THE POWER TO KNOW.

SAS® 9.2 Intelligence Platform

Security Administration Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® 9.2 Intelligence Platform: Security Administration Guide*. Cary, NC: SAS Institute Inc.

SAS® 9.2 Intelligence Platform: Security Administration Guide

ISBN: 978-1-59994-072-4

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

1st electronic book, February 2009
2nd electronic book, May 2009
3rd electronic book, September 2009
4th electronic book, March 2010
5th electronic book, March 2011
6th electronic book, October 2011
1st printing, March 2009
2nd printing, May 2009

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

What's New

Overview

New and enhanced features in the following areas increase security and manageability:

- roles and permissions
- authentication and user management
- auditing, reporting, and encryption
- documentation

Roles and Permissions

- Expanded support for roles enables you to easily customize the user interface in applications including SAS Web Report Studio, SAS Enterprise Guide, the SAS Add-In for Microsoft Office, and SAS Management Console.
- New administrative roles enable you to manage unrestricted access to metadata, user administration capabilities, the ability to operate the metadata server, and the visibility of plug-ins in SAS Management Console.
- The authorization interface always displays effective permissions (a calculation of the net effect of all applicable metadata layer permission settings).
- All applications share a single folder tree. Access control inheritance for most items flows through that tree. Inheritance can cross repository boundaries. Each schema, cube, library, and table inherits permissions from only its parent folder. These inheritance paths are discontinued:
  - **application server ▶ OLAP schema ▶ cube**
  - **application server ▶ library ▶ table**
  - **DBMS server ▶ DBMS schema ▶ DBMS table**
- A new permission, WriteMemberMetadata, enables you to separate the ability to interact with items in a particular folder from the ability to make changes to the folder itself.

- When you define member-level access to OLAP data, you can use a graphical query-builder to build the permission conditions.
- The SAS.IdentityGroups identity-driven property enables you to make row-level or member-level distinctions based on metadata group memberships.
- In BI row-level permissions, an empty string is substituted into the query for an identity who has no value for the identity-driven property that is in use. Previously, a missing value caused the query to fail.
- You can set permissions from within SAS OLAP Cube Studio, SAS Data Integration Studio, SAS Management Console, and SAS Information Map Studio.
- A new **Advanced** button on each item's **Authorization** tab enables unrestricted users to trace the item's inheritance and look up the permissions that any identity has to the item.
- DATA step functions enable you to programmatically define (and query) metadata layer authorization settings. See the *SAS Language Interfaces to Metadata*.

Authentication and User Management

- Users who are logged on to their Windows desktop can seamlessly launch SAS desktop clients (if the metadata server runs on Windows). This feature, Integrated Windows authentication, is particularly useful for sites that use smart cards, biometrics, or other forms of multi-factor authentication.
- Users who have authenticated to the metadata server access most SAS servers seamlessly. SAS token authentication causes the OLAP server, the stored process server, and, in some configurations, the workspace server to accept users who have authenticated to the metadata server.
- You don't have to create external accounts for SAS internal purposes. Instead, you can use internal accounts that exist only in the metadata. It is appropriate to use internal accounts for administrators and some service identities.
- For greater security, you can limit use of trusted peer connections.
- Membership in a new user administration role enables you to manage most users, groups, and roles. You can still use permissions to delegate administration of an existing identity.
- If you have user administration capabilities, you can directly manage authentication domains from the User Manager and Server Manager plug-ins in SAS Management Console.
- Regular users can't change their own user definitions. Regular users can still manage their own personal logins.
- If a user's group memberships make more than one login available in an authentication domain, the highest priority login is used. Priority is determined by identity precedence. This is an aspect of credential management.
- You can use logins on the PUBLIC and SASUSERS groups. This enables you to provide single sign-on to a third-party server using one account that is shared by all users. This is an aspect of credential management.
- You can add, modify, and remove external identity values for users, groups, and roles in SAS Management Console. These values support the identity synchronization process.
- You can give each user, group, and role a display name. For an identity that doesn't have a display name, the name serves as the display name.
- You can use the SASSEC_LOCAL_PW_SAVE option in the metadata server's omaconfig.xml file to control whether the **Save user ID and password in this**

profile check box is available to users. This is an aspect of password management.

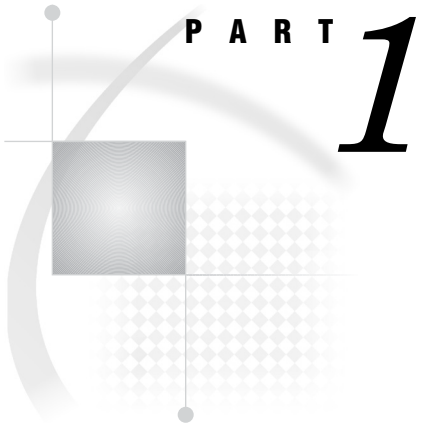
- You can load identity information into the metadata in blocks by using the MDUIMPLB and MDUCHGLB macros. This is a performance enhancement to the user import process. The corresponding macros from the previous release (MDUIMPL and MDUCHGL) are still supported.
- You don't have to give users the Windows privilege **Log on as a batch job** unless they access a standard workspace server using credential-based host authentication.

Auditing, Reporting, and Encryption

- You can use system-wide logging features to audit security events.
- You can use the MDSECDS macro to create authorization data sets for security reporting purposes.
- Configuring encryption of data in transit among SAS clients and servers is no longer a post-installation task. You select an over-the-wire encryption level (what gets encrypted in transit) and algorithm (what type of encryption or encoding is used) during installation.
- By default, passwords in the metadata are encrypted using an industry-standard algorithm (AES fixed key). If you don't have SAS/SECURE, SASProprietary encoding is used instead.
- By default, the PWENCODE procedure uses SASProprietary encoding (sas002). If you have SAS/SECURE, you can choose to use AES encryption (sas003) instead.

Documentation Enhancements

- This document has been reorganized and rewritten for this release.
- A new document, *SAS Management Console: Guide to Users and Permissions*, provides step-by-step instructions for performing selected tasks in SAS Management Console.
- *SAS Intelligence Platform: Web Application Administration Guide* documents security features of the SAS Content Server.



PART 1

Fundamentals

CHAPTER

1

Security Features

About this Document

This document helps you understand and use the security features of the SAS Intelligence Platform. This document assumes that you are familiar with the concepts and terminology that are introduced in *SAS Intelligence Platform: Overview*.

This document is organized as follows:

- The first four chapters contain the following basic information:
  - a brief overview of security features
  - instructions for the main security tasks
  - essential facts about SAS users, groups, and roles
  - an orientation to using SAS metadata-layer permissions
- The other chapters contain detailed reference information and instructions for specialized tasks.

Accessibility Features of the SAS Intelligence Platform

For information about accessibility for any of the products mentioned in this document, see the online Help for that product. If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.

Security in the SAS Intelligence Platform

The security features of the SAS Intelligence Platform offer these benefits:

- single sign-on from and across disparate systems
- secure access to data and metadata

- role-based access to application features
- confidential transmission and storage of data
- logging and auditing of security events
- access control reporting

The SAS Intelligence Platform is not an isolated system. The platform's security model cooperates with external systems such as the host environment, the Web realm, and third-party databases. To coordinate identity information, SAS keeps a copy of one ID (such as a host, Active Directory, LDAP, or Web account ID) for each user. This requirement does not apply to any users for whom a generic PUBLIC identity is sufficient.

Permissions Overview

SAS provides a metadata-based authorization layer that supplements protections from the host environment and other systems. Across authorization layers, protections are cumulative. In order to perform a task, a user must have sufficient access in *all* applicable layers.

You can use the metadata authorization layer to manage access to the following resources:

- almost any metadata object (for example, reports, data definitions, information maps, jobs, stored processes, and server definitions)
- OLAP data
- relational data (depending on the method by which that data is accessed)

CAUTION:
Some clients (such as SAS Data Integration Studio and SAS Enterprise Guide) enable power users to create and run SAS programs that access data directly, bypassing metadata-layer controls. It is important to manage physical layer access in addition to metadata-layer controls. For example, use host operating system protections to limit access to any sensitive SAS data sets. See "Host Access to SAS Tables" on page 187. △

CAUTION:
Not all permissions are enforced for all items. Enforcement of permissions other than ReadMetadata and WriteMetadata varies by item type and (for data) by the method with which a library is assigned. △

Roles Overview

The SAS implementation of roles enables you to manage the availability of application features such as menu items, plug-ins, and buttons. For example, your role memberships determine whether you can see the **`Server Manager`** plug-in (in SAS Management Console), compare data (in SAS Enterprise Guide), or directly open an information map (in SAS Web Report Studio).

Here are some key points about the SAS implementation of roles:

- Roles are an entirely separate concept from permissions. In general, roles don't affect access to metadata or data. An exception is that the unrestricted role provides irrevocable grants of all permissions in the metadata authorization layer. This enables unrestricted users to manage all metadata.

- Not all applications have roles. Applications that have roles include the SAS Add-In for Microsoft Office, SAS Enterprise Guide, SAS Management Console, and SAS Web Report Studio.
- Not all application features are under role management. An application feature that is under role management is called a capability. Each application that supports roles provides a fixed set of capabilities. You can't convert a feature that isn't a capability into a capability. However, if you add custom tasks or develop custom plug-ins, you can register those features as capabilities.
- All capabilities are additive. There are no capabilities that limit what you can do.
- Capabilities can be categorized as follows:
  - An explicit capability can be incrementally added to or removed from any role (other than the unrestricted role, which always provides all explicit capabilities). Most roles have explicit capabilities.
  - An implicit capability is permanently bound to a certain role. The metadata server's roles provide implicit capabilities. For example, the user administration role provides the capability to add users, but there is no explicit **`Create Users`** capability.
  - A contributed capability is an implicit or explicit capability that is assigned through role aggregation. If you designate one role as a contributing role for another role, all of the first role's capabilities become contributed capabilities for the second role.
- You can't assign permissions to a role or capabilities to a group.
- A user can't temporarily assume or relinquish a role; all of a user's roles are active at all times. Administrators can have two user definitions so they can function as regular users some of the time. See "How to Create a Dual User" on page 29.
- If you need detailed information about an application's capabilities and default roles, see the administrative documentation for that application.

