SAS2181-2018

A Programming Approach to Implementing SAS® Metadata-Bound Libraries for SAS® Data Set Encryption

Deepali Rai, SAS Institute Inc.

ABSTRACT

SAS® metadata-bound libraries provide a robust means of protecting SAS data without compromising on accessibility of the secured data. Although seamless access to metadata-bound data controlled through metadata authorizations is an advantage of using metadata-bound libraries, their implementation and maintenance can potentially be a cumbersome task for administrators. Administrators face complexities in maintaining and managing metadata-bound libraries if multiple users create and own SAS data. Proper planning is required to handle complexities around heterogeneous directory ownerships, password and encryption key management, and monitoring of unencrypted SAS data. This paper explains those complexities and provides best practice recommendations for both administrators and SAS programmers working with SAS data. This paper provides code that can be used to empower SAS programmers to define metadata-bound libraries through a controlled process. Code is also provided for administrators to monitor and report on unencrypted SAS data. In addition, the paper explains the nuances of identity resolution and effective authorizations when accessing metadata-bound data through SAS/SHARE® libraries. It then provides a step-by-step approach for creating and managing the binding of a SAS library that is accessed through a SAS/SHARE server.

INTRODUCTION

SAS metadata-bound libraries are used within an organization to enforce enhanced security for SAS data. The two primary use cases for SAS metadata-bound libraries are as follows:

- Authorization: A metadata-bound library binds a physical library and its tables to metadata such that metadata layer authorizations are always honored regardless of how the SAS data is being accessed.
- Encryption: SAS encryption for data at rest can be enforced by specifying encryption parameters in a metadata-bound library definition.

And while end-user access to metadata-bound data is seamless, administrators have to continually work with users that create and own SAS data to ensure that all data is encrypted and secured using metadata-bound libraries. Some of the requirements for creating metadata-bound libraries render their administration and management an ongoing activity. For example:

- Each directory that contains SAS data needs to be individually bound to metadata. New directories and even subdirectories under an existing metadata-bound directory are not automatically bound to metadata.
- For UNIX based file systems, only the user that owns the directory can bind it to metadata, and therefore should be the one creating the metadata-bound library.
- Users have to specify password and encryption values at the time of creation for each metadata-bound library and should have knowledge of the same values when updating the secured library.

The purpose of this paper is to provide building blocks for a process that allows administrators to share the responsibility of data security with data creators, and enables data creators to set up and efficiently manage metadata-bound libraries for their respective directories. By using the recommended process, administrators can enforce pre-defined passwords and encryption attributes for new metadata-bound libraries. Administrators can also monitor all SAS data and have any new SAS data promptly bound to metadata by notifying the data owners.

This paper provides best practice guidelines and code that administrators can use for the metadata-bound library implementation in their SAS environments. The paper is structured to first provide high-level information about metadata-bound libraries and then a more in-depth explanation of how to develop a custom solution for a metadata-bound library implementation. Depending on site-specific requirements, administrators can build a comprehensive solution using the information provided in this paper.

Even with the amount of planning and process management required for a successful implementation of metadata-bound libraries, there are benefits to adopting this solution. Organizations that plan to set up a sophisticated metadata security model can greatly strengthen their SAS data security by using metadata-bound libraries. This paper does not provide guidelines for designing a metadata security model, although it is recommended that a security model is implemented to support metadata-bound libraries.

With environments using a SAS/SHARE server, administrators need to ensure that the SAS/SHARE server is started with the correct options and also take into account various scenarios for identity resolution.

It is assumed that administrators trying to benefit from the information provided in this paper are proficient SAS programmers with a sound understanding of SAS metadata and a basic understanding of SAS metadata-bound libraries.

METADATA-BOUND LIBRARY CONCEPTS

A metadata-bound library is a binding of a physical SAS library to a metadata object such that access to the physical library and data is always controlled through metadata layer authorizations. The binding information for a physical library is stored in a subdirectory named .sassl, in a file named loc. The binding information for a physical SAS data set is located in the data set header. The metadata objects to which a physical library and data are bound are called secured library and secured table objects, respectively.

Access to a physical library and SAS data can originate from a SAS application or directly from a LIBNAME statement. Since the binding information for the secured library and tables is stored at the physical layer, all access to physical data honors metadata-layer authorizations. Metadata-bound libraries also support encryption for data at rest using SAS Proprietary and AES encryption algorithms. By requiring encryption on a metadata-bound library, all secured data within the library are encrypted with the specified encryption algorithm and encryption key.

Enforcing metadata authorizations through SAS metadata-bound libraries provides an additional layer of security. The combined effect of file system and metadata layer permissions can be used to implement a more comprehensive and flexible security model.

Administrators can also choose to encrypt data at rest using an operating system or file system level solution that encrypts all file types, including SAS data, Microsoft Excel files, .CSV files, text files, and so on. Metadata-bound libraries only implement encryption for SAS data sets and SAS data set views.

