey=XXX;
637  tables empinfo /
638  encryptkey=XXX;
639  tables deptname ;
640  run;

NOTE: Changing library to require encryption.
NOTE: Required encryption will use AES encryption with the recorded key.


NOTE: The passwords on ABCDE.EMPLOYEE.DATA do not require modification.
NOTE: Copying data set ABCDE.EMPLOYEE in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: There were 5 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.EMPINFO.DATA do not require modification.
NOTE: Copying data set ABCDE.EMPINFO in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.
NOTE: There were 5 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords on ABCDE.DEPTNAME.DATA do not require modification.
NOTE: Copying data set ABCDE.DEPTNAME in place to do required encryption with the library's required
encryption key and
      passwords.
NOTE: Renaming the data set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.
NOTE: There were 4 observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.DEPTNAME has 4 observations and 2 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The passwords and/or encryption options for the secured library object with path "/System/
Secured Libraries/Department
      XYZZY/ABCDEEmps" were successfully modified."
641  quit;
```

Example 17: Using the REMOVE Statement on a Metadata-Bound Library with Required AES Encryption

Features: PROC AUTHLIB statement options
REMOVE statement options:

PW=
ENCRYPT=

Details

This example demonstrates how to unbind a metadata-bound library. The code does the following:

- deletes metadata that describes the library and its tables from the SAS Metadata Repository
- removes security bindings from the physical library and data sets
- removes the assigned password and encryption from the data sets, leaving them unprotected

The slash (/) after the password is optional and is used to remove or replace the password from the data sets. If a library is bound with READ=, WRITE=, and ALTER= passwords, as in Example 4 on page 105, then you must specify all of the passwords, and they must each have a slash (/).

Program

```
proc authlib lib=abcde;


    remove
        pw=currntpw/
        encrypt=no;
run;
quit;
```

Program Description

Unbinding the metadata-bound library ABCDE.

```
proc authlib lib=abcde;
```

Use the REMOVE statement to unbind the metadata-bound library. The slash (/) after the password is used to remove the password from the data sets. ENCRYPT=NO specifies that encryption is removed from all data sets.

```
    remove
        pw=currntpw/
        encrypt=no;
run;
quit;
```

Results: The library ABCDE and all the data sets bound to it are no longer bound. All passwords and encryption are removed from the unbound data sets making them unprotected.

Log A2.17 *Using the REMOVE Statement on a Metadata-Bound Library with Required AES Encryption*

```
642  proc authlib lib=abcde;
643  remove
644  pw=XXXXXX/
645  encrypt=no;
646  run;

NOTE: Copying data set ABCDE.DEPTNAME in place to remove encryption.
NOTE: Renaming the data set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.
NOTE: There were 4 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.DEPTNAME has 4 observations and 2 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
WARNING: Some or all the passwords on ABCDE.DEPTNAME.DATA were removed along
with the secured library object location,
         leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.DEPTNAME.DATA was successfully
removed.
NOTE: Copying data set ABCDE.EMPINFO in place to remove encryption.
NOTE: Renaming the data set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.
NOTE: There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPINFO has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
WARNING: Some or all the passwords on ABCDE.EMPINFO.DATA were removed along with
the secured library object location, leaving
         the data set unprotected.
NOTE: The secured table object location for ABCDE.EMPINFO.DATA was successfully
removed.
NOTE: Copying data set ABCDE.EMPLOYEE in place to remove encryption.
NOTE: Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.
NOTE: There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
NOTE: The data set ABCDE.EMPLOYEE has 5 observations and 6 variables.
NOTE: Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.
WARNING: Some or all the passwords on ABCDE.EMPLOYEE.DATA were removed along
with the secured library object location,
         leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.EMPLOYEE.DATA was successfully
removed.
NOTE: Successfully deleted the secured library object that was located at:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 157F7ACD-5B71-4BC3-A490-DCED4BD275E8
NOTE: Successfully deleted the recorded location of the secured library object
for the physical library ABCDE.
647  quit;
```

Example 18: Resetting Credentials on Imported SecuredLibrary Objects

Features: PROC AUTHLIB statement options
  MODIFY statement options:
  LIBRARY=
  PW=
  ENCRYPT=

ENCRYPTKEY=

Details

This example shows how to reset the passwords and encryption key on SecuredLibrary objects that are imported from a backup package.

- The LIBNAME statement without the AUTHADMIN=YES option fails because there are no associated password values restored by the import.
- The AUTHADMIN=YES option is used to enable the AUTHLIB procedure to execute with the binding information in the physical library.
- The MODIFY statement is used to reset the metadata-bound library passwords and encryption key value on the library from "Example 13: Binding a Library with Required AES Encryption When Existing Data Sets Are Encrypted with the Same Encryption Key" on page 121 assuming that the SecuredLibrary object was imported from a backup package without those values.

Program

```
libname abcde "sas-library" ;

libname abcde "sas-library" authadmin=yes;

proc authlib lib=abcde;
    modify
        pw=secret
        encrypt=aes
        encryptkey=value;
run;
quit;

libname abcde "sas-library";
```

Program Description

Library ABCDE has three data sets: Employees, EmpInfo, and DeptName. This LIBNAME statement fails because there are no associated password values.

