THE POWER TO KNOW.

SAS® 9.3 Guide to Metadata-Bound Libraries

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2012. *SAS® 9.3 Guide to Metadata-Bound Libraries*. Cary, NC: SAS Institute Inc.

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

1st electronic book, August 2012

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

SAS 9.3 Guide to Metadata-Bound Libraries

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).

Only administrators who set up and maintain metadata-bound libraries need to know the information that this document contains. 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 36.

Requirements

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

Support for metadata-bound libraries begins in the second maintenance release of SAS 9.3.

Accessibility

For information about accessibility for a SAS product, see the online Help for that product or send e-mail to accessibility@sas.com.

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 support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

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. The CREATE statement of the AUTHLIB procedure generates a new metadata object and binds the physical library to that object.

Each physical table within a metadata-bound library has information in its header that points to a specific metadata object (a secured table object). The pointer creates a security binding between the physical table and the metadata object. The binding ensures that SAS universally enforces metadata-layer permission requirements for the physical table—regardless of how a user requests access from SAS.

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

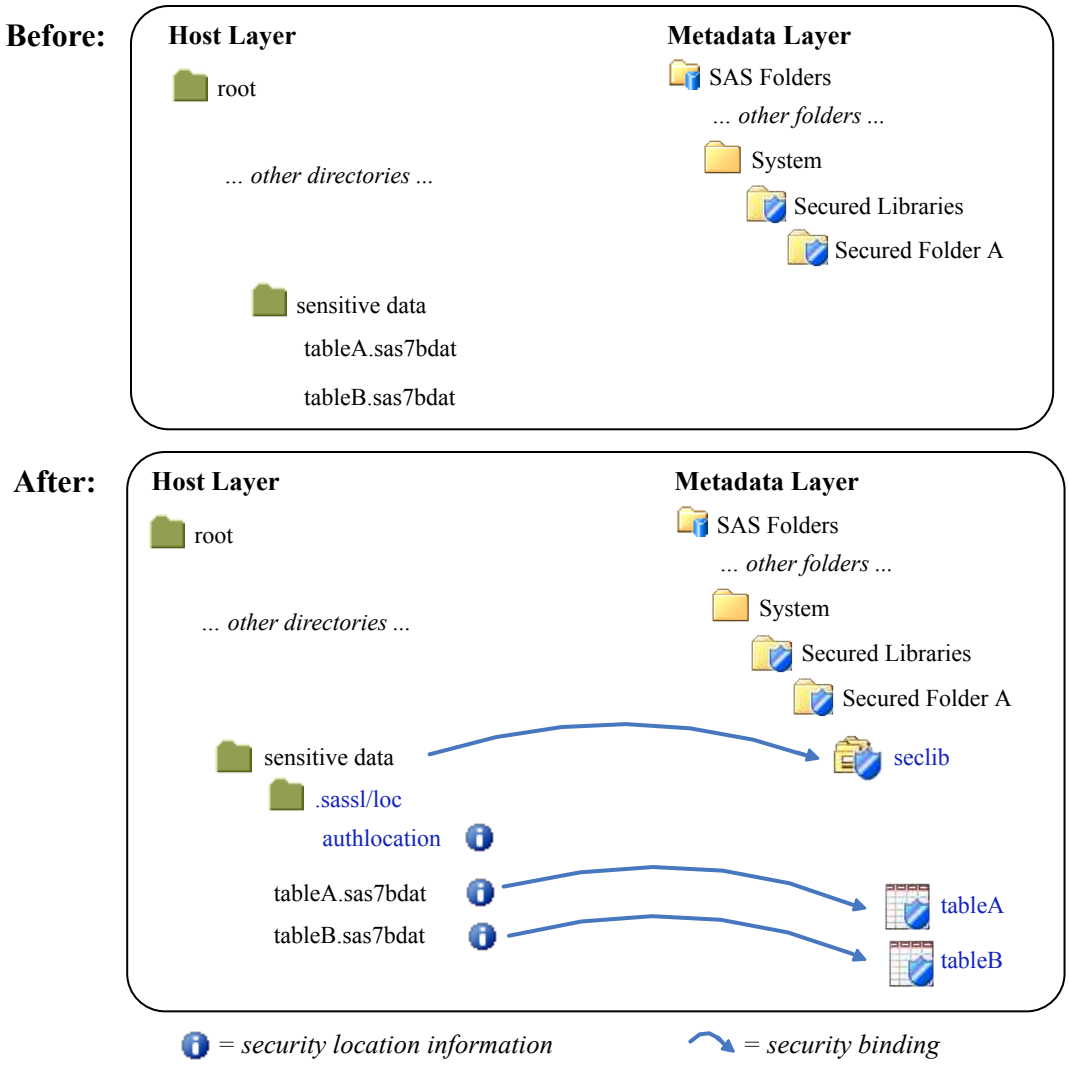
- The requesting user can connect to the metadata server in which the corresponding object is stored.
- The requesting user's metadata identity has all required metadata-layer effective permissions for the requested action.
- The host identity with which the data is retrieved has all required host-layer 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 physical 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 by the CREATE statement of the AUTHLIB procedure.

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*.

    Note: 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.

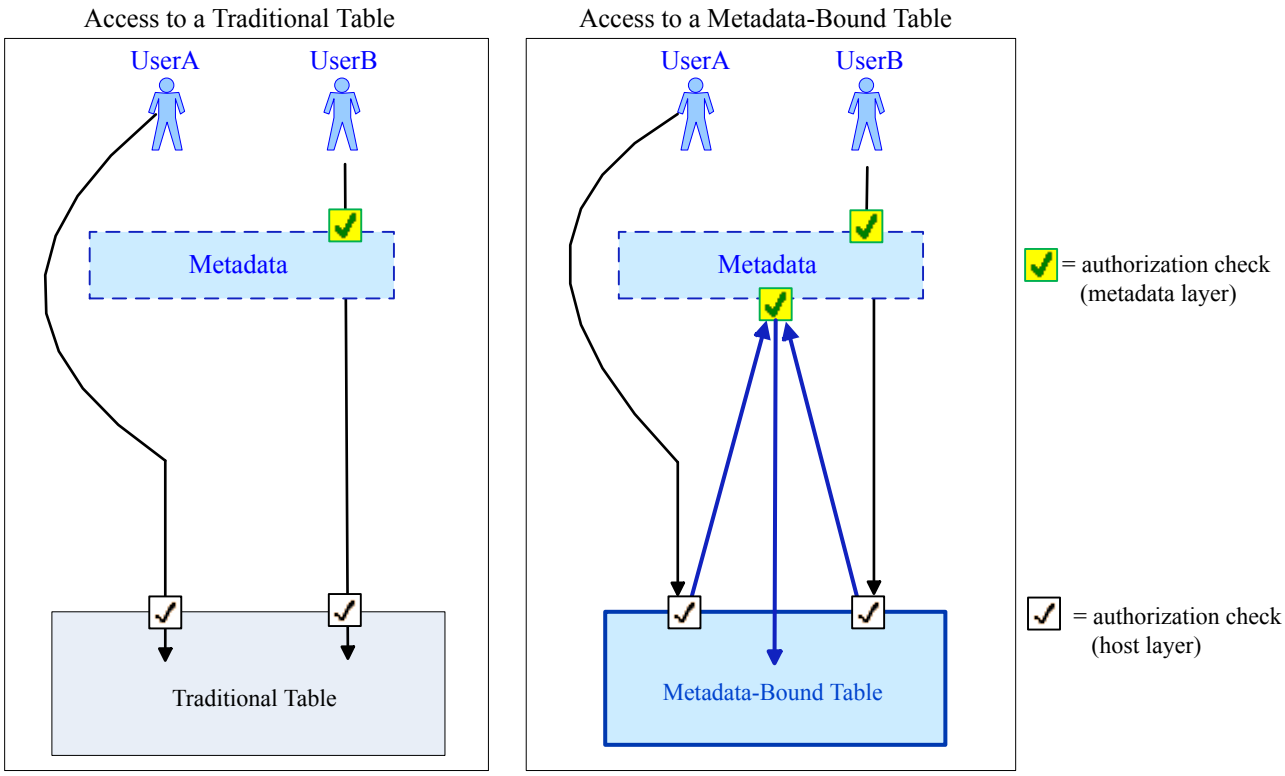
  - For a physical table, the security information consists of information in the header. The corresponding metadata object is called a secured table object. In the figure, *tableA* and *tableB* are secured table objects that correspond to the physical metadata-bound tables *tableA.sas7bdat* and *tableB.sas7bdat*.
- 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 of the physical tables already had a different password, the presence of that password would prevent that table from being affected by the CREATE statement. To move from that situation to a best practice state (where, for clarity, all tables within the security library are protected by that library's bindings), use the MODIFY statement of the AUTHLIB procedure.