Single Sign-On Overview

SAS provides these single sign-on (SSO) features:

- To bypass the logon prompt when launching a desktop application (such as SAS Information Map Studio, SAS Enterprise Guide, SAS Data Integration Studio, SAS OLAP Cube Studio, or SAS Management Console), use Integrated Windows authentication. The client and the metadata server must be in the same Windows domain or in domains that trust each other. See "Integrated Windows Authentication" on page 148.

  Note: An alternate method for avoiding the initial logon prompt for a desktop application is to save credentials in a client-side connection profile. This method is supported on all operating systems. If you have more than one connection profile, designate a profile that includes your credentials as your default connection profile. Some sites don't allow users save credentials in their connection profiles. See "Password Policies" on page 14. △
- To bypass the logon prompt when launching a SAS Web application (such as SAS Web Report Studio or SAS Information Delivery Portal), use Web authentication. See "Web Authentication" on page 156.
- Seamless access to data servers and processing servers is provided by mechanisms including SAS token authentication, Integrated Windows authentication, credential reuse, and credential retrieval. See "Authentication to Data Servers and Processing Servers" on page 130.

Encryption Overview

SAS offers encryption features to help you to protect information on disk and in transit. Here is an overview of encryption support in the SAS Intelligence Platform:

- Passwords in configuration files and the metadata are encrypted or encoded. Most other metadata is not encrypted.
- Passwords in transit to and from SAS servers are encrypted or encoded. You can choose to encrypt all traffic instead of encrypting only credentials.

Security Reporting and Logging Overview

Security reporting creates a snapshot of metadata-layer access control settings. SAS provides the %MDSECDS autocall macro to enable you to easily build data sets of permissions information. You can use those data sets as the data source for security reports. You can also identify changes in settings by comparing data sets that are generated at different times. See Chapter 10, "Security Report Macros," on page 115.

Security logging records security-related events as part of a system-wide logging facility. The following table describes the security log categories:

Table 1.1 Logging of Security Events

Category	Events Captured
Audit.Authentication	Authentication events, client connection information.
Audit.Meta.Security.UserAdm	Changes to users, groups, roles, logins, and authentication domains. Includes additions, deletions, modifications, and failed attempts to perform these actions.
Audit.Meta.Security.GrpAdm	Changes to memberships (for groups or roles). Includes adding members, removing members, and failed attempts to perform these actions.
Audit.Meta.Security.AccCtrlAdm	Changes to permissions, permission settings, ACTs, and passwords.* Includes additions, deletions, modifications, and failed attempts to perform these actions.
Audit.Meta.Security	The parent category for security events. Logging settings that you define for this category apply to its child categories.

* This is for passwords on objects such as Tables, Connections, and ProtectedPassthrus.

CHAPTER

2

Security Tasks

Introduction to Security Tasks

This chapter provides instructions for the main security administration activities. This chapter includes scenario-specific recommendations for these reasons:

- Different sites have different starting points. For example, migrated deployments include user definitions that were created in the original environment, whereas new deployments include only a few predefined users.
- Different sites use different hosts. For example, a Windows site might choose to implement Integrated Windows authentication, whereas a UNIX site can't use that feature.
- Different sites have different priorities. For example, one site might choose to initially grant broad access to data, whereas another site might need to provide that access selectively.

Note: Where possible, the instructions include sufficient detail to enable someone who is new to the material to perform standard tasks successfully. For specialized tasks, concepts, and background information, references to other chapters are provided. △

See Also

Appendix 1, "Checklists," on page 211

Facilitate Authentication

Identify or Create User Accounts

In order to log on to a SAS client, a user must have an account that can provide access to the metadata server. Determine which of the following situations applies to you and complete any tasks as indicated.

- In the simplest case, users already have accounts that are known to the metadata server's host. For example, the metadata server is on UNIX, and users have accounts in an LDAP provider that the UNIX host recognizes. Or the metadata server is on Windows, and users have Active Directory accounts. No action on your part is required.
- In some cases, users have accounts that aren't currently recognized by the metadata server's host. Consider the examples in the following table.

Table 2.1 Incorporating Unrelated Accounts

Scenario	Possible Solution[1]
You have Active Directory accounts but the metadata server is on UNIX.	Enable the UNIX host to recognize the accounts. See "Pluggable Authentication Modules (PAM)" on page 149.
You have accounts in an LDAP provider that isn't known to the metadata server's host.	Enable the metadata server itself to recognize the LDAP provider. See "Direct LDAP Authentication" on page 146.
You have accounts that are known at your Web perimeter but aren't known to the metadata server's host.	Enable the metadata server to trust users who have authenticated at the Web perimeter. See "Web Authentication" on page 156.[2]

1 None of these options enable you to avoid later creating corresponding identity information in the SAS Metadata Repository.

2 This is only a partial solution, because users of desktop applications still need accounts that can be validated by the metadata server or its host.

Note: Even if your Web accounts are recognized by the metadata server's host, you might choose to use Web authentication so that SAS Web applications are launched silently. △

- In other cases, you must add accounts to your environment. Although it is technically possible to instead use SAS internal accounts for this purpose, those accounts aren't intended for regular users.

Note: Someone who directly connects to the OLAP server needs an account with the OLAP server. This situation occurs when someone uses a data provider to access SAS OLAP data from the SAS Add-In for Microsoft Office. △

Coordinate the Workspace Server

Seamless access to the workspace server depends on coordination between that server and the metadata server. This coordination is necessary because authentication

to the workspace server is, by default, performed by the workspace server's host. The following table provides general recommendations:

Table 2.2 Coordinate the Workspace Server With the Metadata Server

Scenario	Recommendation
Both servers are on Windows.	The preferred approach is to ensure that both servers offer Integrated Windows authentication (IWA) for desktop clients. See "Integrated Windows Authentication" on page 148. An alternative is to use credential-based host authentication for both servers. See "Host Authentication" on page 147.
Both servers are on UNIX.[1]	Use credential-based host authentication for both servers. No action on your part is required.
The two servers don't recognize the same accounts.[2]	To minimize requirements for and exposure of host credentials, SAS provides several alternate configurations. See "Mixed Providers" on page 135.

1 Or both servers are on z/OS.
2 For example, one server is on Windows and the other server is on UNIX.

Note: Similar coordination for other types of SAS servers isn't necessary because those servers don't use host authentication for metadata-aware connections. △

See Also
"About User Administration" on page 21
"Single Sign-On Overview" on page 5

Add Administrators

For accountability, we recommend that you establish individual metadata administrators rather than sharing the unrestricted SAS Administrator account.

1. Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).
2. On the **Plug-ins** tab, select **User Manager** (in the foundation repository).
3. Create a user definition for each administrator:
   a. Right-click and select **New** ▶ **User**.
   b. On the **General** tab, enter a name.

      Note: The administrator's internal user ID is based on this name, so it is a good idea to use a short identifier here. △

   c. On the **Accounts** tab, click **Create Internal Account**. In the **New Internal Account** dialog box, enter and confirm an initial password. Click **OK**.

      Note: By initial policy, internal passwords must be at least six characters, don't have to include mixed case or numbers, and don't expire. If you want to force a password change on first use, set a password expiration period. △

   d. Click **OK** to save the new internal user.
4. Provide privileges for each administrator:
   a. Right-click the **SAS Administrators** group and select **Properties**. On the **Members** tab, move the new users to the **Current Members** list. Click **OK**.

b (Optional) Right-click the **Metadata Server: Unrestricted** role and select **Properties**. On the **Members** tab, move the new user to the **Current Members** list. Click **OK**. To perform this step, you must be unrestricted.

Note: Step 4b establishes a single level of administrative privilege. If you omit step 4b for an administrator, that administrator can perform almost all metadata administrative tasks but is subject to all permission and capability requirements. See "Main Administrative Roles" on page 28. △

Here are some details and tips:

- If you log on with an internal account, you must include the @saspw suffix in the user ID that you submit (for example, sasadm@saspw). See "SAS Internal Authentication" on page 150.
- To conform to the rule of least privilege, do not use an administrative identity to perform regular user tasks. See "How to Create a Dual User" on page 29.
- The advantage of using an internal account in step 3c is that this facilitates creation of a dual user, because this approach leaves the user's external account available for use in a second user definition. A disadvantage of using an internal account is that such an account can't launch a standard workspace server. These administrators are prompted for host credentials if they attempt to validate or use that server. See "Who Can Launch a Standard Workspace Server?" on page 187.

  Note: It is possible to avoid the prompt by replacing the step 3c instructions above with the step 3c instructions from the following topic. However, this makes it difficult to establish a dual user (because each account can be referenced in only one user definition). △

Add Regular Users

For accountability, we recommend that you create an individual SAS identity for each person who uses the SAS environment. This enables you to make access distinctions and audit individual actions in the metadata layer. This also provides a personal folder for each user. To create a SAS identity for someone, make a SAS copy of the ID with which that person logs on to SAS applications.

Note: The metadata server maintains its own copy of each ID but doesn't keep copies of passwords for identification purposes. As an alternative to using the following instructions, you can batch import users from a provider such as LDAP into the SAS metadata. See Appendix 2, "User Import Macros," on page 221. △

1 Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).

2 On the **Plug-ins** tab, select **User Manager** (in the foundation repository).

3 For each user:

   a Right-click and select **New ▶ User**.
   b On the **General** tab, enter a name.
   c On the **Accounts** tab, click **New**. In the New Login dialog box, select **DefaultAuth** and enter the user ID for the user's external account. In the standard configuration, this can be any type of account that is known to the metadata server's host (an LDAP, Active Directory, host, or other type of account).

   Note: For a Windows account, qualify the ID (for example, *WIN**myID* or *myID*@*mycompany.com*). △

Note: If your site uses Web authentication, you might create some logins in a different authentication domain. See "Logins for Users Who Participate in Web Authentication" on page 166. △

Note: In a specialized configuration where you set **-authpd LDAP:** , you must append a suffix to each user ID that is authenticated by your LDAP provider. See "Direct LDAP Authentication" on page 146. △

d Click **OK** to save the new login (it is not necessary to include a password in this login). Click **OK** again to save the new user definition.