```
libname abcde "sas-library" ;
```

Use the AUTHADMIN=YES option. The AUTHADMIN=YES option enables the AUTHLIB procedure to execute with the binding information in the physical library.

```
libname abcde "sas-library" authadmin=yes;
```

Use the MODIFY statement to reset the metadata-bound library passwords and encryption key value. The PW= option resets the password. The ENCRYPTKEY= option resets the encryption key value.

```
proc authlib lib=abcde;
    modify
        pw=secret
        encrypt=aes
        encryptkey=value;
run;
quit;
```

Reissue the LIBNAME statement without the AUTHADMIN=YES option . It is good practice to reassign the library without AUTHADMIN=YES as soon as your administrative need is complete, so that any other access that you make to the library is not in administrative mode. In this case, it also ensures that the credentials are reset.

```
libname abcde "sas-library";
```

Log A2.18 *Resetting Credentials*

```
253 libname abcde "library-name" ;
ERROR: The secured library object information for library ABCDE could not be obtained
       from the metadata server or has invalid data.
ERROR: Association not found.
ERROR: Error in the LIBNAME statement.
254  libname abcde "library-name" authadmin=yes;
NOTE: Libref ABCDE was successfully assigned as follows:
      Engine:            V9
      Physical Name:     library-name
      Secured Library:   /System/Secured Libraries/Department XYZZY/ABCDEEmps
      Authenticated ID:  user-id@site as user-id
      Encryption Key:    YES
      Require Encryption: YES
255  proc authlib lib=abcde;
256  modify
257        pw=XXXXXX
258        encrypt=aes
259        encryptkey=XXX ;
260  run;

NOTE: Required encryption will use AES encryption with the recorded key.


NOTE: The passwords on ABCDE.DEPTNAME.DATA do not require modification.
NOTE: The passwords on ABCDE.EMPINFO.DATA do not require modification.
NOTE: The passwords on ABCDE.EMPLOYEE.DATA do not require modification.
261  quit;
```

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Intelligence Platform: Overview*
- *SAS Intelligence Platform: Data Administration Guide*
- *SAS Intelligence Platform: Security Administration Guide*
- *SAS Management Console: Guide to Users and Permissions*
- *Base SAS Procedures Guide*

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books

Glossary

Base SAS
the core product that is part of SAS Foundation and is installed with every deployment of SAS software. Base SAS provides an information delivery system for accessing, managing, analyzing, and presenting data.

data set
See SAS data set.

encryption
the conversion of data by the use of algorithms or other means into an unintelligible form in order to secure data (for example, passwords) in transmission and in storage.

identity
See metadata identity.

library reference
See libref.

libref (library reference)
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

metadata identity (identity)
a metadata object that represents an individual user or a group of users in a SAS metadata environment. Each individual and group that accesses secured resources on a SAS Metadata Server should have a unique metadata identity within that server.

metadata object
a set of attributes that describe a table, a server, a user, or another resource on a network. The specific attributes that a metadata object includes vary depending on which metadata model is being used.

metadata repository
a collection of related metadata objects, such as the metadata for a set of tables and columns that are maintained by an application.

metadata server
a server that provides metadata management services to one or more client applications.

metadata-bound library
a physical SAS library that is tied to a corresponding secured library object. All access from SAS to a metadata-bound library is subject to the requesting user's effective permissions on the corresponding metadata object.

metadata-bound table
a physical SAS data set (a table or view) that is tied to a corresponding secured table object. All access from SAS to a metadata-bound table is subject to the requesting user's effective permissions on the corresponding metadata object.

procedure
See SAS procedure.

SAS data set (data set)
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views.

SAS procedure (procedure)
a type of SAS language element that refers to a self-contained program for performing a specific task, such as to produce reports, to manage files, or to analyze data.

SAS statement (statement)
a type of SAS language element that is used to perform a particular operation in a SAS program or to provide information to a SAS program.

SAS system option (system option)
a type of SAS language element that is applied to any of a number of operations during a SAS session. System options can control SAS session initialization, SAS interactions with hardware and software, and input and output processing of SAS files.

SAS table
the visual rendering of a SAS data set in tabular format. *See also* SAS data set.

SAS view
a type of SAS data set that retrieves data values from other files. A SAS view contains only descriptor information such as the data types and lengths of the variables (columns), plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. SAS views can be created by the SAS DATA step, as well as by the SAS SQL procedure.

secured data folder
a SAS metadata object that serves as a specialized container for secured library objects, as part of the metadata-bound libraries feature. In each metadata repository, the first secured data folder is Secured Libraries, in the System folder. Additional secured data folders can be added only beneath a Secured Libraries folder.

secured library object
a SAS metadata object to which a physical SAS library is bound. Metadata-layer permissions on each secured library object manage access to its corresponding physical library. Secured library objects are stored beneath a repository's Secured Libraries folder.

secured table object
a SAS metadata object to which a physical SAS data set (a table or view) is bound. Metadata-layer permissions on each secured table object manage access to its corresponding physical data set. Each secured table object is stored beneath a secured library object.

statement
See SAS statement.

submit
to perform an action that causes a software application such as SAS to compile and execute a program.

system option
See SAS system option.

Index