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)*



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 34.

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 28
- "Host Access to SAS Tables" in Chapter 12 of *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 recreate 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 30.
- In the current release, there is no graphical user interface (GUI) for creating metadata-bound libraries. You use SAS code to perform that task. See "Overview of Setting Up a Metadata-Bound Library" on page 9.
- In the current release, metadata-bound libraries don't support column-level permissions or metadata-based permission conditions. 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 "Set Up Fine-Grained Access" on page 15.
- 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, but can't be bound to a traditional library object. Similarly, tables within a metadata-bound library are bound to corresponding secured table objects, not to 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 34.

  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.

Note: 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 standalone 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 know how to write and submit simple SAS code.
- 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).

- As the first release of the secured libraries feature, this release might not offer optimal usability for the administrative tasks.
- 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.

Chapter 2

Tasks for Metadata-Bound Libraries

Setting Up a Metadata-Bound Library

Overview of Setting Up a Metadata-Bound Library

Setting up a metadata-bound library involves the following tasks:

1. In the SAS metadata, below a **`/System/Secured Libraries/`** folder, identify or create an appropriately secured folder for the data.
2. In SAS code, submit a CREATE statement (within the AUTHLIB procedure) that references your physical data directory and the metadata folder that you identified or created in step 1.

3. If you want to support access from clients that use metadata in order to locate data, make sure that the physical library and tables are also registered 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.

TIP Binding a physical library introduces additional constraints on access, so it is a good practice to review existing access patterns beforehand. For help resolving any unanticipated disruptions in end-user access, see "Facilitate End-User Access" on page 39.

Who Uses the CREATE Statement?

Administrators use the CREATE statement of the AUTHLIB procedure to bind a physical library to metadata.

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

- Your 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, your 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.
- Your SAS session connects to the metadata server as an identity that has Read and Write access (the ReadMetadata and WriteMemberMetadata permissions) to the target secured data folder.

Introductory Demonstration

As a simple demonstration, limit access to a library that contains tables copied from the SASHELP library.

To prepare:

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. In a SAS session, assign the libref **secdemo** to the new directory.

   ```
   libname secdemo 'path-to-your-test-directory';
   ```

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

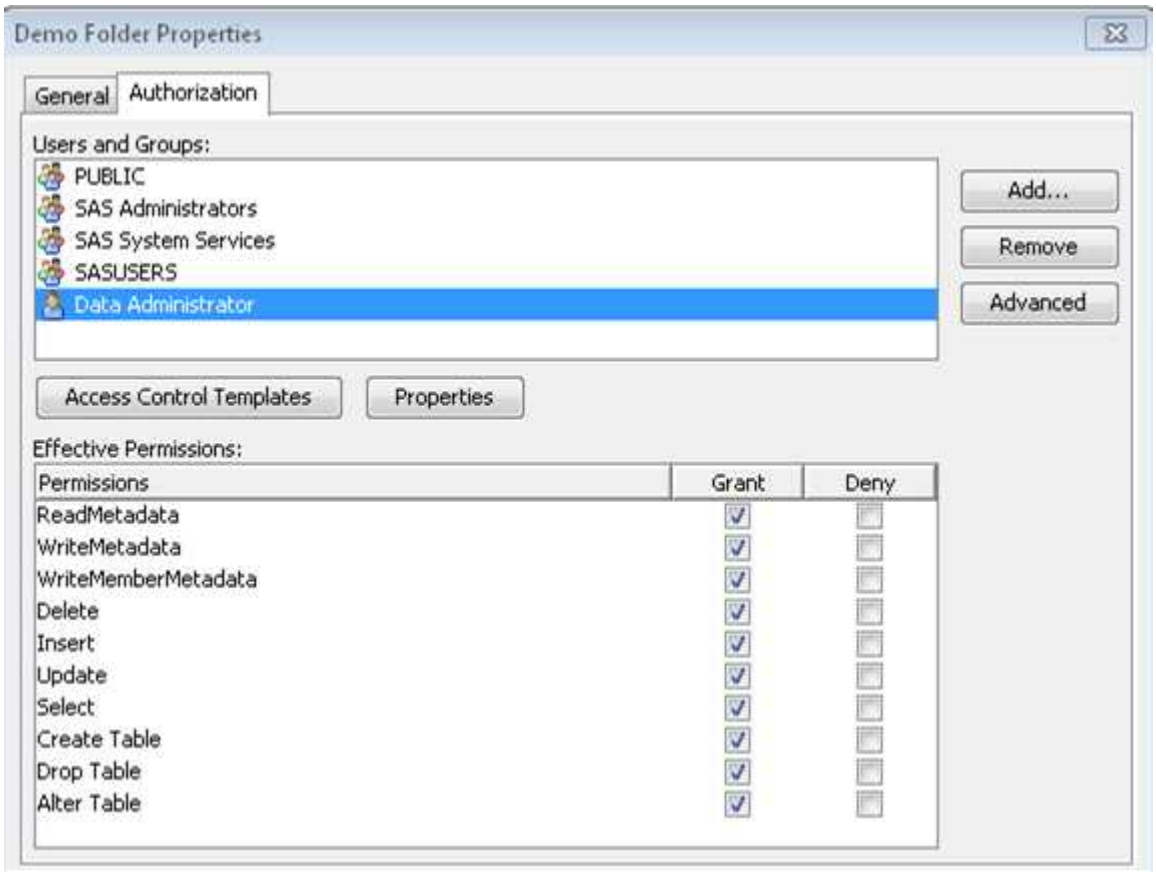
1. Create an appropriately secured metadata location.
   a. Log on to SAS Management Console as someone who has the ReadMetadata and WriteMemberMetadata permissions on the **/System/Secured Libraries**

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.



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).

2. In the same SAS session in which you assigned the **`secdemo`** libref, submit the following code:

```
proc authlib library=secdemo;
 create
   securedfolder='/System/Secured Libraries/Demo Folder'
   securedlibrary='Demo Library'
   pw=secret;

run;
```

Here are some details about the preceding code:

- The LIBRARY= option references your previously defined physical library. This is the library for which a corresponding metadata-bound library object will be created and in which security location information will be stored.
- The SECUREDFOLDER= option specifies the metadata folder location for the new secured library object. To minimize exposure, we suggest that you specify a folder for which you have already made any necessary adjustments to access (as discussed in step 1c above).
- The SECUREDLIBRARY= argument specifies a name for the new secured library object (in metadata).
- The PW= value is an initial assignment of a password for the library. This value can be encoded using the PWENCODE procedure. In order to submit an encoded password, you must enclose the value in quotation marks.

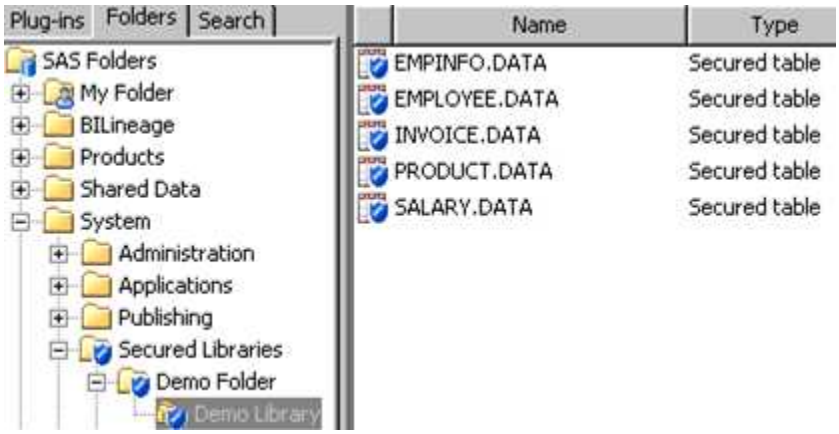
CAUTION:

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

- Your SAS session must be able to connect to the target metadata server as an identity that has the ReadMetadata and WriteMemberMetadata permissions on the target folder (**`SAS Folders/System/Secured Libraries`**).