DEPICTION

The following figure depicts a physical library before and after it has been bound to metadata. The binding of the physical library and tables creates corresponding secured library and secured table objects in metadata, as shown in the figure. The security information for a physical library is stored in a subdirectory named .sassl and a file named loc. The security information for a physical table is located in the physical table header.

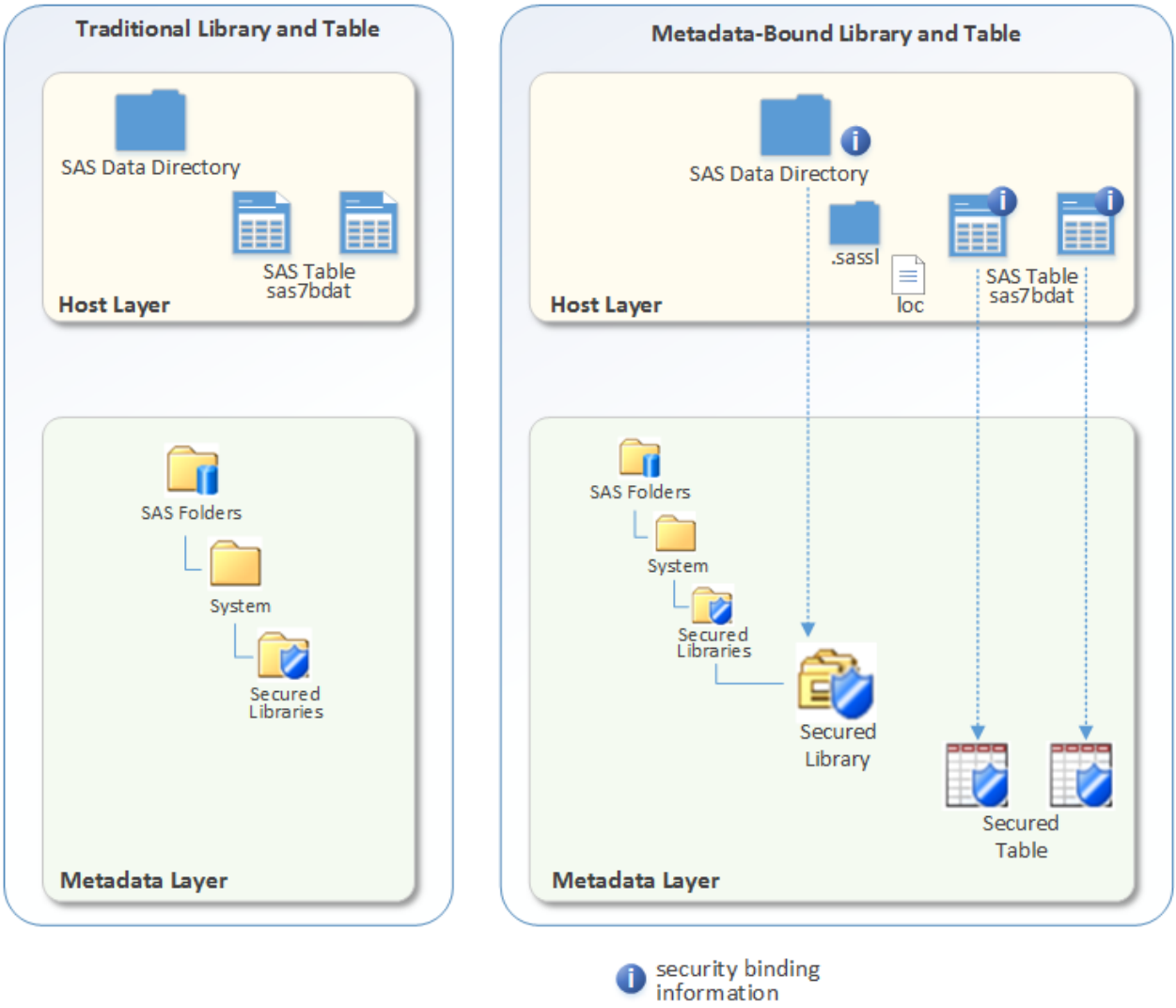

Figure 1. Depiction of a Metadata-Bound Library

AUTHORIZATION MODEL

The following diagram is a simplified depiction of the authorization model for a metadata-bound table.

There are a few different ways users can access physical SAS data, including direct host layer access, access through LIBNAME statements, or SAS applications such as SAS® Enterprise Guide.