4 If the workspace server is on Windows, give anyone who accesses that server using credential-based host authentication the **Log on as a batch job** privilege. See "Windows Privileges" on page 36.

Here are some details and tips:

- These instructions create registered users who automatically belong to PUBLIC (everyone who can access the metadata server) and SASUSERS (those members of PUBLIC who have a well-formed user definition).
- A user who doesn't have a well-formed definition can still log on to most applications. However, the user has only the PUBLIC identity. In the standard configuration, a PUBLIC-only user can't access any resources.
- You don't have to make changes on a user's **Authorization** tab. This tab has no effect on what a user can do.

Open Up Access

The initial configuration provides sufficient access to data and resources, with these exceptions:

- Only unrestricted users can access data through information maps, reports that are based on information maps, the metadata LIBNAME engine, or the OLAP server. In the initial configuration, the only grants of the Read permission are in each user's personal content area (**My Folder**).
- Only unrestricted users and members of the SAS Administrators group can register cubes.

To ensure access to resources and data:

1 Log on to SAS Management Console as an administrator (for example, sasadm@saspw).

2 (Optional) Verify that all registered users have at least the minimum required repository-level access.

a On the **Plug-ins** tab, under **Authorization Manager**, expand the **Access Control Templates** node.

b Right-click the repository ACT  (Default ACT) and select **Properties**.

c On the **Permission Pattern** tab, select **SASUSERS**. Verify that the ReadMetadata and WriteMetadata permissions are granted.

3 (Optional) Verify that all registered users have basic access to the folder tree.

a On the **Folders** tab, right-click the root folder ( **SAS Folders**) and select **Properties**.

b On the folder's **Authorization** tab, select **SASUSERS**. Verify that the ReadMetadata permission is granted.

4 Provide metadata layer access to data (this is a broad approach).

a On the **Authorization** tab for the root folder (**SAS Folders**), select **SASUSERS**.

Note: To access this tab, select the **Folders** tab, right-click the root folder, and select **Properties**. △

b Grant the Read permission. This enables registered users to perform tasks such as querying cubes, accessing data through information maps, and viewing the contents of tables.

If you want to manage access to data more narrowly, set grants of the Read permission on specific folders for specific users. Users need the Read permission as follows:

- Users need Read permission on an information map in order to access data through that information map. For example, if Joe is denied Read permission on an information map, he can't view reports that are based on that information map.
- Users always need Read permission on OLAP data in order to access that data.
- Users sometimes need Read permission on relational data in order to access that data. Read permission is required when data is accessed using the metadata LIBNAME engine.

5 If users who aren't in the SAS Administrators group will register cubes, grant those users the WriteMetadata permission on the OLAP schema.

a On the **Folders** tab, expand the **Shared Data** and **SASApp - OLAP Schema** folders.

b Right-click the schema and select **Properties**.

c On the **Authorization** tab, select or add an identity and grant WriteMetadata permission to that identity. For example, to allow all registered users to add cubes, assign the grant of WriteMetadata permission to SASUSERS.

6 Verify that physical-layer access is available. These are the requirements:

- Anyone who accesses SAS data sets from a standard workspace server needs host operating system permissions to those files.
- Anyone who performs tasks that involve writing to a host directory needs host layer write access to that directory.
- The launch credential for the pooled workspace server, stored process server and, if applicable, workspace server that uses SAS token authentication needs physical-layer access to any SAS data that the server retrieves. Initially, the SAS Spawned Servers account (sassrv) is the launch credential for all of these servers.

Manage Access

Limit the Ability to Update or Delete Servers

In the initial configuration, the **Server Manager** capability is available to only the SAS Administrators group. This prevents other users from accessing server definitions under that plug-in. For greater security, use permissions to protect server definitions. See "Protecting Server Definitions" on page 85.

Establish Access Distinctions for Content and Data

In a new deployment, access to most resources and data is undifferentiated. All registered nonadministrators have identical metadata-layer access to content, data, and application features. Everyone who uses a stored process server or pooled workspace server has identical host-layer access to any SAS data that server retrieves. In a migrated deployment, access to most resources and data mirrors access in the original environment.

To manage access to items such as reports, stored processes, information maps, and data definitions, create custom folders that reflect the distinctions that you want to make. See Chapter 6, "Permissions on Folders," on page 71.

To fully protect SAS data sets, you must also address host access. See "Host Access to SAS Tables" on page 187.

Provide PUBLIC Access (Optional)

Note: This is not a widely applicable task. Before you use these instructions, verify that they are appropriate for your environment and goals. See "About PUBLIC Access and Anonymous Access" on page 140. △

If you need to set up a specialized configuration in which unregistered users can participate, make these changes in SAS Management Console:

1 On the **Plug-ins** tab, under **Authorization Manager ▸ Access Control Templates** right-click the repository ACT (default ACT) and select **Properties**. On the **Permission Pattern** tab, grant the ReadMetadata and WriteMetadata permissions to PUBLIC.

   Note: Even users who only consume content need both of these permissions at the repository level, because some applications write system information about user activity, even during what appears to be a view-only transaction. △

2 On the **Folders** tab, give the PUBLIC group the Read permission for any information maps, cubes, and MLE data that you want to make universally available. A good approach is to create a folder branch for such content, set the grant on the top folder in that branch, and allow the grant to flow through the branch.

   Note: Users also need ReadMetadata permission to folders and content items. In general, it isn't necessary to set specific grants because this permission must flow through from the repository ACT into the public areas of the folder tree (for navigational purposes). △

   Note: If you want to allow everyone (including unregistered users) to contribute content to a particular folder, give the PUBLIC group a grant of the WriteMemberMetadata permission on that folder's **Authorization** tab. △

3 On the **Plug-ins** tab, under **User Manager**, right-click the PUBLIC group and select **Properties**. Review the PUBLIC group's role memberships. Typically, no adjustments are necessary, because the initial role assignments give the PUBLIC group basic capabilities.

4 Make sure that the PUBLIC group can use servers.

   a On the **Plug-ins** tab, under **Server Manager**, verify that the PUBLIC group has appropriate permissions to the servers that unregistered users will access.
   b If necessary, add one or more logins on the PUBLIC group's **Accounts** tab (for example, to provide seamless access to a third-party DBMS).

c If you have configured client-side pooling, verify that PUBLIC is a designated puddle group.

5 On the **Plug-ins** tab, under **Application Management ▶ Configuration Manager ▶ Web Report Studio**, select the **Settings** tab and the **Application Configuration** section. Set the **Allow Public Users** property for SAS Web Report Studio to **yes**. This change takes effect after you restart the SAS Web Infrastructure Services application and then restart SAS Web Report Studio.

Note: Some applications don't accept PUBLIC-only users. △

Manage Passwords

Password Policies

Each authentication provider sets password policies for accounts in that provider. For example, the password expiration policy for a host account is determined by that host. For the SAS internal authentication provider, you can set server-level and per-account policies such as password strength requirements and password expiration periods. See "How to Change Internal Account Policies" on page 163.

In the initial configuration, users can choose to store their credentials in their client-side connection profiles. This prepopulates the logon dialog box in desktop applications. To prevent users from creating a local copy of their credentials, set SASSEC_LOCAL_PW_SAVE="N" (or ="0" or ="F") in the metadata server's omaconfig.xml file and restart the server.

In desktop clients, this option controls the availability of a check box that enables the user to choose whether to store their credentials locally.

Note: A change to the SASSEC_LOCAL_PW_SAVE= setting takes effect after the metadata server is restarted. Each client uses the previous setting for its first connection, discovers the revised metadata server setting, and conforms to that revised setting for subsequent connections. If you change the setting to disallow saved credentials, and credentials are already present in a user's connection profile, those credentials must be manually removed. △

Password Updates for Service Accounts

Passwords for a few service accounts require special coordination because these passwords are included in configuration files. The follow table and list provide details.

Table 2.3 Overview of Storage of Service Account Passwords

Owning Metadata Identity	Example Account ID	Location	
		In Files	In Metadata
SAS Administrator	sasadm@saspw	✓	✓
SAS Trusted User	sastrust@saspw	✓	✓

Owning Metadata Identity	Example Account ID	Location	
		In Files	In Metadata
SAS Anonymous Web User	webanon@saspw	✓	✓
SAS General Servers	sassrv		✓
LSF Services[1]	lsfuser		✓

1 For sites that use a standard configuration of scheduling with Platform Suite for SAS (with SAS Web Report Studio).

Here are some exceptions to the preceding table:

- Not all sites use all accounts.
- Not all sites use the standard account IDs.
- Some sites have additional service accounts (for example, additional logins on the SAS General Servers Group).
- Some sites choose to use external accounts for all service identities (instead of using internal accounts where appropriate).

To update a service account password that is included only in metadata, use either SAS Management Console or the SAS Deployment Manager. To update a password that is included in configuration files, use the SAS Deployment Manager. Here are some key points about using the SAS Deployment Manager to update passwords:

- The utility updates both configuration files and metadata. You can update multiple passwords in a single pass.
- You must run the utility on each machine that hosts affected components. If you have servers on multiple machines, run the utility on each host, beginning with the metadata server machine.
- It might be necessary to update the same password on multiple hosts. For example, if you update the password for the SAS Trusted User on the metadata server's host, you must also do the same update on the middle-tier machine.
- Be sure to supply the same new password for an account on all machines on which you update that account.
- If you enter a plaintext password into the utility, the utility encodes that password using SAS proprietary encoding (SAS002).
- Passwords for any service accounts that you introduce in SAS Management Console aren't managed by this tool. For example, if you designate a new login as the launch credential for a server, that launch credential isn't automatically added to the list of accounts that the SAS Deployment Manager can update. Server launch credentials aren't added to a configuration file, so you can update any such passwords from the owning identity's **`Accounts`** tab in SAS Management Console.
- Each run of this utility generates an UpdatePasswords.html file that documents the updates that the utility performed and provides instructions for any required post-update activities.