  Note: The preceding code does not explicitly supply connection information for the metadata server. This example assumes that your SAS session already knows how to connect to the target metadata server.

- After the code runs, each table in your test library is represented in the metadata as a new secured table object (a child of the new secured library object). The following image depicts the new objects in SAS Management Console.



3. 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).

   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).

Set Up Mutually Exclusive Access

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.

The following 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). The example assumes that the following prerequisites are met:

- GroupA and GroupB exist in the SAS metadata.
- The data exists in the host, and each physical library has been assigned a libref (liba1, liba2, libb1, and libb2) in your SAS session.
- You have a host account that has host-layer access to the data.
- Your SAS session knows how to connect to the metadata server using an account that has the ReadMetadata and WriteMemberMetadata permissions on the folder within which you will add the secured data folders (**`SAS Folders/System/Secured Libraries`**).

Here are the implementation steps:

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
`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 don't allow members of GroupA and GroupB to update or delete data.

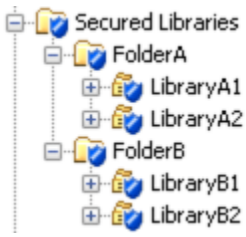
TIP To add GroupA and GroupB to the **Authorization** tab, click the **Add** button next to the **Users and Groups** list box. In SAS Management Console, an explicit setting has a white background color (not gray or green).

4. To bind the physical data to metadata, submit SAS 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.

```
proc authlib;
   create
      library=liba1
      securedfolder='FolderA'
      securedlibrary='LibraryA1'
      pw=secret;
   create
      library=liba2
      securedfolder='FolderA'
      securedlibrary='LibraryA2'
      pw=secret;
   create
      library=libb1
      securedfolder='FolderB'
      securedlibrary='LibraryB1'
      pw=secret2;
   create
      library=libb2
      securedfolder='FolderB'
      securedlibrary='LibraryB2'
      pw=secret2;
run;
```

TIP In the SECUREDFOLDER= option, if you supply a path that does not begin with a slash (/), the path is calculated relative to **`/System/Secured Libraries/`**.

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



TIP If the new secured library objects are not immediately visible, right-click the **`Secured Libraries`** folder and select **Refresh** from the popup 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.

7. 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.

Set Up Fine-Grained Access

Overview

To provide 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, use the AUTHLIB procedure and metadata-layer permissions to protect 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 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.

Note: This example also demonstrates how you can explicitly provide metadata server connection information in an OPTIONS statement.

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

- "CREATE Statement" on page 55
- "Metadata Server Connection Options" on page 36

- Chapter 27, "SAS Views," in *SAS Language Reference: Concepts*
- "Connection Options " in Chapter 5 of *SAS Language Interfaces to Metadata*

Validating a Metadata-Bound Library

About Validating a Metadata-Bound Library

For a specified physical library, the REPORT statement of the AUTHLIB procedure lists any missing or mismatched physical tables, security location information, and metadata objects.

TIP In general, you use the REPORT statement against a metadata-bound library. However, you can also use this statement against a traditional library in order to determine whether any individual physical tables within that library are bound to metadata.

Who Uses the REPORT Statement?

Administrators use the REPORT statement of the AUTHLIB procedure 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:

- Your 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.
- Your 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.

Example

In this example, you use the REPORT statement to determine whether there are any inconsistencies between a metadata-bound library and its corresponding metadata objects.

```
libname secdemo 'path';

proc authlib library=secdemo;
   report;
run;
```

Note: The preceding code does not explicitly supply connection information for the metadata server. This example assumes that your SAS session already knows how to connect to the target metadata server.

In this example, no inconsistencies exist. The following image depicts output from the preceding code.

The OS library is properly registered to this SecuredLibrary. These data sets are properly registered to SecuredTables in it.
SecuredLibrary Path: /System/Secured Libraries/Demo Folder/Demo Library
SecuredLibrary Guid: 56984631-323B-46FF-B9BB-3FDB00550EA8
Registered in OS Path: \\machine\directory
Password Set: 0

Obs	MemberName	MemberType	SecuredTableName	SecuredTableGUID
1	EMPINFO	DATA	EMPINFO.DATA	BA70AF05-07F2-403A-875A-23B8ADF31316
2	EMPLOYEE	DATA	EMPLOYEE.DATA	E4418A38-55D0-4B5C-9CE4-590A85EBD57B
3	INVOICE	DATA	INVOICE.DATA	9F7BAC91-C22D-4FCD-8014-26220FC836C2
4	PRODUCT	DATA	PRODUCT.DATA	1ACCB68B-DA96-494A-AE6E-3968CB46C810
5	SALARY	DATA	SALARY.DATA	EB1E4DDA-28E6-4579-9FB6-6C4342144235

See Also

- "REPORT Statement" on page 63
- "Metadata Server Connection Options" on page 36

Binding an Individual Table to Metadata

About Binding Tables to Metadata

Binding data to metadata is a library-level feature. In general, all tables within a metadata-bound library are protected by that library (and its secured table objects) and share the library's password (or passwords). If an inconsistency arises, you can use the MODIFY statement of the AUTHLIB procedure to perform the following tasks:

- Apply the library's password (or passwords) to any unbound tables.
- Update any mismatched table passwords, so they match the library passwords.

For example, if someone uses a host copy command to add physical tables to a metadata-bound library, the added tables are not automatically bound. In order to create corresponding metadata objects, you must apply the password (or passwords) of the metadata-bound library to the added physical tables.

Who Uses the MODIFY Statement?

Administrators use the MODIFY statement of the AUTHLIB procedure to add or update physical tables so that they contain the same password (or passwords) as their parent metadata-bound library.

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

- Your 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 change passwords, your 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.
- You know the current password (or passwords) for the metadata-bound library.
- Your SAS session connects to the metadata server as an identity that has Read and Write access (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 Chapter 3 of *SAS Intelligence Platform: Security Administration Guide*.

Example

In this example, you use the MODIFY statement to apply passwords to physical tables that were added to the library through a host copy command. The following code applies passwords to any unsecured physical tables in the secdemo library and binds those tables to metadata (creating corresponding secured table objects).

```
libname secdemo 'path';

proc authlib library=secdemo;
   modify pw=secret;
run;
```

Here are some details about the preceding code:

- The preceding code does not include the SECUREDFOLDER and SECUREDLIBRARY parameters. It is not necessary to use these parameters, because the physical directory of the specified library (secdemo) contains information that references a particular metadata folder and secured library object.
- The preceding code affects the physical library and all of the tables that it contains. If you don't want to affect the library, set the TABLESONLY option. If you want to affect only some of the tables, add a TABLES statement after the MODIFY statement. If you use one or more TABLES statements after a MODIFY statement, only the specified tables are processed.

  Note: In this example, all of the copied tables are initially unsecured. If any of the copied tables were already secured, with a different password, those tables would not be affected by the preceding code.
- The preceding code does not explicitly supply connection information for the metadata server. This example assumes that your SAS session already knows how to connect to the target metadata server.

CAUTION:
If you lose the password (or passwords) for a metadata-bound library, you can't unbind the library or change its passwords. Be sure to keep track of passwords that you assign in the CREATE and MODIFY statements of the AUTHLIB procedure.

TIP This code creates a new secured table object in metadata. Whenever you create new secured library or secured table objects, you should review the permissions on those objects in SAS Management Console. You can adjust access if needed. For

example, you can 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).

See Also

- "MODIFY Statement" on page 56
- "Metadata Server Connection Options" on page 36

Unbinding a Metadata-Bound Library

About Unbinding a Library

The REMOVE statement of the AUTHLIB procedure deletes both physical and metadata content as follows:

- The REMOVE statement deletes physical security information from the library directory and table headers.
- The REMOVE statement deletes corresponding secured library and secured table objects from the SAS metadata.

TIP 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 in which such physical tables exist, their security location information is unaffected by the REMOVE statement, unless you specify AUTHADMIN=YES on the LIBNAME statement. You should use the AUTHADMIN=YES option on the LIBNAME statement in this circumstance.