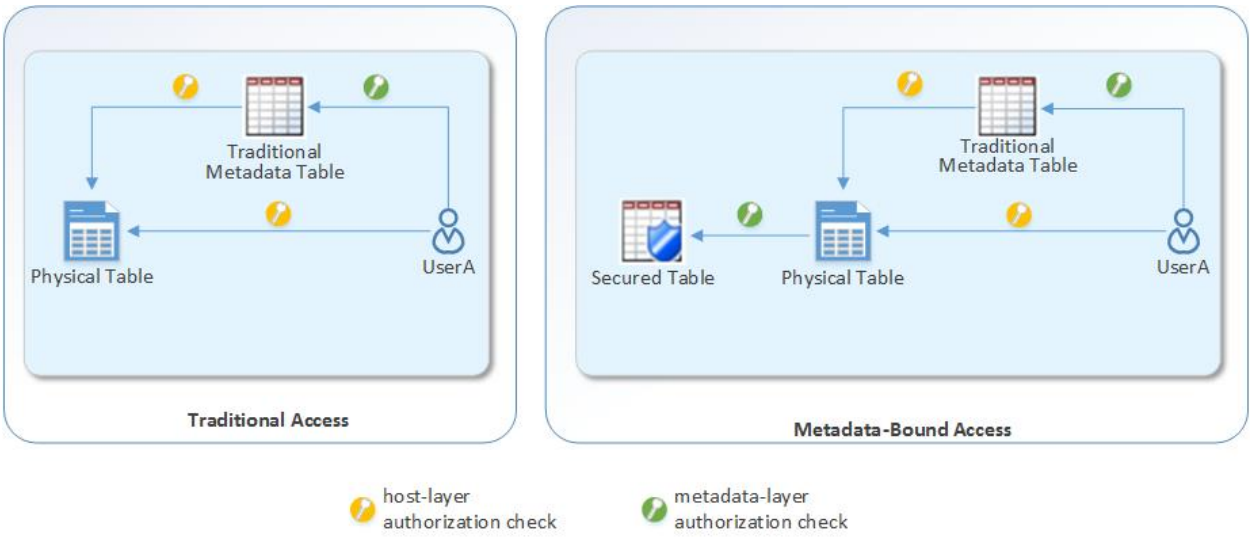

Figure 2. Authorization Checks for Traditional Versus Metadata-Bound Table on Type of Access

With traditional access, users can submit a LIBNAME statement in code or use a traditional SAS metadata table to access the physical data. In this scenario, the option exists for users to bypass the metadata-layer authorizations to directly access the physical data.

With metadata-bound access, whether users submit a LIBNAME statement or use a traditional SAS metadata table to access the physical data, the metadata layer permissions and host layer permissions are both always honored. Therefore, a user accessing a metadata-bound table is always subject to metadata authorization checks before the data can be accessed.

METADATA LAYER ACCESS PERMISSIONS

SAS metadata enforces general purpose permissions on all metadata objects and additional special permissions are enforced on certain types of metadata objects. In order to work with secured library and table objects, a combination of these general and special permissions are required.

General Purpose Permissions

ReadMetadata and WriteMetadata are general purpose permissions that apply to all metadata objects. In addition, the WriteMemberMetadata permission can be used on metadata folders to control who can create new objects within a metadata folder. These permissions must be configured to control how users interact with secured library and table objects. Users require ReadMetadata permission on all relevant metadata folders, secured libraries, and secured tables in order to be able to access metadata-bound data. Users need either WriteMetadata or WriteMemberMetadata permission on the target metadata folder under */System/Secured Libraries* to create and update metadata-bound libraries. Detailed information on SAS metadata permissions can be found in the SAS 9.4 Intelligence Platform: Security Administration Guide.

Special Permissions for Secured Objects

In addition to the general purpose permissions, special permissions for secured libraries and secured table objects are enforced by SAS. These permissions control how users can interact with the secured table data. These permissions should be configured by administrators to achieve the desired level of security. The special permissions for secured objects are: Select, Insert, Update, Delete, Create Table, Drop Table, and Alter Table. At a minimum, users must have the Select permission on a secured table in order to be able to view data.

PLANNING

The planning phase should be used to discuss the need for implementing metadata-bound libraries and configuring the environment in preparation for a metadata-bound library implementation. Given an organization's preferences around security configurations, there can be different scenarios for which a decision for using metadata-bound libraries can be made. The following scenarios can be used to understand when using metadata-bound libraries might be beneficial:

1. Administrators might choose to implement a sophisticated metadata security model when the same cannot be achieved using file system permissions. This situation could be a result of restrictive policies or a reluctance towards managing a large number of security groups. In this scenario, metadata-bound libraries allow administrators to implement data access using metadata groups. A user always has to be authenticated and authorized in metadata in order to access SAS data.
2. Metadata-bound libraries can be used to implement additional authorizations, such as row-level and column-level access for increased security of SAS data. With this type of a security model, since authorizations are enforced by both the file system and metadata layers, administrators can create very restricted and complex security implementations to achieve the desired level of security.
3. Metadata-bound libraries provide an effective and strong encryption solution for encrypting SAS data. With proper planning, administrators can implement a systematic process that encrypts all SAS data being stored in the server file system.
4. With metadata-bound libraries, users can seamlessly encrypt all new data sets without having to code encrypt statements in their SAS programs.
5. Operating system-level or file system-level encryption solutions should be used in situations where all file system content needs to be encrypted. Since a metadata-bound library only encrypts SAS data, its implementation cannot meet the requirement of encrypting all content such as Excel files, text files, CSVs, and so on.
6. Implementation of metadata-bound libraries becomes unnecessary if the situation does not demand very restricted access controls to be placed on SAS data and the security requirements are sufficiently satisfied using file system-level permissions.