To update a password with SAS Deployment Manager:

1 (Optional) If you are updating the password for an internal account, review the server-level password policies for internal accounts. Also, check each internal account's properties to determine whether any more (or less) stringent requirements apply.

Note: In particular, make sure that the account is not subject to a forced password change after the password is reset (either set the password to never expire or change the server-level policy for pre-expired passwords). △

Note: By default policy, internal passwords must be at least six characters and don't have to include mixed case or numbers. The five most recent passwords for an account can't be reused for that account. △

2 (Optional) If you have licensed SAS/SECURE and you want to use stronger encryption than SAS002, use the PWENCODE procedure to prepare an AES-encrypted version of each new password. For example:

```
proc pwencode in='PWsassrv1' method=sas003;
run;
```

The encrypted password is written to your SAS log. When you use method=sas003, the first part of the password is {sas003}.

3 Stop all SAS servers and services. Make any necessary adjustments to the state of your third-party Web components, as explained in the following table:

Table 2.4 State of Web Components for a Password Update

Product	Component	State
WebSphere	dmgr (the IBM deployment manager server)	Running
	nodeagent (the IBM managed node server)	Running
	Web application servers (for example, SASServer1)	It doesn't matter
WebLogic	node manager	Running
	ManagedWebLogic server	Stopped
JBoss	Web application servers (for example, SASServer1)	Stopped

4 If you are updating the password for an external account (for example, sassrv), change that password in your external authentication provider (for example, in the host operating system).

5 Restart the metadata server. Do not restart other servers or services.

6 On the metadata server's host, navigate to your equivalent of *SAS-installation-directory*`/SASDeploymentManager/9.2` and launch config.exe (Windows), config.sh (UNIX), or config.rexx (z/OS).

7 In the SAS Deployment Manager, select the update passwords task, select a configuration directory on the current machine, and log on as an unrestricted user (for example, sasadm@saspw).

8 Perform the update. If you need detailed assistance with the user interface, see the Help within the utility.

9 If you have servers on multiple machines, repeat steps 6–8 on each server host as applicable for the accounts that you are updating. Remember that you might have to update the same account on multiple hosts.

Note: Not all accounts are used on all hosts. If the accounts that you are updating aren't on a particular host, proceed to the next host. △

10 Restart all servers and services, and complete any additional post-update tasks as specified in the generated UpdatePasswords.html file.

Note: Because of dependencies, it is important to start servers and services in a particular order. In particular, you should start the metadata server first and start

Remote Services (the SAS Services Application) before you start the Web servers. For a complete discussion, see the chapter "Operating Your Servers" in the *SAS Intelligence Platform: System Administration Guide*. △

Note: You can automate running the deployment manager when you need to perform the same configuration action on many machines in your deployment. The deployment manager uses the same record and playback mechanism as the SAS Deployment Wizard to perform a non-interactive, silent configuration. For more information, see the topic "Automating the SAS Installation on Multiple Machines" in the *SAS Intelligence Platform: Installation and Configuration Guide*. △

CAUTION:

If you choose to use the deployment manager's record and playback mechanism to update passwords, passwords are written to the response file. For greater security, delete the response file (or remove the passwords from the response file) when you are finished. A response file is present only if you use the record and playback mechanism, instead of completing the task manually as documented in the preceding steps. △

See Also

"Encryption Overview" on page 6

"SAS Internal Authentication" on page 150

"Passwords That Are Managed By the SAS Deployment Manager" on page 215

Password Updates for Users and Groups

External Accounts

In most cases, the SAS copy of an external account includes only a user ID and doesn't include a password, so no password updates in metadata are necessary.

If any external passwords are stored, updates to those passwords are driven by changes that first occur in the external authentication provider. For example, if a copy of the password for an Oracle account or a host account is stored in the metadata as a group login, you must maintain that copy so that it always matches the actual password. Any change to the actual password (in Oracle) must be followed by a corresponding update to the SAS copy of the password (in the group login in the metadata).

You can update external passwords in SAS Management Console. If you own logins that include passwords, you can also update those passwords in SAS Personal Login Manager. To update the SAS copy of an external password in SAS Management Console, navigate to the owning user or group definition, select the **Accounts** tab, select a login, and click **Edit** (next to the table of logins).

SAS Internal Accounts

Every SAS internal account has a password. By initial policy, these passwords don't expire. See "How to Change Internal Account Policies" on page 163.

To update a SAS internal password in SAS Management Console, navigate to the owning user definition, select the **Accounts** tab, and click **Update** (at the bottom of the tab). If you have your own SAS internal account, you can also update your internal password in SAS Personal Login Manager.

Note: If repeated attempts to log on with an internal account fail, that account might be locked. See "How to Unlock an Internal Account" on page 29. △

Ensure Availability of Application Features

In general, the initial configuration provides appropriate access to application features. Most nonadministrative capabilities are available to either PUBLIC (everyone who can access the metadata server) or SASUSERS (those members of PUBLIC who have a well-formed user definition). To ensure availability of application features:

1. Log on to SAS Management Console as an administrator (for example, sasadm@saspw).
2. (Optional) Verify current role memberships.
   a. On the **Plug-ins** tab, select **User Manager**.
   b. Clear the **Show Users** and **Show Groups** check boxes. The roles that exist in your deployment are displayed.
   c. Right-click any role, select **Properties**, and select the **Members** tab. The following table documents initial membership of the predefined roles in a new deployment. Not all deployments include all roles. For any unlisted roles, see the administrative documentation for the associated product or solution.

Table 2.5 Initial Role Memberships in a New Deployment

Role	Initial Members
Add-In for Microsoft Office: Advanced	PUBLIC
Add-In for Microsoft Office: Analysis	(Empty)[1]
Add-In for Microsoft Office: OLAP	(Empty)[1]
BI Dashboard: Administration	BI Dashboard Administrators
Enterprise Guide: Advanced	PUBLIC
Enterprise Guide: Analysis	(Empty)[1]
Enterprise Guide: OLAP	(Empty)[1]
Management Console: Advanced	SAS Administrators
Management Console: Content Management	SASUSERS
Metadata Server: Operation	SAS Administrators
Metadata Server: Unrestricted	SAS Administrator
Metadata Server: User Administration	SAS Administrators
Web Report Studio: Advanced	(Empty)[2]

Role	Initial Members
Web Report Studio: Report Creation	PUBLIC[2]
Web Report Studio: Report Viewing	PUBLIC[2]

1 In the initial configuration, the advanced role for this application provides a superset of this role's capabilities, so those capabilities are widely available even though this role has no members.

2 In a migrated deployment, membership in the SAS Web Report Studio roles is different because the migration maps the roles from the previous release to the new 9.2 roles. An exception is that the ability to manage distribution lists is not preserved. See step 5c below.

3 (Optional) Populate the analysis and OLAP roles for SAS Enterprise Guide and the SAS Add-In for Microsoft Office. Initially, only the advanced roles for these applications have members. This can create problems if you later choose to narrow the membership of these advanced roles. To ensure that you don't inadvertently disable the lower-level roles, make either SASUSERS or PUBLIC a member of the other roles for these applications.

4 Make BI Dashboard administrative capabilities available.

  a Right-click the **BI Dashboard: Administration** role, select **Properties**, and select the **Members** tab.

  b Move **SAS Administrators** (or other identities) to the **Current Members** list.

5 Make advanced SAS Web Report Studio features available.

  a Right-click the **Web Report Studio: Advanced** role, select **Properties**, and select the **Members** tab.

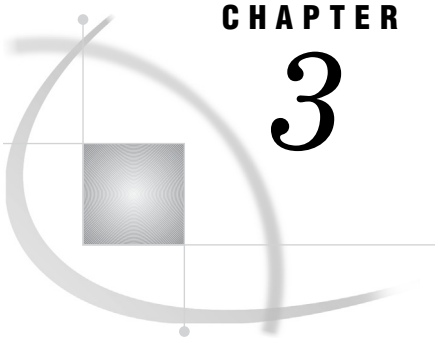
  b Move **SASUSERS** to the **Current Members** list.

  Note: This broad approach makes the menu items for features such as creating cascading prompts and report linking, scheduling, and distribution available to all registered users. A narrower approach is to instead make the SAS Administrators group a member of this role. △

  c On the **Capabilities** tab, notice that the **Manage Distribution List** capability isn't provided by this role. You can use any of these approaches to manage availability of this feature:

    - Do not select that check box. Only unrestricted users can use this feature.
    - Select this check box. All members of this role can use this feature.
    - Add this capability to a role that has limited membership (for example, the **Management Console: Advanced** role, which has the SAS Administrators group as a member).
    - Create a new role that offers only this capability. Make selected users or groups members of that role.

If you want to further adjust the initial role configuration, make changes on the **Members** tab of a role, make changes on the **Capabilities** tab of a role, or create a new role.



CHAPTER

3

Users, Groups, and Roles

About User Administration

In order to make access distinctions and track user activity, a security system must know who is making each request. The primary purpose of user administration is to provide information that helps systems identify each connecting user. In the SAS Intelligence Platform, the primary user management task is to create SAS copies of external account IDs. For identification purposes, only the account IDs are needed. SAS doesn't maintain copies of external passwords for identification purposes. For each user, you must create a SAS copy of a unique account ID, with these exceptions:

- For metadata administrators and some service identities, you can use a SAS internal account instead of a SAS copy of an external account ID.
- For users who don't need an individual identity, you don't have to store an account ID. Such users share the generic PUBLIC identity.

SAS uses its copy of these IDs to establish a unique SAS identity for each connecting user. All of a user's group memberships, role memberships, and permission assignments are ultimately tied to the user's SAS identity. See "How SAS Identity is Determined" on page 126.

To access user administration features in SAS Management Console, select the **User Manager** on the **Plug-ins** tab. If you don't want to use SAS Management Console to interactively create user definitions, you can write SAS programs that import and periodically update account IDs from your authentication provider. See Appendix 2, "User Import Macros," on page 221.