Who Uses the REMOVE Statement?

Administrators use the REMOVE statement of the AUTHLIB procedure to remove protection from a metadata-bound library.

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

- Your 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 unbind a physical library from metadata, your 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.
- Your SAS session connects to the metadata server as an identity that has Read and Write access (the ReadMetadata, WriteMetadata, and WriteMemberMetadata permissions) to the target secured data folder, secured library object, and secured table objects.

- You know the password (or passwords) for the current metadata-bound library. You must supply the password in the REMOVE statement.

Example

In this example, you use the REMOVE statement to remove the metadata binding from an existing physical library.

```
libname secdemo 'path';

proc authlib library=secdemo;
   remove pw=secret/;
run;
```

Here are some important points about the preceding code:

- A forward slash (/) is entered at the end of the password. For a password change, the slash separates the current and new passwords. For password removal, the slash indicates the end of the current password that is being removed and is not being replaced with another password).
- The SECUREDFOLDER and SECUREDLIBRARY parameters are not specified. It is not necessary to use these parameters, because the physical directory of the specified library (secdemo) contains information that references a particular secured folder and secured library object.
- The preceding code deletes the corresponding secured library object and secured table objects from the metadata.

  TIP If those objects are still visible in SAS Management Console, right-click the **Secured Libraries** folder and select **Refresh** from the popup menu.
- Connection information for the metadata server is not explicitly supplied. This example assumes that your SAS session already knows how to connect to the target metadata server.

Note: The REMOVE statement unbinds only those physical tables that are located within the specified library (directory). If a table is host-copied from a metadata-bound library to another location, that table's security location information will be unaffected by subsequent REMOVE statements against the metadata-bound library. In order to reestablish access to the table, corrective action against the table is necessary.

See Also

- "REMOVE Statement" on page 58
- "Metadata Server Connection Options" on page 36

Verifying Read Access to Metadata-Bound Data

Who Can Read Metadata-Bound Data?

In order to read metadata-bound data, you must be connected 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

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

- Chapter 15, “DATASETS Procedure,” in *Base SAS Procedures Guide*
- “Metadata Server Connection Options” on page 36

Best Practices for Metadata-Bound Libraries

The following list reviews key guidelines and recommendations that help you minimize administrative effort in setting up and maintaining metadata-bound libraries.

- Use SAS (not host commands) for management of physical data. Using host commands doesn’t compromise security, but it can decrease clarity and create noise (warnings) in the audit logs. See “Security Impact of Moving Tables” on page 43.

  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 metadata Write access (the WriteMetadata or WriteMemberMetadata permissions) to only administrators that should be able to alter 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 Chapter 21 of *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 AUTHLIB procedure 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.
- 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 secured). You can use the REPORT statement to verify that this guideline is met.
- Ensure that all physical tables that are protected by a particular metadata-bound library remain within that library (directory). This standard, default state maximizes clarity and is essential 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.
- 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 Chapter 11, "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.

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 30.

Permission Requirements

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
Create a metadata-bound library (using the CREATE statement of the AUTHLIB procedure)	WriteMemberMetadata	-	-

Task	Secured Data Folder	Secured Library Object	Secured Table Object
Remove protections from a metadata-bound library (for example, using the REMOVE statement of the AUTHLIB procedure)	WriteMemberMetadata	WriteMetadata	-
Add tables to a metadata-bound library (for example, using the COPY procedure)	-	Create Table	Alter Table*

* Alter Table is required on any target table object that will be 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 create a metadata-bound library, the host identity must have host-layer control of the physical directory, 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.
  - 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).
- 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.

Note: The preceding tables address tasks that are performed using SAS. Tasks that are instead performed using host commands are not subject to metadata-layer permission requirements. See "Security Impact of Moving Tables" on page 43.

Note: 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.

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.

- 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.
- 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 (using the AUTHLIB MODIFY statement).
  - 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 (using the AUTHLIB MODIFY statement).
- 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.

See Also

"Security Impact of Moving Tables" on page 43

Auditing for Metadata-Bound Libraries

Which Events Can Be Logged?

For metadata-bound libraries, certain events are logged as part of 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 don't 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 a circumvention of security (for example, a user uses SAS to copy protected data to an unsecured location, updates that data, and then host-copies it back to the secured location). Only users who have Write access to the directory could do this. However, anyone who needs Create Table access to any secured table object within a library must have Write access to the corresponding directory.

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

- Chapter 9, "Administering Logging for SAS Servers," in *SAS Intelligence Platform: System Administration Guide*
- *SAS Logging: Configuration and Programming Reference*

Considerations for Data File Encryption

Encrypting Metadata-Bound Data

The process for encrypting metadata-bound tables is the same as for encrypting traditional tables, except that the Read password is obtained from the secured library object (in metadata). The password is not supplied in SAS code.

See "SAS Data File Encryption" in Chapter 34 of *SAS Language Reference: Concepts*.

Changing Encrypted Table Passwords

No Direct Changes

Nobody can directly change or remove the password on encrypted tables in a metadata-bound library.

Suggested Process

If you need to change the password on a metadata-bound library that contains encrypted tables, use a process such as the following:

1. Use SAS code to create a copy of the encrypted physical tables with the new password, in a separate directory. You can use the OVERRIDE option of the COPY procedure to set the new password.
2. Delete the original physical tables from the original library, using either a host delete command or the KILL option of the DATASETS procedure.
3. Change the password on the original library to the new value, using the MODIFY statement of the AUTHLIB procedure.
4. Use SAS to copy the physical tables back to the original library.
5. Delete the copy of the physical tables that you created in step 1.

Integrity Constraints

If there are referential integrity constraints in any of the tables, adjust the preceding process as follows:

- Specify CONSTRAINT=YES in step 1 and step 4.
- Either remove the constraints (delete the foreign keys) before performing step 2, or use a host delete command in step 2.

TIP If only a few of the tables are encrypted, and none of those tables have integrity constraints, consider using SELECT statements in steps 1 and 4 (instead of making adjustments as indicated in the preceding list).

Example

For example, the following code fragment performs steps 1 through 4 of the suggested process. In this example, X is the encrypted library, B is a table that has referential integrity constraints, and FKB is a foreign key within table B.

```
proc copy in=x out=emptydir override=(pw=secret2) constraint=yes;
run;

proc datasets library=x;
  modify b;
  ic delete fkb;
run;
quit;

proc datasets library=x kill;
run;

proc authlib library=x;
  modify pw=secret/secret2;
run;
quit;

proc copy in=emptydir out=x constraint=yes;
run;
```

To complete the process (step 5), the following code fragment deletes the copy of the physical tables from the WORK library.

```
proc datasets library=emptydir;
   modify b(pw=secret2);
   ic delete fkb;
run;
quit;

proc datasets library=emptydir kill pw=secret2;
run;
```

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, using the CREATE statement of the AUTHLIB procedure. The CREATE statement adds security information to the physical data and creates corresponding metadata objects at the same time.

You create traditional library and table objects by registering physical data, using a client such as 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:

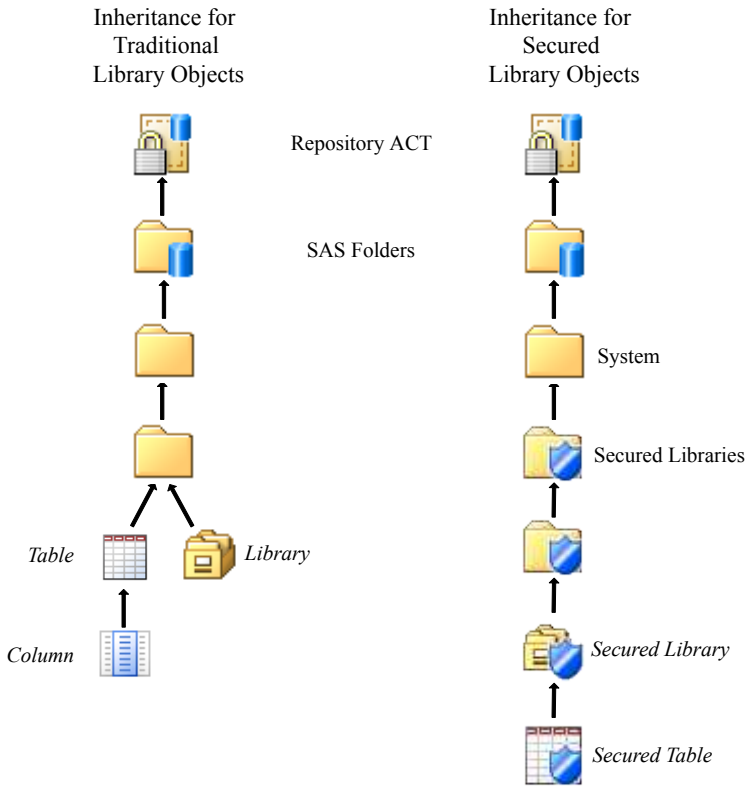
- Each secured library object inherits from its parent secured data folder.
- Each secured table object inherits from its parent secured library.