After a decision has been made to utilize metadata-bound libraries for authorization, and optionally encryption of SAS data, administrators should identify the factors that affect how the implementation process needs to be defined. The following factors are important to consider before designing and executing the process.

IMPLEMENTATION CONSIDERATIONS

1. Only BASE SAS data sets and SAS data set views can be bound to metadata.
2. Concatenated libraries cannot be bound to metadata.
3. Binding of a physical library is limited to the directory and the data that is referenced by the physical library, and does not automatically include any sub-directories. Each physical library, regardless of its directory nesting, should be individually bound to metadata.
4. Host commands can be used to delete and replace files within a directory, independent of any metadata binding.
5. If a physical table has been deleted or replaced using host commands, the corresponding metadata table object has to be manually deleted by an administrator using SAS® Management Console.
6. In a UNIX environment, only the account that owns the directory can bind the directory to metadata.
7. In a Microsoft Windows environment, only the accounts that have Full Access permission on the directory can bind the directory to metadata.
8. The user that makes the bind request must have ReadMetadata and WriteMetadata (or WriteMemberMetadata) permission granted on the target metadata folder under */Systems/Secured*

Libraries.

9. It is recommended that all encrypted tables within a metadata-bound library use the same encryption key.
10. Enough physical disk space should be available since a copy-in-place approach is used in the following cases:
    - when a physical library is registered as an encrypted metadata-bound library. All tables in the library are copied in place. The new tables are encrypted and bound to metadata.
    - when encryption information, such as the encryption algorithm or the encryption key, is modified for a metadata-bound library.
    - when binding a host-copied table to an encrypted metadata-bound library.

IDENTIFY USERS

Administrators, developers, analysts, and consumers are different types of user personas who might interact, directly or indirectly, with SAS data. A user has read-only or read-write access depending on their functional persona. Administrators should identify users who create or update SAS data as part of their role within the organization. These users might also have privileges to create new directories on the file system for organizing their files. Administrators will work with these users to help implement and maintain a successful metadata-bound library implementation. We refer to these users as data creators. Data creators are responsible for binding all SAS data directories that they own, and continually bind any new SAS data directories that they create.

Since users with read-only privileges to SAS data have seamless access to metadata-bound data, they do not need to be involved in the implementation process.

IDENTIFY SAS DATA DIRECTORIES

Metadata-bound libraries secure and encrypt only SAS data and views. SAS data could be located in server directories, Windows network drives, or Linux file system mounts. Administrators should identify all locations where SAS data is stored or could be stored in the future. The identified SAS data locations need to be regularly monitored for maintaining security and encryption.

DESIGN AND IMPLEMENT A METADATA SECURITY MODEL

A well designed metadata security model is of the utmost importance when using metadata-bound libraries for securing SAS data. A metadata security model implementation sets up users in metadata with appropriate privileges for creating metadata-bound libraries and accessing metadata-bound data. The combined effect of permissions applied on metadata objects and capabilities granted for an application determines the privileges of all users within the environment. Not having a well-planned metadata security model in place before implementing metadata-bound libraries can result in undesired outcomes for users accessing SAS data. For more information on how to design an effective SAS metadata security model design, please reference this paper on Getting Started with Designing and Implementing a SAS® 9.4 Metadata and File System Security Design.

All secured library and table objects are created under the */System/Secured Libraries* metadata folder. Administrators should create and secure sub-folders as per requirements under which users create the metadata-bound library objects.

MANAGE PASSWORDS AND ENCRYPTION KEYS

Each metadata-bound library requires a password that is used to physically password-protect the SAS data, and optionally an encryption key to encrypt the data. It is recommended that administrators determine a process for storing and maintaining passwords and encryption keys. The password and encryption key are required when modifying metadata-bound library attributes. Therefore, it is recommended that administrators manage the values used for these attributes. Administrators might choose to use the same password and encryption key for all metadata-bound data, or they might choose

to create a unique password and encryption key for each line of business. It is recommended that the password and encryption key attributes are not shared with data creators who are responsible for creating metadata-bound libraries. Only administrators should manage passwords and encryption keys.

SETUP CODE DIRECTORY

Administrators should create a secured location for source code and compiled code that is used to implement this solution. Users creating metadata-bound libraries should have read access to SAS macro compiled code and no access to the SAS macro source code. Administrators can provide template programs that provide macro definitions for execution by users. Users need only to define variables such as directory path, metadata-bound library name, and secured library folder path into these macro programs.

DEVELOPMENT

Metadata-bound libraries can be configured via an interactive user interface within SAS Management Console, or programmatically using the AUTHLIB procedure. In order to allow administrators to have more control over the attribute values outlined in this paper, it is recommended to use SAS macros and programs for creating, managing, and monitoring metadata-bound libraries using the AUTHLIB procedure. The code provided in this section can be encapsulated in SAS macros that can be distributed for execution by users; thus eliminating the need for these users to use SAS Management Console for managing metadata-bound libraries. SAS macros can be created for each task that a user might need to perform on a metadata-bound library.