User Definitions

Each SAS user has identity information in two distinct realms:

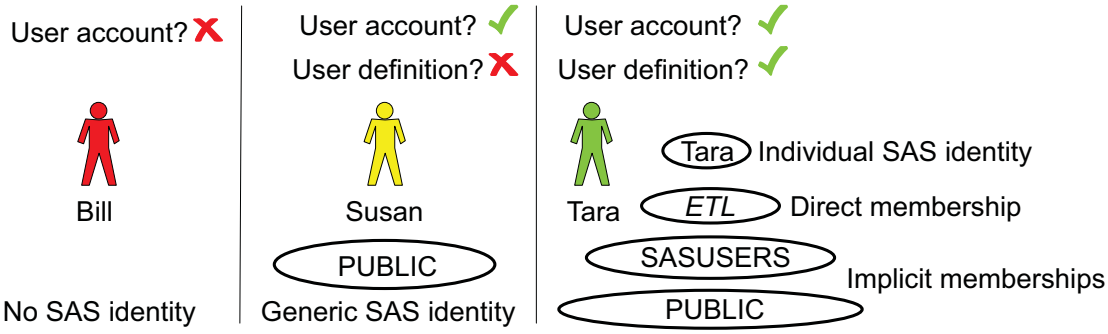
- In an authentication provider, the user has an account that can access the metadata server. See "Identify or Create User Accounts" on page 8.
- In the SAS environment, the user has a definition that includes a copy of the account ID with which the user accesses the metadata server.

Coordination between the two realms establishes SAS users. Every SAS user is based on a match between these values:

- the account ID with which a user authenticates
- an account ID that is listed on that user's **Accounts** tab

The following figure depicts some examples:

Figure 3.1 Examples: User Accounts and User Definitions

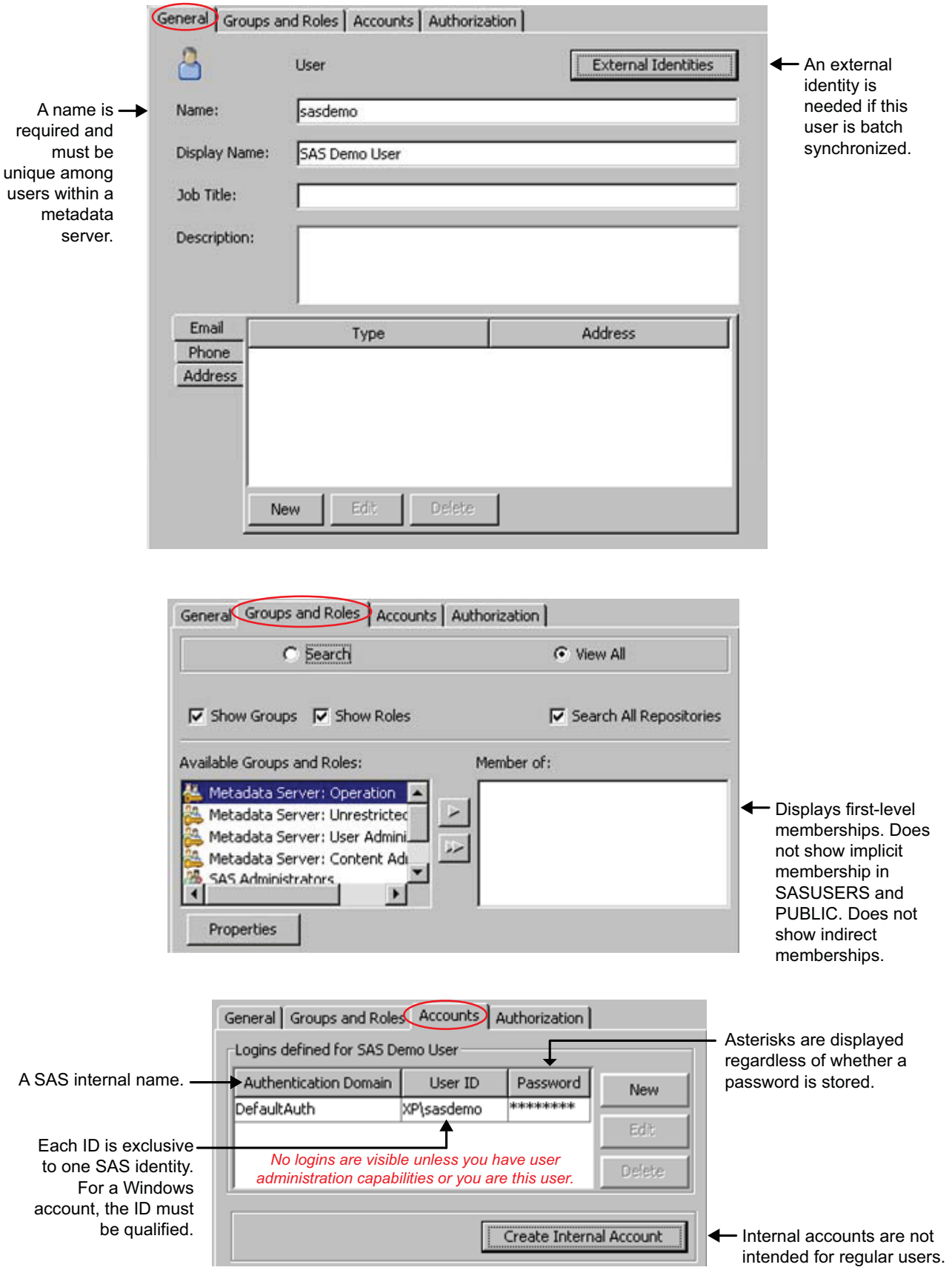


Note: In the preceding figure, the term "user account" refers to an account in an authentication provider. The term "user definition" refers to a metadata object that represents the user. Someone who doesn't have an account that provides access to the metadata server can't connect to the server at all. △

The following list and display provide details about user definitions:

- On the **Accounts** tab, any Windows user ID must be fully qualified (for example, **WindowsDomain\user-ID**, **MachineName\user-ID**, or **user-ID@company.com**).
- If you find that a user has only the PUBLIC identity even though the user has a user definition, examine the user's **Accounts** tab. The account ID might be missing, not accurately entered, or not properly qualified. Passwords and authentication domain assignments are never the cause of this problem. The match is based only on the account ID.
- Users can maintain their own logins (with SAS Personal Login Manager or SAS Management Console) but can't make other changes to their definitions.
- A user's **Authorization** tab does not determine what that user can do. This tab can affect the ability of other users to modify or delete this user.
- For instructions for adding users, see Chapter 2, "Security Tasks," on page 7.

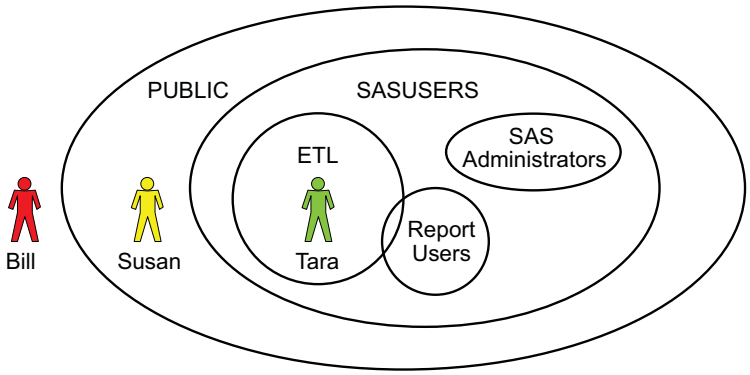
Figure 3.2 A User Definition



Group Definitions

Groups are primarily used in permission assignments. You can also use a group to populate a role or to make a shared credential available to multiple users. The following figure illustrates how the users in the previous topic might participate in a group structure:

Figure 3.3 Example: Users in a Group Structure



The following table introduces three important predefined groups:

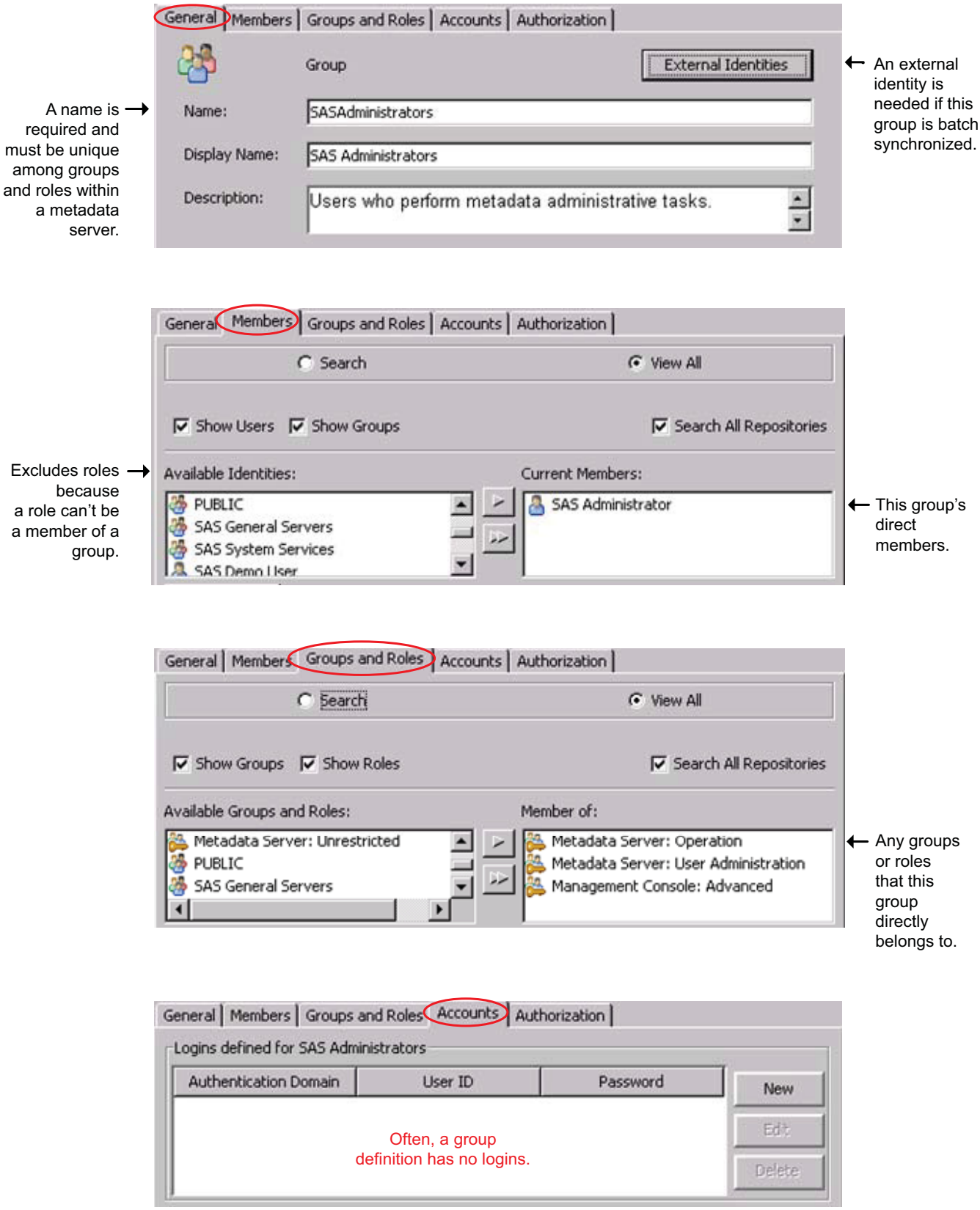
Table 3.1 PUBLIC, SASUSERS, SAS Administrators Groups