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.

  Note: 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 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.

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 Chapter 5 of *SAS Language Interfaces to Metadata*.

LIBNAME Statement Options

AUTHADMIN

An administrator specifies the AUTHADMIN=YES option on 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 on 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) on the LIBNAME statement to provide the metadata-bound library password (or passwords). See "AUTHPW (and Related Options)" on page 37.

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 "Who Uses the CREATE Statement?" 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 statement is a preproduction feature.

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

AUTHPW (and Related Options)

An administrator can choose to specify AUTHPW=*password-value* on 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 on 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 29

AUTHLIB Procedure

An administrator uses 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 36.
- 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 (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 Read Access to Metadata-Bound Data" on page 21.
- 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 28.

- 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 Chapter 24 of *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, which is a preproduction feature in the current release. 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 "(Preproduction) REPAIR Statement" on page 61.

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 Chapter 18, "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**.
- Restore the folder (and its contents) from a metadata server backup. See Chapter 12, "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 Chapter 18, "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.
- Restore the secured library object from a metadata server backup. See Chapter 12, "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 Chapter

18, “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.
- Restore the secured table object from a metadata server backup. See Chapter 12, “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, which is a preproduction feature in the current release. 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 “(Preproduction) REPAIR Statement” on page 61.

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.

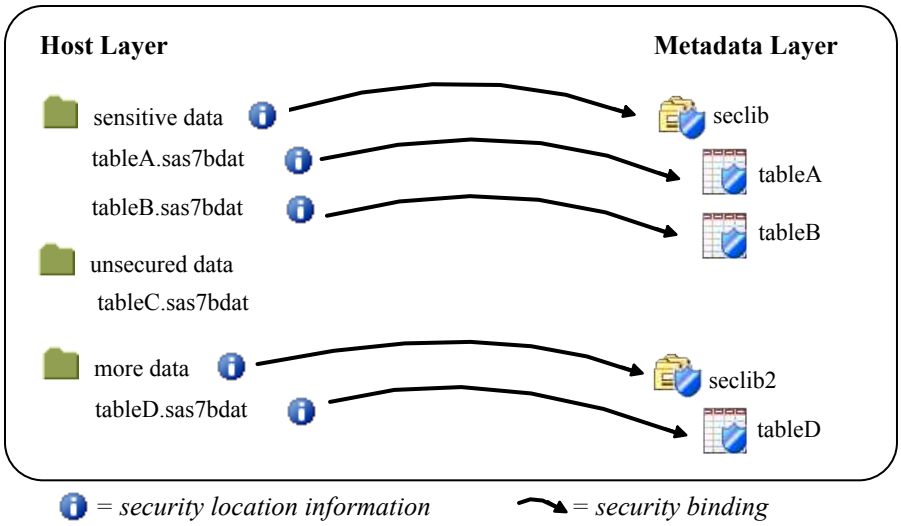
Introduction

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*

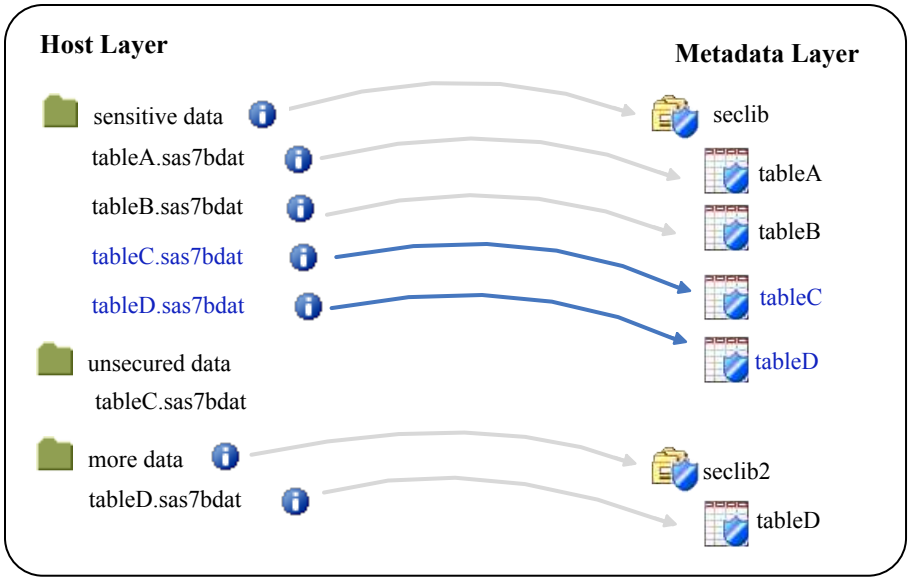


Using SAS

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*



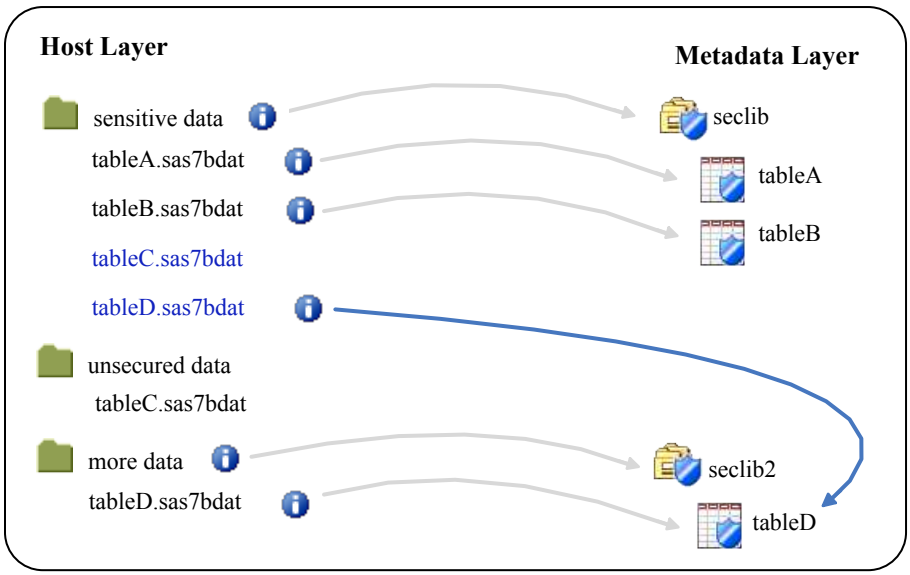
Notice that security information and bindings are generated for the added tables and that corresponding secured table objects are automatically created in metadata. With a SAS copy, both of the added tables are automatically secured by their parent library.

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

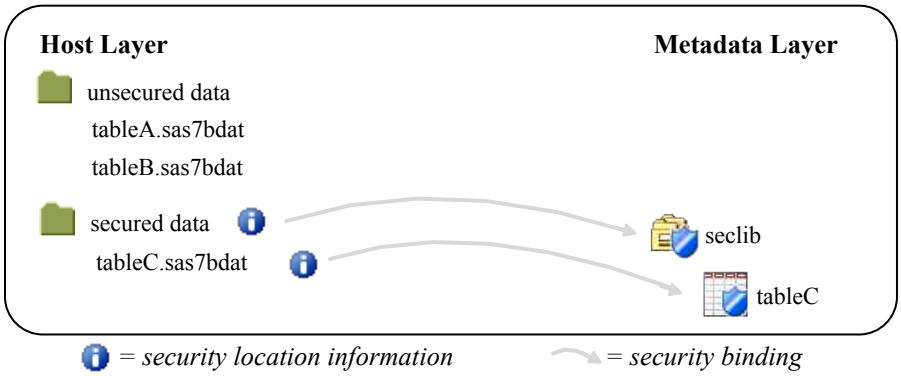
The examples in this topic use the copy action. The same results occur if tables are moved, except that the original physical tables are deleted.