The syntax for the AUTHLIB procedure is included below, and only includes statements and options that are being used as part of this solution. For details regarding the full syntax of the AUTHLIB procedure, refer to the AUTHLIB Procedure section in the SAS® 9.4 Guide to Metadata-Bound Libraries.

```
PROC AUTHLIB <option(s)>;
      CREATE
            SECUREDLIBRARY='secured-library-name'
            <SECUREDFOLDER='secured-folder-path'>
            <LIBRARY=libref>
            PW=all-password-value </ new-all-password-value> |
            <REQUIRE ENCRYPTION=YES | NO>
            <ENCRYPT=YES | NO | AES>
            <ENCRYPTKEY=key-value </ new-key-value>>;
      MODIFY <LIBRARY=libref>
            PW=all-password </ new-all-password> |
            <TABLESONLY=YES | NO>
            <REQUIRE ENCRYPTION=YES | NO>
            <ENCRYPT=YES | NO | AES>
            <ENCRYPTKEY=key-value </ new-key-value>>;
      REMOVE<LIBRARY=libref>
            PW=all-password </ <new-all-password>> |
            <TABLESONLY=YES | NO>
            <ENCRYPT=YES | NO | AES>
            <ENCRYPTKEY=key-value </ new-key-value>>;
      REPORT <LIBRARY=libref>
            <ENCRYPTKEY=key-value>;
      TABLES SAS-dataset(s) | ALL | NONE
            </>
            <PW=all-password > </ <new-all-password>> |
            <ALTER=alter-password> </ <new-alter-password>>
            <READ=read-password> </ <new-read-password>>
            <WRITE=write-password> </ <new-write-password>>;
            <MEMTYPE= DATA | VIEW>
```

```
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=key-value< / new-key-value>>;
```

The following sections describe the different tasks that users need to perform when implementing metadata-bound libraries and the SAS code for performing those tasks.

CREATING METADATA-BOUND LIBRARIES

Data creators should create and maintain secured library and secured table objects for SAS data directories that are created and managed by them. The code provided in this section uses the CREATE statement of the AUTHLIB procedure to create a new metadata-bound library. This code can be assigned to a macro that administrators can compile and make available to data creators. By using compiled macro code, administrators can control the number of attributes that can be specified by data creators.

The code to create the metadata-bound library requires the following values:

- LIB= specifies the libref for the physical library path to be secured.
- SECUREDLIBRARY= specifies the name of the secured library metadata object.
- SECUREDFOLDER= specifies the metadata path of the target folder for the secured library metadata object.
- PW= is the password for the secured library. The password must be a minimum of 8 characters in length. ***TIP***: Use PROC PWENCODE to use an encoded value for the password.

The following optional values can be used to configure additional metadata-bound library attributes:

- REQUIRE_ENCRYPTION=YES | NO specifies whether encryption is enforced on all SAS datasets.
- ENCRYPT=AES | AES2 specifies the encryption algorithm.
- ENCRYPTKEY= specifies the passphrase from which an encryption key for AES/AES2 encryption is derived. The maximum length is 64 characters.

```
libname myencdat "/my/data/directory";

proc authlib lib=myencdat;
      create securedlibrary="FirstMBL"
             securedfolder="/System/Secured Libraries/UserMBL"
             pw="Password123"
             require_encryption=yes
             encrypt=AES
             encryptkey="EncKey123";
run;
quit;
```

UPDATING METADATA-BOUND LIBRARIES

Administrators might need to update the password and/or encryption key for existing metadata-bound libraries as part of on-going security maintenance. To reduce the effort required to perform such an activity, data creators should perform these updates on their respective metadata-bound libraries. The code provided in this section uses the MODIFY statement of the AUTHLIB procedure to update the password and/or encryption key for an existing metadata-bound library. This code can be made available to data creators as compiled macro code.

The code to modify password and/or encryption key attributes for an existing metadata-bound library requires the following values:

- LIB= specifies the libref for the physical library path to be secured.

- PW= specifies the current [and new] password for the secured library; new password must be a minimum 8 characters in length. ***TIP***: Use PROC PWENCODE to use an encoded value for the password.

Note that with the MODIFY statement, the SECUREDLIBRARY= and SECUREDFOLDER= attributes are not explicitly defined. Since the binding information is stored at the physical layer, assigning a libref to the physical library is sufficient to identify the corresponding secured library metadata object.

The following optional values can be used to modify encryption:

- REQUIRE_ENCRYPTION=YES specifies whether to enforce encryption on all SAS datasets
- ENCRYPT=AES | AES2 specifies the encryption algorithm. ***NOTE***: Although, it is recommended to use AES encryption, SAS Proprietary encryption algorithm is also available to use by specifying YES as value for this attribute.
- ENCRYPTKEY= specifies the current [and new] encryption key. The value specified is the passphrase from which an encryption key for AES encryption is derived. The maximum length is 64 characters.