Group	Description
PUBLIC	A standard group with implicit membership. This group includes everyone who can access the metadata server, either directly or through a trust relationship. A user who does not have an individual identity uses the PUBLIC group identity.
SASUSERS	A standard group with implicit membership. This group includes those members of the PUBLIC group who have an individual identity. All members of the SASUSERS group are also members of the PUBLIC group.
SAS Administrators	A standard group for metadata administrators. In a standard configuration, members are granted broad access and administrative capabilities, but aren't unrestricted.

The following list and display highlight important details:

- The **`Groups and Roles`** tab enables you to create a nested group structure.
- The **`Accounts`** tab is often blank for a group. An entry on a group's **`Accounts`** tab makes a shared external account available to all members of the group. Such logins typically provide access to a third-party database server and should include both a user ID and a password. See "How to Store Passwords for a Third-Party Server" on page 174.
- A group's **`Authorization`** tab does not determine what that group can do. This tab can affect the ability of other users to modify or delete this group.

Figure 3.4 A Group Definition



Role Definitions

Roles manage the visibility of application features such as menu items, plug-ins, and buttons. In the initial configuration, registered users have almost all nonadministrative capabilities. If you want to alter the initial configuration, use either or both of these techniques:

- To increase or reduce the availability of a role, adjust a role's membership on its **Members** tab. For example, on the **Enterprise Guide: Advanced** role's **Members** tab, you might remove PUBLIC and add a few individual power users (or a custom group of power users). Any capabilities that are provided exclusively by the **Enterprise Guide: Advanced** role will no longer be available to other restricted users.

  Initially, the other SAS Enterprise Guide roles have no members, so narrowing the membership of this role limits the availability of all capabilities. To preserve the availability of the capabilities of the less privileged roles, assign a group (such as PUBLIC) as the member of those roles.

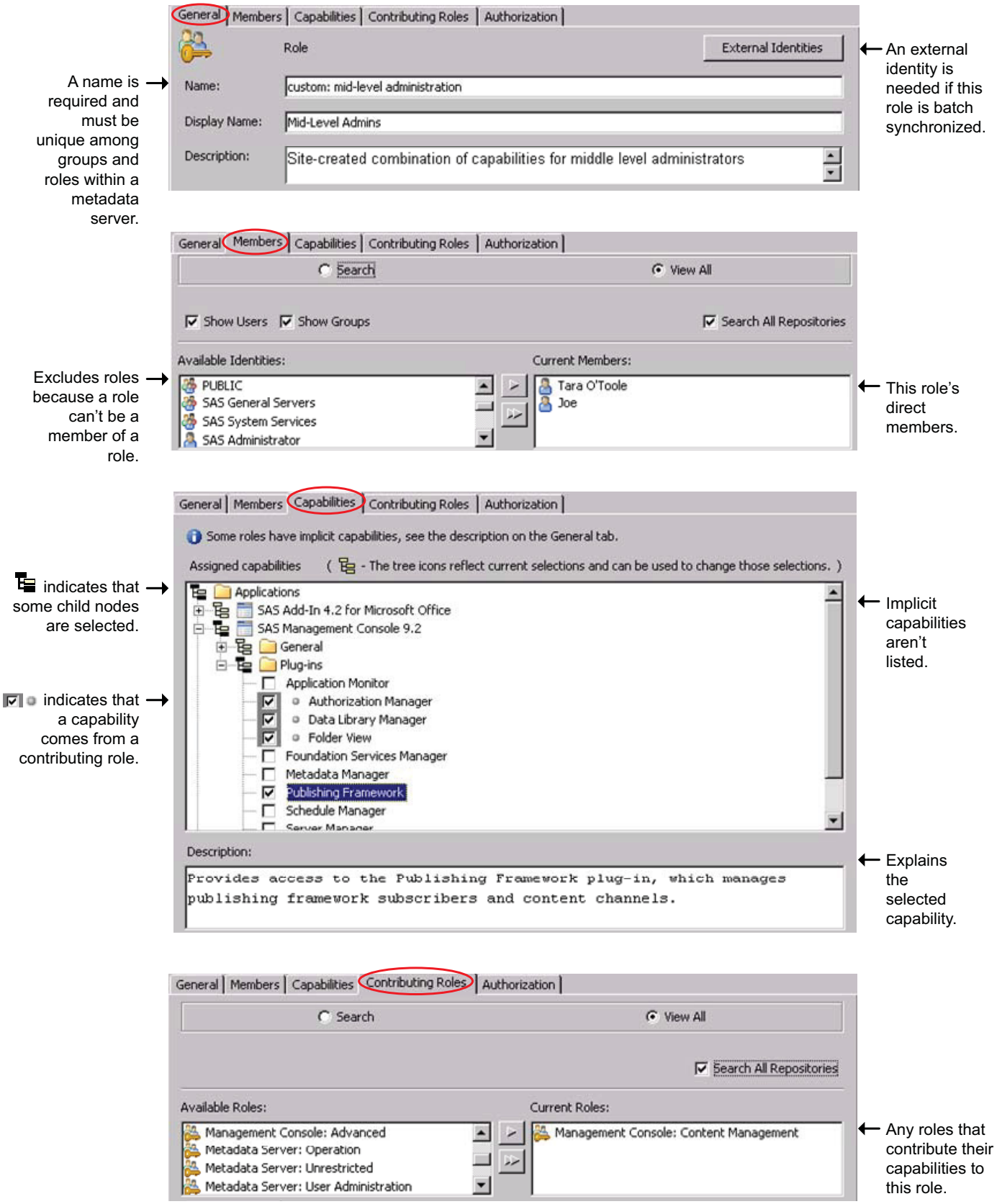
- To redistribute capabilities, change the selections on a role's **Capabilities** tab or assign contributing roles on a role's **Contributing Roles** tab. For example, on the **Management Console: Content Management** role's **Capabilities** tab, you might clear the **User Manager** check box. Only users who get this capability from another role (such as **Management Console: Advanced**) or are unrestricted will be able to see this plug-in.

  Initially, some capability assignments are redundant across an application's roles. For example, SAS Web Report Studio's basic print capability is provided by all of that application's predefined roles. To prevent someone from having a capability, make sure they aren't in any role that provides that capability.

The following list and display highlight important details:

- Not all application features are under role management. Not all applications have roles.
- Roles and groups serve distinct purposes. You can't assign permissions to a role or capabilities to a group.
- Don't change the **Name** of any of the predefined roles.
- The **Members** tab doesn't show indirect memberships. For example, the SAS Administrator is a member of the **Metadata Server: Operation** role, but the SAS Administrator is not listed on that role's **Members** tab. Only the direct member (SAS Administrators) is listed.
- The metadata server roles have implicit capabilities. For example, members of the **Metadata Server: User Administration** role can create new users, but there is no **Create Users** check box on any **Capabilities** tab.
- The relationships that you create using the **Contributing Roles** tab are monolithic; you can't deselect a contributed capability. These relationships are also dynamic; a change to the capabilities of one role affects any roles to which the first role contributes its capabilities.
- There are no negative capabilities (capabilities that limit what someone can do). You can't deny a capability to anyone.
- A role's **Authorization** tab does not determine what that role can do. This tab can affect the ability of other users to modify or delete this role.

Figure 3.5 A Role Definition



Main Administrative Roles

Table 3.2 Main Administrative Roles

Role	Capabilities	Initial Membership
Metadata Server: Unrestricted	Members have all capabilities and can't be denied any permissions in the metadata environment.[1]	SAS Administrator
Metadata Server: User Administration	Members can create, update, and delete users, groups, roles (other than the unrestricted role), internal accounts, logins, and authentication domains.[2]	SAS Administrators
Metadata Server: Operation	Members can administer the metadata server (monitor, stop, pause, resume, quiesce) and its repositories (add, initialize, register, unregister, delete).[3]	SAS Administrators
Management Console: Advanced	Members can see all plug-ins in SAS Management Console (in the initial configuration).	SAS Administrators

1 Unrestricted users can use only those logins that are assigned to them (or to groups to which they belong). They don't automatically have implicit capabilities that are provided by components other than the metadata server.

2 Restricted user administrators can't update identities for which they have an explicit or ACT denial of WriteMetadata.

3 Only someone who has an external user ID that is listed in the adminUsers.txt file with a preceding asterisk can delete, unregister, add, or initialize a foundation repository. Only an unrestricted user can analyze and repair metadata or perform tasks when the metadata server is paused for administration.