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*

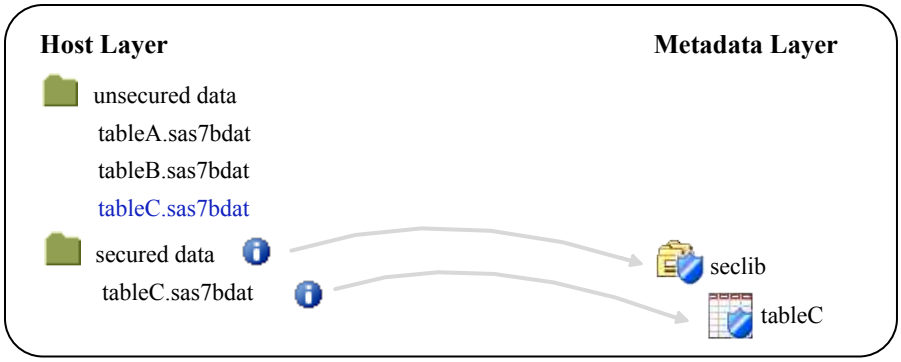


Using SAS

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 25.

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

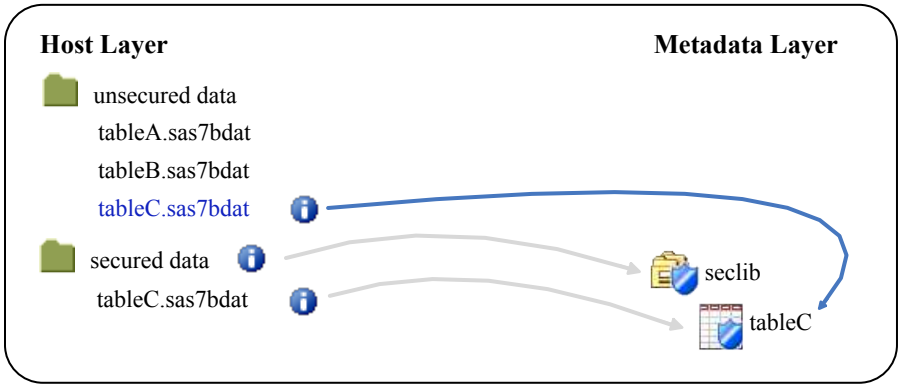
Figure A1.5 *Example: After a SAS Copy*



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 values for a metadata-bound library
- repair metadata-bound libraries by recovering security information, secured library objects, and secured table objects (this functionality is preproduction in this release)
- 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 SAS 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

What Is a 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. For more information, see *SAS Guide to Metadata-Bound Libraries*.

What Are Metadata-Bound Library Passwords?

A metadata-bound library has a single set of passwords stored in the secured library object, which are 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, but rather to authorize administrator access to repair the binding of physical data to the secured library or table metadata objects. They are also validated in the process of authorizing a user's access to a data set but 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, rather than specifying different password values using READ=, WRITE=, and ALTER= options. 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, 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 as you must specify them to unbind the library, modify the passwords, or repair any inconsistencies in the binding information between what is recorded in the physical files and the actual metadata objects.

CAUTION:

If you lose the password (or passwords) for a metadata-bound library, 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.

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

It is possible to have physical data sets in a metadata-bound library that do not have the metadata-bound library passwords. This can occur if the data sets existed with passwords that differ from the metadata library passwords when the library was bound. See "Example 2: Binding a Physical Library That Contains Password-Protected Data Sets" on page 69. It can also occur if data sets with different passwords are copied into the library by an operating system copy utility. These data sets are not considered to be part of the bound library for authorization purposes. If the data set was in an operating system copied from another metadata-bound library, the data set is still protected by the permissions users have in the secured table object to which it is bound in the original secured library. If the data set is not copied from a metadata-bound library, then metadata permissions do not apply, and you must supply the appropriate passwords to access the data. You can use the MODIFY statement of PROC AUTHLIB to modify the passwords to those of the metadata-bound library so that it will be bound to a secured table object in the secured library object to which the library is bound. See "Example 5: Changing Passwords on Data Sets " on page 73.

Requirements to Use PROC AUTHLIB Statements

Except for the REPORT statement, all statements within PROC AUTHLIB 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 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.

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.

This 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.

This procedure does not support libraries accessed through a SAS/SHARE Server.

Requirement: A connection to the target metadata server.

See: For more information, see *SAS Guide to Metadata-Bound Libraries*.

PROC AUTHLIB <*option-1* <*...option-n*>>;

CREATE

SECUREDLIBRARY='*secured-library-name*'
<SECUREDFOLDER='*secured-folder-path*'>
<LIBRARY=*libref*>
PW=*all-password-value* | |
ALTER=*alter-password-value*
READ=*read-password-value*
WRITE=*write-password-value*;

MODIFY

<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
<TABLESONLY=YES|NO>;

REMOVE

<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*;

REPAIR ADD | UPDATE | DELETE

LOCATION | METADATA

SECUREDLIBRARY='*secured-library-name*'
SECUREDFOLDER='*secured-folder-path*'
<SECUREDLIBRARYGUID='*secured-library-guid*'>
<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
<TABLESONLY=YES | NO>;

REPORT

<LIBRARY=*libref*>;

TABLES *SAS-file-1* | _ALL_ | _NONE_

<*SAS-file-n*>
<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
< SECUREDTABLEGUID='*secured-table-guid*'>;

Statement	Task	Example
"PROC AUTHLIB Statement"	Create and manage metadata-bound libraries	

Statement	Task	Example
"CREATE Statement"	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
"MODIFY Statement"	Modify password values for a metadata-bound library	Ex. 5, Ex. 6
"REMOVE Statement"	Remove the physical security information and metadata objects that protect a metadata-bound library	Ex. 7
"(Preproduction) REPAIR Statement"	Recover security information (in physical data) or secured library and table objects (in metadata)	
"REPORT Statement"	For a specified metadata-bound library, compare physical library contents with corresponding metadata objects (in order to identify any inconsistencies).	Ex. 8
"TABLES Statement"	Specify which tables within a specified metadata-bound library are affected by certain AUTHLIB statements	Ex. 4, Ex. 9

PROC AUTHLIB Statement

Manages metadata-bound libraries.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

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, 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 or temporary library 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 an invalid password is entered when required.

NO
prevents a dialog box from appearing. If a missing or invalid password is entered, 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 to metadata by generating corresponding metadata objects in the SAS Metadata Repository and creating a record of the metadata objects in the physical directory.

Requirement: A connection to the target metadata server. For more requirements, see "Requirements to Use PROC AUTHLIB Statements" on page 51.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

CREATE

SECUREDLIBRARY='*secured-library-name*'
<SECUREDFOLDER='*secured-folder-path*'>
<LIBRARY=*libref*>
PW=*all-password-value* | |
ALTER=*alter-password-value*
READ=*read-password-value*
WRITE=*write-password-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*
sets a single password for a metadata-bound library.

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

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

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

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 will be created.

If the SECUREDFOLDER= option is not specified, 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 (/), it is a relative path and the value will be appended to **`/System/Secured Libraries/`**

to find the folder. If the SECUREDFOLDER= option begins with a slash (/), 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.

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, the physical library from the AUTHLIB procedure is used.

Alias: LIB=, DDNAME=, DD=

Restriction: The physical library specified cannot be a concatenated library or temporary library and must be processed by an engine that supports metadata-bound libraries.

Details

Specifying Passwords

If your physical library does not contain password-protected data sets, 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. See "Example 1: Binding a Physical Library That Contains Unprotected Data Sets" on page 68.

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 (/). See "Example 3: Securing a Library When Existing Data Sets Are Protected with the Same Passwords" on page 70.

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: Securing a Library When Existing Data Sets Are Protected with Different Passwords" on page 71) 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 73).

MODIFY Statement

Modifies password values for a metadata-bound library.

Requirement: A connection to the target metadata server. For more requirements, see "Requirements to Use PROC AUTHLIB Statements" on page 51.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

MODIFY

<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
<TABLESONLY=YES | NO>;

Required Arguments

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

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

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

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

Optional Arguments

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, the physical library from the AUTHLIB procedure is used.