The current password needs to be specified if it's being changed to a new password, if encryption is being applied, or if the encryption key is being changed. The current encryption key needs to be specified if it's being changed to a new encryption key.

```
libname myencdat "/my/data/directory";

proc authlib lib=myencdat;
      modify pw="currentpassword"/"newpassword"
             require_encryption=yes
             encrypt=AES
             encryptkey="currentkey"/"newkey";
run;
quit;
```

BINDING A TABLE TO METADATA

To maintain security bindings, metadata-bound tables should not be copied, moved or replaced using host commands. If host commands are used to copy a physical table to a metadata-bound library, the physical table is not automatically bound to metadata. In such situations, the user might be required to use host commands to delete the host-copied physical tables and use SAS to copy or replace the data sets. An alternative is to use the AUTHLIB procedure with the MODIFY statement on the metadata-bound library that contains host-copied tables. The MODIFY statement can be executed with the current password to register host-copied tables without actually changing the password or encryption key values. This is not a recommended approach but can be used in situations where a quick resolution is required for registering host-copied tables.

The code to bind a table to a secured library requires the following values:

- LIB= specifies the libref for the physical library path to be secured.
- PW= specifies the current password for the secured library.

```
libname myencdat "/my/data/directory";

proc authlib lib=myencdat;
      modify pw="currentpassword";
run;
quit;
```

UNBINDING A METADATA-BOUND LIBRARY

If required, the metadata binding of a physical library to a secured library can be removed using the REMOVE statement of the AUTHLIB procedure. This unbinds the library as well as the tables in the library, and as such, the library and its tables are no longer password-protected or encrypted. Any new tables registered to the library after it is unbound are not secured.

The code to unbind a secured library requires the following values:

- LIB= specifies the libref for the physical library path to be secured.
- PW= specifies the password for the secured library, followed by "/". The forward slash (/) after the password value ensures that the unbound tables are not password protected.
- ENCRYPT=NO ensures that the unbound tables are no longer encrypted. ***NOTE***: By specifying YES for this attribute, the unbound tables can remain encrypted but their encryption key is no longer stored in metadata. The encryption key value is required when accessing these unbound encrypted tables.

```
libname myencdat "/my/data/directory";

proc authlib lib=myencdat;
      remove pw="currentpassword" /
      encrypt=no;
run;
quit;
```

UNBINDING A METADATA-BOUND TABLE

The binding of individual physical tables to metadata can also be removed without affecting the library or any other tables in the secured library. To remove the binding of one or more physical tables in a library, use the TABLES statement with the REMOVE statement of the AUTHLIB procedure. The result is one or more tables that are no longer being password-protected or encrypted within a secured library. The TABLESONLY=YES attribute ensures that the binding of the secured library remains unchanged.

The code to unbind one or more tables in a secured library requires the following values:

- LIB= specifies the libref for the physical library path to be secured.
- PW= specifies the password for the secured library, followed by "/". The forward slash (/) after the password value ensures that the unbound tables are not password protected.
- ENCRYPT=NO ensures that the unbound tables are no longer encrypted. ***NOTE***: By specifying YES for this attribute, the unbound tables can remain encrypted but their encryption key is no longer stored in metadata. The encryption key value is required when accessing these unbound encrypted tables.
- TABLESONLY=YES ensures that unbinding occurs only for the specified tables.
- TABLES 'tablename' specifies the names of one or more tables to unbind from. metadata

```
libname myencdat "/my/data/directory";

proc authlib lib=myencdat;
      remove pw="currentpassword" /
      encrypt=no
      tablesonly=YES;
      tables tablename1 tablename2;
run;
quit;
```

VALIDATING METADATA-BOUND LIBRARIES

To ensure that there are no discrepancies between the physical objects and their corresponding secured objects in metadata, the REPORT statement of the AUTHLIB procedure can be used to create a report for the specified metadata-bound library.

The code to report on a secured library requires the following attributes:

- LIB= specifies the libref for the physical library path to be secured.

```
libname myencdat "/my/data/directory";

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

EXECUTION

The above sample code snippets for creating, updating, binding, unbinding, and reporting on metadata-bound libraries can be assigned to individual macros for better control and manageability of securing sensitive data. The STORE option in a %MACRO statement can be used to compile and save the macro code in a library catalog. Data creators can simply call the macro in their session, which executes the corresponding compiled macro code. By using this method, administrators can hide from data creators the password and encryption key values that are contained in the macro source code.

Unique macros can optionally be set up with different password and encryption key values for lines of business. In this scenario, data creators should only have access to the corresponding compiled macros for their line of business. With this approach users are not required to have knowledge of the password and encryption key that is in use, but are still able to efficiently maintain security of their SAS data.

MONITORING

Metadata-bound libraries are often used to enforce encryption of on-disk SAS data, or data at rest. When data security requires enforcing encryption of all SAS data, it is important for administrators to monitor and ensure encryption is being applied to all data. To efficiently monitor and report on any unencrypted SAS data, it is recommended to implement a process that scans the file system and generates a list of unencrypted SAS data, its physical location on the file system, and additionally the owner of the physical library in the case of UNIX environments.

The sample code provided in this SAS Note can be modified to recursively scan a file system for SAS7BDAT extension files. With slight modifications, this code can be used in both Linux and Windows file systems. In a Linux file system, some additional attributes such as the file system owner, group owner, permissions, and last modified date can also be extracted using the following code snippet.