Here are some details:

- Many of the preceding tasks have permission requirements in addition to capability requirements. In a standard configuration, the SAS Administrators group has the necessary permissions.
- To operate servers other than the metadata server, you need the Administer permission, not a particular role or capability.
- The metadata server's roles have implicit capabilities. Implicit capabilities aren't listed on any **`Capabilities`** tab.
- You can't deselect capabilities for the unrestricted role.
- The metadata server's adminUsers.txt file provides many of the same privileges that it did in previous releases. However, we recommend that you use roles instead, except as specified in documentation for a particular task.
- The method that most applications use to retrieve credentials supports normal use of stored credentials, regardless of role memberships. However, if someone who has user administration capabilities makes a raw metadata request for logins, no usable passwords are returned.
- Do not give user administration capabilities to the identity that the object spawner uses to retrieve server launch credentials from the metadata. In a typical configuration, the spawner uses the SAS Trusted User to retrieve server launch credentials (through a raw metadata request). If the SAS Trusted User is a member of the user administration role (or the unrestricted role), the spawner will not operate properly.

Differences Between Roles and Groups

Roles and groups serve distinct purposes. You can't assign permissions to a role or capabilities to a group. Here are some additional distinctions:

- The identity hierarchy is relevant for groups, but not for roles. If you are a member of a role, you have all of that role's capabilities, regardless of whether you are a direct member of that role and what your other memberships are.
- You can deny a permission to a group, but you can't deny a capability to a role. Each role either provides or doesn't provide each capability. No role takes capabilities away from its members.
- A group's permissions are not displayed as part of a group definition, but a role's capabilities are displayed as part of a role definition.
- A group can be a member of another group, but a role cannot be a member of another role. Instead, one role can contribute its capabilities to another role.

Relationship Between Capabilities and Permissions

Having a certain capability is not an alternative to meeting permission requirements. Permissions requirements and capability requirements are cumulative. For example, even if you are in the **Enterprise Guide: OLAP** role, you can't access data in a cube for which you don't have the Read and ReadMetadata permissions.

Use roles and permissions in conjunction with one another. You can use permissions to constrain the scope of a role. For example, to allow someone to create reports but add them to only one folder, give the person the **Web Report Studio: Report Creation** role, but grant them the WriteMemberMetadata permission on only that one folder.

How to Create a Dual User

To enable someone to alternately function as an administrator and as a nonadministrator, create two user definitions for that person as follows:

- One definition is based on an internal account and is a member of the SAS Administrators group. See "Add Administrators" on page 9.
- The other definition is based on an external account and is not a member of the SAS Administrators group. See "Add Regular Users" on page 10.

Dual users log on with their internal account when they need administrative privileges and with their external account the rest of the time.

Note: The only way to make someone a dual user is to give that person two user definitions, each based on a different account. You can't create a dual user by adding a login to a definition that already has an internal account or by adding two logins to one definition. △

Note: Dual users should use a dedicated client-side connection profile for their internal account. In that profile, the user should leave the **Authentication Domain** field blank. This optimizes credential reuse. △

How to Unlock an Internal Account

By initial policy, making three consecutive unsuccessful attempts to log on with a SAS internal account causes that account to be locked for one hour. This topic explains

how to unlock the account immediately, so that you don't have to wait until the account lockout period has passed.

The preferred approach is to locate another user who has user administration capabilities. That user can unlock the internal account by completing these steps:

1. Log on to SAS Management Console as someone who is unrestricted or who has user administration capabilities.
2. On the **Plug-ins** tab, select **User Manager**.
3. In the display pane, clear the **Show Groups** and **Show Roles** check boxes. Right-click the user definition of the person whose SAS internal account is locked out and select **Properties**.
4. Select the **Accounts** tab. A message box asks whether you want to unlock the account. Click **Yes**.
5. In the user's **Properties** dialog box, click **OK**. The account is now unlocked.

   Note: It isn't necessary to reset the user's internal password as part of unlocking the user's internal account. △

If there is no other administrator available and you have the necessary host access, you can use this approach as a last resort:

1. Edit the metadata server's adminUsers.txt file to create a new unrestricted user.
   a. Navigate to your equivalent of **SAS/Lev1/SASMeta/MetadataServer/adminUsers.txt**. Open the file with a text editor.
   b. Add a user ID for an account that is known to the metadata server's host. Include a preceding asterisk (for example, ***WIN\winID** or ***unixID**).
   c. Stop and restart the metadata server to make the change take effect.

   CAUTION:
   Stopping the metadata server stops other components. △

2. Log on to SAS Management Console using the user ID that you added to the adminUsers.txt file. In the status bar at the bottom of the application window, notice that you are unrestricted. Unlock the locked account (see the preceding instruction list).
3. To verify that the account is unlocked, log on to SAS Management Console using the account.
4. Open the adminUsers.txt file, remove the entry that you added, and stop and restart the metadata server.

Here are some additional tips:

- If you choose to change the password for the original SAS Administrator, you might need to update the deployment. See "Password Updates for Service Accounts" on page 14.
- You can customize the account lockout policy. See "How to Change Internal Account Policies" on page 163.
- We recommend that you establish individual metadata administrators rather than sharing the predefined SAS Administrator account. See "Add Administrators" on page 9.

How to Assign Capabilities to Roles

To learn about customizing the distribution of capabilities across roles, complete this exercise in SAS Management Console.

1 Log on as someone who has user administration capabilities and is a member of the SAS Administrators group (for example, sasadm@saspw).

2 On the **Plug-ins** tab, select **User Manager** (make sure you are in the foundation repository). In the display area, clear the **Show Users** and **Show Groups** check boxes. The roles that exist in your deployment are displayed.

3 Right-click **User Manager** and select **New ▸ Role**. On the **General** tab, enter **Test Role** in the **Name** field.

Note: Creating a new role isolates this exercise from the rest of your deployment, ensuring that your current configuration is preserved. △

4 To learn how to directly assign capabilities, select the **Capabilities** tab:

a Notice that a message at the top of the tab reminds you that a few capabilities (for example, those of the metadata server's roles) aren't listed on this tab (because those capabilities are implicit).

b Notice that the first node (**Applications**) has an empty branch icon  . This indicates that no explicit capabilities are assigned to this role.

c Notice that there is a second-level node for each component that provides explicit capabilities. A role can provide capabilities from multiple applications.

d Click **+** to expand the **SAS Management Console** node. Click **+** to expand the **Plug-ins** node. Select the **Authorization Manager** check box. Notice that the branch icons are now partial  . This indicates that some of the capabilities are selected.

Note: To see a description of any capability, click that capability's text and look at the **Description** field at the bottom of the tab. △

e Click the partial icon  for the **Plug-ins** folder. This action causes all of the capabilities beneath that node to be selected. Click again to cycle back to the empty branch icon (no capabilities assigned). Click a third time to revert to the immediately preceding state (only the **Authorization Manager** check box selected).

f Click the **Authorization Manager** check box to clear it.

5 To learn how to indirectly assign capabilities, select the **Contributing Roles** tab:

a In the **Available Roles** list, select **Management Console: Content Management**. Before you make this a contributing role, verify its capabilities.

i Click **Properties**, select the candidate role's **Capabilities** tab, and expand **SAS Management Console ▸ Plug-ins**. Notice that four capabilities are selected. All check boxes are disabled because this dialog box is in read-only mode when it is accessed in this manner.

ii Check the role's description on the **General** tab to determine whether the role provides any further capabilities (implicit capabilities).

iii Check the role's **Contributing Roles** tab (just in case it has a contributing role that provides implicit capabilities).

iv Click **Cancel** to return to the **Contributing Roles** tab of your test role.

b Move the **Management Console: Content Management** role to the **Current Roles** list. This role now contributes all of its capabilities to your new role. If capabilities of this contributing role change, the capabilities of your test role change also.

It is necessary to use contributing roles in these circumstances:

□ You want to extend implicit capabilities (like the capabilities of the metadata server roles) to other roles.

- You want to provide dynamic aggregation of roles so that changes to one role propagate to other roles that have the first role as a contributing role.

6 To learn about interactions between contributed and directly assigned capabilities, select your test role's **Capabilities** tab again.

   a Under **SAS Management Console ▶ Plug-ins**, notice that capabilities from the **Management Console: Content Management** role are now selected. A visual indicator identifies these as contributed capabilities.
   b Select the already-selected **Authorization Manager** check box. This adds a direct assignment on top of the contributed assignment, making the assignment independent from the underlying contributing role.
   c Click the tree icon for the **Plug-ins** folder three times (stop when only the **Authorization Manager** check box is explicitly selected).
   d Select the **Authorization Manager** check box again. It reverts back to the contributed state. You can't incrementally remove a contributed capability.
   e If you want to make your role independent of the **Management Console: Content Management** role:

      i Add explicit selections on top of all contributed capabilities.
      ii Select the **Contributed Roles** tab. Move the **Management Console: Content Management** role back to the **Available roles** list box.
      iii Select the **Capabilities** tab. Notice that you can now clear any capability.

      Note: This technique (trace a contributing role and then remove it) is a good way to create a new role that is based on another role but offers fewer capabilities. Of course, anyone who has both roles has any capability that is offered by either role. △

7 To close the dialog box (and not save the test role), click **Cancel**.

User ID Formats

In most cases, users can launch SAS applications using the same ID and password as they use in the rest of your computing environment. However, when you create a SAS copy of a Windows user ID, you must qualify the ID (for example, **WindowsDomain\user-ID**, **MachineName\user-ID**, or **user-ID@company.com**).

Failure to meet the preceding requirement doesn't prevent a successful logon. However, it prevents SAS from recognizing the user's individual identity and causes the user to have only the PUBLIC identity. See "Authentication to the Metadata Server" on page 126.