Alias: LIB=, DDNAME=, DD=

Restriction: The physical library specified cannot be a concatenated library or temporary library and must be processed by an engine that supports metadata-bound libraries.

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

Default: NO

Details

Using the MODIFY Statement

The MODIFY statement can modify the value of the required metadata-bound library passwords. 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 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, specify the current and new values for the metadata-bound library passwords separated by a **/** in the MODIFY statement. See "Example 6: Changing Metadata-Bound Library Passwords" on page 74.

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 on separate TABLES statements. See "Example 5: Changing Passwords on Data Sets " on page 73.

Using the LIBRARY= Option

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

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

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

```
AUTHLIB 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.

REMOVE Statement

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

Requirement: A connection to the target metadata server. For more requirements, see "Requirements to Use PROC AUTHLIB Statements" on page 51.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

REMOVE

<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*;

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.

Optional Argument

LIBRARY=*libref*
name of the physical library where the metadata-bound library is created and the security information is stored.

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

Alias: LIB=, DDNAME=, DD=

Restriction: The physical library specified cannot be a concatenated library or temporary library and must be processed by an engine that supports metadata-bound libraries.

Details

The REMOVE statement is used to unbind the metadata-bound library feature from a SAS data 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 `/` after the current password but not specifying a new password value. If you choose to do this, you are warned in the SAS log that the data sets no longer have any SAS protection.

The REMOVE statement will not unbind any data sets that are currently bound to a secured table object in a different secured library than the one to which this physical library is bound.

Note: Ensure that all physical tables that are protected by a particular metadata-bound library remain within that library (directory). This standard, default state maximizes clarity and is essential 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.

(Preproduction) REPAIR Statement

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

Requirement: A connection to the target metadata server. For more requirements, see “Requirements to Use PROC AUTHLIB Statements” on page 51.

Note: The REPAIR statement is a preproduction feature. For more information, see “Details” on page 61.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

REPAIR ADD | UPDATE | DELETE

LOCATION | METADATA

SECUREDLIBRARY='*secured-library-name*'
SECUREDFOLDER='*secured-folder-path*'
<SECUREDLIBRARYGUID='*secured-library-guid*'>
<LIBRARY=*libref*>
PW=*all-password* | |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
<TABLESONLY=YES | NO>;

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

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

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

Alias: LIB=, DDNAME=, DD=

Restriction: The physical library specified cannot be a concatenated library or temporary library 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 will be 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.

SECUREDLIBRARYGUID=*secured-library-guid*
external identity assigned to the secured library object and stored as part of the security information in the physical library.

Note: The secured library GUID can be found in the REPORT statement output. The GUID that is stored as an external ID in the metadata is not reported, but it matches when correctly configured.

TABLESONLY=YES | NO
specifies whether the REPAIR statement action is applied at the library level or just to the tables. If TABLESONLY=NO, the action is applied to the library and the tables. If TABLESONLY=YES, 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

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.

The REPAIR statement is a preproduction feature, which means it 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.

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, 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 "c:\lib2";

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

libname abcde "c:\mylib"
   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
```

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.

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:

If you lose the password (or passwords) for a metadata-bound library, 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.

REPORT Statement

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

Requirement: A connection to the target metadata server. For more requirements, see "Requirements for Using the REPORT Statement" on page 64.

Syntax

PROC AUTHLIB <*option-1* <*...option-n*>>;

REPORT

<LIBRARY=*libref*>;

Optional Argument

LIBRARY=*libref*

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

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

Alias: LIB=, DDNAME=, DD=

Restriction: The physical library specified cannot be a concatenated library or temporary library and must be processed by an engine that supports metadata-bound libraries.

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, 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, 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, REPAIR, and REPORT statement to specify the tables to process a statement action and to specify the current passwords on the data sets, if different from the metadata-bound library passwords.

Default: When no TABLES statement is specified, the TABLES _ALL_ statement is the default behavior.

Requirement:
- The TABLES statement can follow only a CREATE, MODIFY, REPAIR, or REPORT statement.
- A connection to the target metadata server.

Syntax

TABLES *SAS-dataset-1 <SAS-dataset-n>* | _ALL_ | _NONE_
</>
<PW=*all-password*> | |
<ALTER=*alter-password*>
<READ=*read-password*>
<WRITE=*write-password*>
<MEMTYPE= DATA || VIEW>
< SECUREDTABLEGUID='*secured-table-guid*'>;

Optional Arguments

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

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

PW=*all-password*
specifies the current password of the data set.

ALTER=*alter-password*
specifies the current ALTER= password of the data set.

READ=*read-password*
specifies the current READ= password of the data set.

WRITE=*write-password*
specifies the current WRITE= password of the data set.

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

DATA
specifies SAS data file member type.

VIEW
specifies SAS view member type.

Alias: MTYPE=, MT=

Default: ALL

SECUREDTABLEGUID=*secured-table-guid*
the external identity of the secured table object that was assigned when the object was created and that is stored as part of the location information in physical data sets that are bound to the secured table object.

Restriction: SECUREDTABLEGUID= option is used only in TABLES statements that follow certain REPAIR statements.

Details

Using the TABLES Statement

CAUTION:

If you lose the password (or passwords) for a metadata-bound library, 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.

The TABLES statement is primarily used to specify the current password(s) on data sets when different from the current metadata-bound library required password(s). It usually follows a CREATE or MODIFY statement to make the data set passwords change to the metadata-bound library passwords. See "Example 4: Securing a Library When Existing Data Sets Are Protected with Different Passwords" on page 71.

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 to some other value. You are not restricted to changing to a common metadata-bound library password. In that case, you might choose to specify both a current and new password separated by a slash / for data sets in a TABLES statement.

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 password options to apply to 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 for data sets when different from the metadata-bound library passwords. If the TABLES statement is not used, only two groups of data sets will be bound:

- data sets without passwords
- data sets with passwords 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 55.

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 in the data sets. If no TABLES statement follows the MODIFY statement, 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. For more information, see "MODIFY Statement" on page 56.

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. For more information, see "(Preproduction) REPAIR Statement" on page 59.

Using the TABLES Statement with the REMOVE Statement

Using a TABLES statement with a REMOVE statement that does not select all tables will produce an ERROR and not execute.

Using the TABLES Statement with the REPORT Statement

The TABLES statement is syntactically accepted with the REPORT statement but has little or no use.

Results: AUTHLIB Procedure

The REPORT statement produces the following output.

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: FD72FF6F-8989-4716-B19A-A3F2B0BD61F5
Registered in OS Path: c:\abcde
Password Set: 0

Obs	MemberName	MemberType	SecuredTableName	SecuredTableGUID
1	PRODUCT	DATA	PRODUCT.DATA	BF2730AE-C9B7-4473-8CBA-D9246A28FB4B

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: FD72FF6F-8989-4716-B19A-A3F2B0BD61F5
Registered in OS Path: c:\abcde
Password Set: 1

Obs	MemberName	MemberType	SecuredTableName	SecuredTableGUID
2	EMPINFO	DATA		
3	EMPLOYEES	DATA		

Examples: AUTHLIB Procedure

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

Features: CREATE Statement and Options
SECUREDLIBRARY=
SECUREDFOLDER=
PW=

Details

Here is an example of binding a physical library that contains data sets that do not have passwords. Library ZYXWVUT contains three data sets — EMPLOYEES, EMPINFO, and PRODUCT — that do not have passwords. The library and data sets are bound with the password **secretpw**. The binding is straightforward, as PROC AUTHLIB has unhindered access to the data.