```
%let owner=%sysfunc(dinfo(&did, Owner Name));
%let grpowner=%sysfunc(dinfo(&did, Group Name));
%let perm=%sysfunc(dinfo(&did, Access Permission));
%let lstmod=%sysfunc(dinfo(&did, Last Modified));
```

A LIBNAME statement can be issued for each of the identified directories containing SAS data. The LIBNAME execution for each SAS data directory populates the DICTIONARY.TABLES view. The DICTIONARY.TABLES view captures detailed information for all librefs and tables during a SAS session. The DICTIONARY.TABLES view also has encryption information for the SAS data sets.

The information extracted from the file system combined with information from the DICTIONARY.TABLES view can be used to generate comprehensive reports. These reports can be used to monitor all SAS data and alert administrators or users of unsecured or unencrypted SAS data.

The DICTIONARY.TABLES view provides four different values for the encryption status of a table:

- AES2: data is AES2 encrypted
- AES: data is AES encrypted
- YES: data is SAS Proprietary encrypted
- NO: data is not encrypted

The program for generating these reports can be scheduled for regular monitoring, or executed interactively in a SAS client application or batch submitted for ad hoc reporting.

The program that extracts the encryption information for a data set reads the physical table header. This requires opening the SAS data set, which means that for a metadata-bound table, metadata layer authorization checks occur against the identity that is executing the program. The account running the program requires a minimum of Read access to all directories and tables. Therefore, an account that has appropriate metadata-layer access and physical-layer access, allowing to extract information for all SAS data sets, should be used when running a monitoring program. For UNIX systems, it is recommended to use the root user since it has access to all files on the file system. The metadata connection options can use a different account, such as the SAS administrator account (sasadm@saspw), for ensuring unrestricted access to all metadata.

The following sample code can be used to combine the SAS data directory information that is extracted from the file system with information available in the DICTIONARY.TABLES view for that library. In the code below, the `scan.dir_info` table contains the directory path, file system owner, group owner, permission, and last modified information for scanned SAS data directories in the file system. The macro `libprefix` contains the libref prefix value that is used for issuing a LIBNAME statement in the scan program. The DICTIONARY.TABLES view contains information for all librefs in the current session, so the `libprefix` macro is used to filter only the desired librefs that are executed from the scan program.

```
proc sql;
      create table scan.monitor_liball as select libname, dirpath
      'Directory', memname 'Member Name', memtype 'Member Type', encrypt
      'Encryption', owner 'Owner Name', grpowner 'Group Name', perm 'Access
      Permission', lstmod 'Last Modified'
      from dictionary.tables, scan.dir_info where libname
      like UPCASE("&libprefix%")
      order by dirpath, memname;
quit;
```

MAINTENANCE

GENERAL USAGE GUIDELINES FOR USERS

1. Do not use host commands to manage physical tables.
2. Always connect to the SAS® Metadata Server before issuing a LIBNAME statement for a metadata-bound library.
3. Create or modify a metadata-bound library at a time when the underlying physical data is not being accessed by other users.
4. For maintaining security, generate reports on metadata-bound libraries and report any inconsistencies to administrators.
5. Work with an administrator to resolve inconsistencies in a metadata-bound library or data sets.

WORKING WITH METADATA-BOUND DATA

Use the COPY procedure or DATASETS procedure to copy new data sets in the following situations:

1. from an unencrypted metadata-bound library to an encrypted metadata-bound library
2. from a traditional library to an encrypted or unencrypted metadata-bound library
3. from an unencrypted metadata-bound library to a traditional library

Using this approach creates a security binding for the new data set and registers it in the target metadata-bound library.

The COPY or DATASETS procedures should not be used when data sets from an encrypted metadata-bound library need to be copied to an unencrypted metadata-bound library or a traditional library. In this case, since the target library is not encrypted, the use of COPY or DATASETS procedures will result in the following error:

ERROR: Missing ENCRYPTKEY option on member libref.tablename.

Instead, the SQL procedure should be used to read from the encrypted data and write into a data set in the non-encrypted target library.

Use the DATASETS procedure or SQL procedure to delete metadata-bound data sets. Using SAS as an interface for interacting with metadata-bound data ensures the security of data and eliminates inconsistencies between the physical objects (libraries and data sets) and their corresponding metadata objects. SAS interfaces include, but are not limited to the SAS procedures that are mentioned above or SAS client applications such as SAS® Enterprise Guide or SAS® Studio.

VALIDATING FREQUENTLY

Users responsible for creating and managing metadata-bound libraries should periodically generate a validation report for their metadata-bound libraries. This is beneficial in identifying and fixing any inconsistencies with missing physical or metadata objects and security information. The REPORT statement of the AUTHLIB procedure that was explained in a previous section can be used for validating the metadata-bound libraries. Inconsistencies should always be reported to administrators.

HANDLING MIXED STATES

The information presented in this section should be strictly available to administrators only. It is not fully comprehensive and should be used as a general guide for working with the solution provided in this paper. If required, administrators can work with a user to perform any of these tasks as needed.

The REPAIR statement of the AUTHLIB procedure has not been included as part of this solution, given that it is preproduction feature for SAS® 9.4. Reference the SAS(R) 9.4 Guide to Metadata-Bound Libraries for information on the REPAIR statement; and as recommended, contact SAS Technical Support if there is a need to use this statement.

1. **Bind unsecured tables that have been copied using host commands into the metadata-bound library (Not Recommended)**

   It is recommended to use the COPY procedure or SAS client applications for replacing or copying SAS data sets to a secured library.

   If needed, administrators can work with data creators and use the code provided in this paper for binding individual tables to a secured library.

2. **Bind secured tables that have been copied using host commands into another metadata-bound library (Not Recommended)**

   If a metadata-bound table from one library has been copied using host commands to another metadata-bound library (not recommended), it is required that the security information for this metadata-bound table is updated to the new metadata-bound library. The simple approach to resolving this situation is to delete the new table from the host and then use the COPY procedure.

If deleting from the host and copying using the COPY procedure is not an option, then an administrator should work with the data creators to unbind the metadata-bound table such that it is no longer secured by a password and encryption key. After unbinding a table, it should be registered to the new metadata-bound library.

IDENTITY RESOLUTION

BASE ENGINE LIBRARIES

BASE engine libraries that access metadata-bound data require users to be connected to the SAS Metadata Server. Authorization decisions are based on the identity that is used to connect to the SAS Metadata Server. When accessing BASE engine libraries through SAS client applications, the user ID specified in the connection profile, in that application, is used to make the authorization decisions for access to metadata-bound data.

When submitting batch jobs that access metadata-bound libraries, it is required that a connection to the SAS Metadata Server is established. Metadata connection options can be used in SAS code to initialize the metadata connection. The user ID used for the metadata connection determines authorizations on metadata-bound data.

SAS/SHARE LIBRARIES

In an environment that is using metadata-bound libraries, the SAS/SHARE server should be started with the PROC SERVER AUTHENTICATE=REQUIRED option. With the AUTHENTICATE=REQUIRED option it is required that all connections from the client to the server use a valid user ID and password.

When accessing metadata-bound data using a SAS/SHARE library, the authorization decisions are made either through the user ID that connects to the SAS/SHARE server using PROC OPERATE, or the server connection information that is specified in the LIBNAME statement. Authorization checks are performed against the user ID that is provided in the LIBNAME statement, except in the following scenarios:

- 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)

For the above two scenarios, the authorization checks are performed using the user ID with which the client authenticates to the SAS/SHARE server.

SETUP SAS/SHARE SERVER STARTUP

When SAS/SHARE is used to access data in a SAS platform using metadata-bound libraries, certain options are required for starting the SAS/SHARE server. The following steps should be followed to ensure correct options are used for starting the SAS/SHARE server:

1. Stop the SAS/SHARE server.
2. Update the SAS/Share server startup script to include the following options in the beginning of the script:

```
%let TCPSEC=_secure_;
options comamid=tcp;
options metaport=8565
        metarepository="Foundation"
        metaserver="metadata.server.name"
        metauser="user" metapass='encoded password';
```

3. Add the AUTHENTICATE=REQUIRED option in the PROC SERVER statement:

```
proc server id=sasshare authenticate=required;
run;
```

4. Start the SAS/SHARE server with the updated script.
5. Check the SAS/SHARE server log to ensure successful startup.

CONCLUSION

A managed process is necessary for a successful implementation and efficient use of metadata-bound libraries. The information provided in this paper can be used as guidelines to design a process that is most suitable for any environment. The goal of creating this process is to share responsibilities between administrators and data creators, thus, reducing administrative overhead without compromising on the level of security and control exerted by administrators. The variables in a managed metadata-bound library implementation could be policies around passwords, encryption algorithm, encryption keys, operating system, file system security, and SAS metadata security. Administrators should take these variables into account in order to create a comprehensive implementation process.

ACKNOWLEDGEMENTS

I would like to thank Angie Hedberg for her constant support, providing valuable input, and reviewing this paper.

RECOMMENDED READING

SAS® 9.4 Guide to Metadata-Bound Libraries:
https://support.sas.com/documentation/cdl/en/seclibag/66930/PDF/default/seclibag.pdf

SAS® 9.4 Intelligence Platform: Security Administration Guide:
http://support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf

Getting Started with Designing and Implementing a SAS® 9.4 Metadata and File System Security Design:
http://support.sas.com/resources/papers/proceedings17/SAS0709-2017.pdf

CONTACT INFORMATION

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

Deepali Rai
SAS Institute Inc.
deepali.rai@sas.com