Program

```
proc authlib lib=zyxwvut;
  create securedfolder="Department XYZZY"
    securedlibrary="ZYXWVUTEmps"
    pw=secretpw;

run;
quit;
```

Log

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;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.68 seconds
      cpu time            0.03 seconds
```

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

Features: CREATE Statement and Options
SECUREDLIBRARY=
SECUREDFOLDER=
PW=

Details

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 of **`abcd`**, a Write password of **`efgh`**, and an Alter password of **`ijkl`** before the library is secured by the statements in the last example. The third data set, PRODUCT, is not protected with passwords.

Program

```
/*  To secure a library that has password-protected data sets  */
/*  that all share the same alter password ijkl with */
/* PROC AUTHLIB CREATE, submit: */
```

```
proc authlib lib=abcde;
 create securedfolder="Department XYZZY"
     securedlibrary="ABCDEEmps"
     pw=secretpw;
run;
quit;
```

Log

The library was bound and the unprotected data set password was set. The protected data sets' passwords did not change because their current passwords were not specified.

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.
NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.14 seconds
      cpu time            0.09 seconds
```

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

Features: CREATE Statement and Options
SECUREDLIBRARY=
SECUREDFOLDER=
PW=

Details

This example shows how you could have modified the passwords for the EMPLOYEES and EMPINFO data sets from the preceding example in the PROC AUTHLIB CREATE statement. The EMPLOYEES and EMPINFO data sets are protected with the same

passwords. The Alter password for the data sets,**ijkl**, is specified in the PW= argument before the new password, separated by a slash (/).

Program

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

Log

Log A2.4 *Securing a Library with Data Sets That Are Protected with the Same Passwords*

```
99   proc authlib lib=abcde;
100   create securedlibrary="ABCDEEmps"
101        securedfolder="Department XYZZY" pw=XXXX/XXXXXXXX;
102  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: 87165DCD-3C60-4C7D-BD53-903AAC68CCC7
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 "EMPLOYEES.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEES.DATA.
NOTE: The passwords on ABCDE.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/ABCDEEmps" for data set ABCDE.PRODUCT.DATA.
NOTE: The passwords on ABCDE.PRODUCT.DATA were successfully modified.
103  quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.38 seconds
      cpu time            0.09 seconds
```

Example 4: Securing a Library When Existing Data Sets Are Protected with Different Passwords

Features: CREATE Statement and Options
SECUREDLIBRARY=
SECUREDFOLDER=
PW=
ALTER=
READ=
WRITE=
TABLES

Details

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 of **abcd**, a Write password of **efgh**, and an Alter password of **ijkl**. The PRODUCT data set is not protected. Because these data sets have different passwords, to change these passwords, you must use TABLES statements with the CREATE statement. When the TABLES statement is used, a TABLES statement must be specified for all tables. This example also uses three passwords to bind the library: READ=ABCDEFGH, WRITE=IJKLMNO, and ALTER=PQRSTUVW.

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;
```

Log

Log A2.5 *Securing a Library with Existing Data Sets That Are Protected with Different Passwords*

```
187  proc authlib lib=klmno;
188
189   create securedlibrary="KLMNOEmps"
190
191        securedfolder="Department XYZZY"
192        read=XXXXXXXX write=XXXXXXX alter=XXXXXXXX;
193
194  tables employees / pw=XXXX;
195  tables empinfo / read=XXXX write=XXXX alter=XXXX;
196  tables product;
197
198  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: ABA4F5FB-F30A-4F20-9C0B-9F05E877B7BD
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.
199  quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.17 seconds
      cpu time            0.04 seconds
```

Example 5: Changing Passwords on Data Sets

Features: MODIFY Statement and Options
- LIBRARY=
- PW=
- TABLESONLY=

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 69 to match the metadata-bound library password. 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. The existing data sets' Alter password is specified in the PW= argument before the metadata-bound password, separated by a slash (/). The TABLESONLY statement specifies to modify table passwords only.

Program

```
proc authlib lib=abcde;
   modify tablesonly=yes pw=ijkl/secretpw;
run;
quit;
```

Log

Log A2.6 *Changing Data Set Passwords*

```
200  proc authlib lib=abcde;
201   modify tablesonly=yes pw=XXXX/XXXXXXXX;
202    run;

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 "EMPLOYEES.DATA" to the secured library object at
      path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEES.DATA.
NOTE: The passwords on ABCDE.EMPLOYEES.DATA were successfully modified.
NOTE: The passwords on ABCDE.PRODUCT.DATA do not require modification.
203  quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.07 seconds
      cpu time            0.01 seconds
```

Example 6: Changing Metadata-Bound Library Passwords

Features: MODIFY Statement and Options
SECUREDLIBRARY=
SECUREDFOLDER=
PW=

Details

If you believe that the metadata-bound library passwords have been compromised, use the MODIFY statement to modify the library passwords. The following code changes the library passwords and the data set passwords of all data sets in the library that contain the specified passwords.

Program

```
proc authlib lib=abcde;
   modify securedlibrary="ABCDEEmps"
      securedfolder="Department XYZZY"
      pw=secretpw/newpassd;
run;
quit;
```

Log

The preceding code changed the library passwords and the data set passwords of all data sets in the library with those passwords. An error message is displayed for any data set with different passwords.

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;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.09 seconds
      cpu time            0.03 seconds
```

Example 7: Using the REMOVE Statement

Features: REMOVE Statement and Options
LIBRARY=
PW=

Details

This example shows how to unbind a metadata-bound library. The code deletes metadata that describes the library and its tables from the SAS Metadata Repository, removes security bindings from the physical library and data sets, and removes the assigned password from the data sets, leaving them unprotected. The slash (/) after the password is optional and used to remove or replace the password from the data sets. Note that if a library is bound with Read, Write, and Alter passwords, as in Example 4 on page 71, you must specify all of the passwords, and they must each have a /.

Program

```
proc authlib lib=abcde;
   remove pw=newpassd/;
run;
quit;
```

Log

Log A2.8 *Unbinding a Metadata-Bound Library*

```
25   proc authlib lib=abcde;
26    remove pw=XXXXXXXX/;
27   run;

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.EMPLOYEES.DATA were removed along with the secured
         library object location, leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.EMPLOYEES.DATA was successfully removed.
WARNING: Some or all the passwords on ABCDE.PRODUCT.DATA were removed along with the secured library
         object location, leaving the data set unprotected.
NOTE: The secured table object location for ABCDE.PRODUCT.DATA was successfully removed.
NOTE: Successfully deleted the secured library object that was located at:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 99F963DC-CD45-4704-96C7-DB9355B65857
NOTE: Successfully deleted the recorded location of the secured library object for the physical
      library ABCDE.
28   quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.75 seconds
      cpu time            0.26 seconds
```

Example 8: Using the REPORT Statement

Features: Report Statement
LIBRARY=

Details

The following code checks a library's bindings.

Program

```
proc authlib lib=abcde;
  report;
run;
quit;
```

Log

Log A2.9 *Creating a Report*

```
49   proc authlib lib=abcde;
50    report;
51   run;

52   quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.37 seconds
      cpu time            0.21 seconds
```

Results

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: 87165DCD-3C60-4C7D-BD53-903AAC68CCC7
Registered in OS Path: c:\abcde
Password Set: 0

Obs	MemberName	MemberType	SecuredTableName	SecuredTableGUID
1	EMPINFO	DATA	EMPINFO.DATA	7C07DF02-D4A5-4443-BDB3-5D0DCE0D83AA
2	EMPLOYEES	DATA	EMPLOYEES.DATA	F0296103-88A2-4735-B898-D35D0D271E7F
3	PRODUCT	DATA	PRODUCT.DATA	8F2D3F85-3797-42D2-9DF3-71334BC222EB

Example 9: Using the TABLES Statement

Features: TABLE Statements and Options
- PW=
- ALTER=
- READ=
- WRITE=

Details

Library KLMNO has three data sets in the TABLES statements: EMPLOYEES, EMPINFO, and PRODUCT.

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;
```

Log

Log A2.10 *TABLES Statement*

```
187  proc authlib lib=klmno;
188
189   create securedlibrary="KLMNOEmps"
190
191        securedfolder="Department XYZZY"
192        read=XXXXXXXX write=XXXXXXX alter=XXXXXXXX;
193
194  tables employees / pw=XXXX;
195  tables empinfo / read=XXXX write=XXXX alter=XXXX;
196  tables product;
197
198  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: ABA4F5FB-F30A-4F20-9C0B-9F05E877B7BD
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.
199  quit;

NOTE: PROCEDURE AUTHLIB used (Total process time):
      real time           0.17 seconds
      cpu time            0.04 seconds
```

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 act or process of converting data to a form that is unintelligible except to the intended recipients.

identity
See metadata identity

library reference
See libref

libref
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
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. A SAS Metadata Server is an example.

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
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
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
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
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
another term for 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