If your site accepts Windows IDs in disparate formats, you must coordinate the format of the copies with the format in which users submit their IDs. This table describes the common forms for an Active Directory user ID:

Table 3.3 Overview: Forms of an Active Directory User ID

Form[1]	Basic Syntax	Examples
Short	*user-ID*	**joe**
UPN	*user-ID@UPNsuffix*	**joe@orionsports.com** or **joe@sales.orionsports.com**

Form[1]	Basic Syntax	Examples
Down-level	*down-level-domain-name**user-ID*	**orionsports\joe** or **sales\joe** or **mymachine\joe**
Kerberos	*user-ID*@*realm*	**joe@orionsports.com**[2]

1 User Principal Name (UPN) is an Active Directory concept. Down-level domain is a Windows NT concept.

2 The realm in a Kerberos name is usually a Windows domain. A Kerberos name can include an instance (in the format *user-ID*/*instance*@*realm*). Additional site-specific variations might occur.

In the SAS Intelligence Platform, follow these standards for Windows user IDs:

- If users log on interactively, they can use the short form except in these unusual cases:
  - The user needs to host authenticate to a metadata server that has been configured to directly use a provider other than its host. See "Direct LDAP Authentication" on page 146.
  - The user has multiple accounts with the same user ID in different down-level domains (for example, **machine\joe**, **domain1\joe**, and **domain2\joe**).
- If users log on interactively, they can also use one other site-supported form (either the UPN form or the down-level form). Use one of these approaches:
  - In the metadata, store each user ID in UPN form. Tell users not to use the down-level form when they log on.
  - In the metadata, store each user ID in down-level form. Tell users to not use the UPN form when they log on.
- If users log on to SAS desktop applications through Integrated Windows authentication, their user IDs should usually be stored in down-level form. In general, that is the form in which SAS obtains user IDs after Kerberos authentication occurs.

  Note: If you prefer to store user IDs in the native Kerberos form, add the setting **SASUSEKERBNAME true** as a Windows system environment variable on the server host. For example, on the Windows desktop, right-click **My Computer**, select **Properties**, select the **Advanced** tab, click the **Environment Variables** button, add the setting under **System variables**, and reboot the machine. This setting affects only connections that use Integrated Windows authentication. If you use this setting, you might want to make sure that the Integrated Windows authentication process always chooses the Kerberos protocol. See "How to Force Use of Kerberos" on page 172. △
- If users log on to SAS Web applications through Integrated Windows authentication (which occurs only if you configure Web authentication and have set up Integrated Windows authentication with your Web provider), the form of the returned user ID might differ. See the documentation for your Web application server.

Note: In the status bar of applications such as SAS Management Console, a currently connected Windows user ID is always displayed in the format **user-ID@VALUE**, regardless of how the user logged on or how the user's ID is stored in the metadata. For example, if you log on as Joe and your stored user ID is WIN\joe, the status bar displays your authenticated ID as joe@WIN. △

Unique Names and IDs

The metadata server enforces these identity-related constraints:

- You can't create a user definition that has the same name as an existing user definition. The display names don't have to be unique.

  Note: We recommend that you avoid using spaces or special characters in the name of a user, group, or role that you create. Not all components support spaces and special characters in identity names. △

- You can't create a group or role definition that has the same name as an existing group or role definition. The display names don't have to be unique.
- You can't assign the same external account ID to two different identities. All of the logins that include a particular ID must be owned by the same identity. This requirement enables the metadata server to resolve each ID to a single identity.
  - This requirement is case-insensitive. For example, you can't assign a login with an ID of *smith* to one user and a login with an ID of *SMITH* to another user.
  - This requirement applies to the fully qualified form of the ID. For example, you can assign a login with an ID of *winDEV\brown* to one user and a login with an ID of *winPROD\brown* to another user. In this example, *winDEV* and *winPROD* are Windows domain names, which are incorporated into the fully qualified form of an external account ID.
  - This requirement can't be mitigated by associating the logins with different SAS authentication domains. For example, assume that one user has a login with an ID of *smith* that is associated with a SAS authentication domain named *DefaultAuth*. In that case, you can't give any other user a login with the ID *smith*, even if you plan to assign the login to a different SAS authentication domain.

  Note: To enable multiple users to share an account, store the credentials for that account in a login as part of a group definition. Then add the users who will share the account as members of that group definition. △
- If you give a user two logins that contain the same ID, the logins must be associated with different authentication domains. Within an authentication domain, each ID must be unique. For example, if you give the person *Tara O'Toole* two logins that both have an ID of *tara*, then you can't associate both of those logins with the *OraAuth* authentication domain.

  Note: Like the previous requirement, this requirement is case-insensitive and is applied to the fully qualified form of the external account ID. △
- A user can have multiple locations, e-mail addresses, and telephone numbers. However, each user can have only one item of a given type. For example, a user can have one **`home`** e-mail address and one **`work`** e-mail address, but not two **`work`** e-mail addresses.

Identity Precedence

Identity precedence affects authorization decisions when a user has more than one relevant permission setting because of the user's group memberships. Identity precedence affects login priority when someone has more than one login in an authentication domain. Identity precedence is not relevant for roles.

This is the precedence ranking for users and groups:

1. the user's individual identity, based on the user's authenticated ID.
2. a user-defined group that has user as a member. This is a first-level group membership for the user.

3 a user-defined group that has another user group as a member. For example, assume that the user belongs to a group named ETL_Advanced, and that group is a member of another group called ETL_Basic. In that case, the ETL_Basic group is a second-level group for that user. If you have additional levels of nesting, each successive level has less precedence.

4 the SASUSERS implicit group, which includes everyone who has an individual identity.

5 the PUBLIC implicit group, which includes everyone who can access the metadata server (regardless of whether they have an individual identity or not).

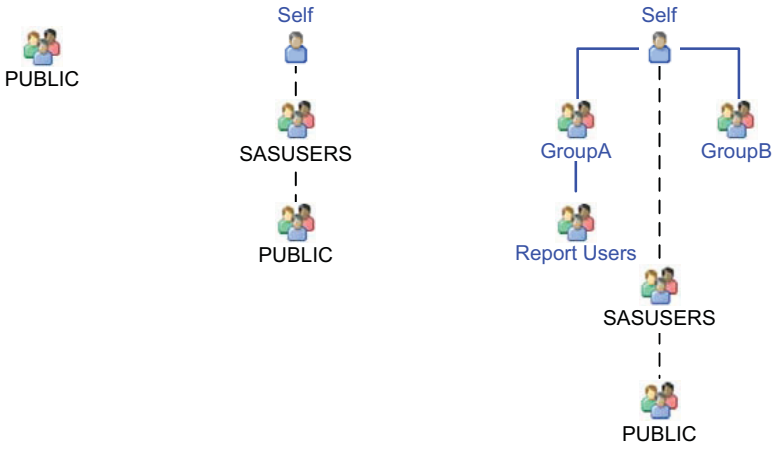
The examples in the following table illustrate the hierarchy:

Table 3.4 Examples of Identity Hierarchies

User Information	User's Identity Hierarchy
User has no individual identity.	Primary identity: PUBLIC
User has an identity and no explicit group memberships.	Primary identity: self First-level memberships: SASUSERS Second-level memberships: PUBLIC
User is a direct member of two user-defined groups (GroupA and GroupB), and one of those groups is a member of a third group (GroupA is a member of the Report Users group).	Primary identity: self First-level memberships: GroupA, GroupB Second-level memberships: Report Users Third-level memberships: SASUSERS Fourth-level memberships: PUBLIC

The following figure illustrates the examples from the preceding table:

Figure 3.6 Examples of Identity Hierarchies



Note: To avoid introducing unnecessary complexity, don't make PUBLIC or SASUSERS a member of another group. For example, if you make PUBLIC a member of GroupA, then a user who is an indirect member of GroupA (through his automatic membership in PUBLIC) has GroupA as his lowest precedence membership. This contradicts the usual expectation that every user's lowest precedence membership is PUBLIC. This is not an issue for roles. △

Windows Privileges

Table 3.5 Access this computer from the network

Description	This privilege is required in order to connect to SAS servers.
To Whom	Give this privilege to all users who access SAS servers on Windows.
How	Typically, this right is already granted to the Windows group **`Everyone`**. To confirm, check the Windows local policy settings.

Table 3.6 Log on as a batch job

Description	This privilege is required in order to run a stored process server or any type of workspace server.
To Whom	On the Windows computer that hosts the SAS object spawner, give this privilege to the accounts under which workspace servers and stored process servers run: □ any service account under which one of these servers run □ all puddle logins for any client-side pooled workspace servers □ any user accounts under which a standard workspace server runs[1]
How	Modify the local security policy. For example, on Windows XP, this right is managed from the Windows control panel under Administrative Tools ▶ Local Security Policy ▶ User Rights Assignment ▶ Log on as a batch job. If you have an operating system group (such as SAS Server Users) that has this right, you just add users and service account identities to that group.

1 Users who authenticate to the standard workspace server by Integrated Windows authentication or SAS token authentication don't need this privilege.

Table 3.7 Trusted for delegation

Description	This privilege enables a process to allow each user's credentials to be sent to further machines for authentication (for example, to access a UNC path). The privilege is needed if the workspace server is accessed through Integrated Windows authentication and provides access to Windows network resources.[1]
To Whom	Give this privilege to the account under which the object spawner runs. By default, the spawner runs as a service under the local system account, so the computer account for spawner's host needs the privilege.
How	As a Windows domain administrator, under Start ► Control Panel ► Administrative Tools ► Active Directory Users and Computers, access the properties dialog box for the relevant account and grant the privilege. For example, if the spawner runs under the local system account, select the spawner host machine under **`Computers`**. On the **`Delegation`** tab (or the **`General`** tab), select the **`Trust this computer for delegation`** check box. Or, if the spawner runs under a service account, select that account under **`Users`**. On the **`Delegation`** tab (or the **`Accounts`** tab), select the **`Account is trusted for delegation`** check box. This setting is available only for service accounts that have registered service principal names.

1 With Integrated Windows authentication, the workspace server does not receive the requesting user's credentials, so the workspace server cannot provide credentials for downstream servers. Instead, the spawner account must be trusted to delegate each requesting user's identity as necessary.

Note: In most cases, an object spawner on Windows runs as a service under the local system account account. If the spawner instead runs under some other account, that account must be a Windows administrator on the spawner's host and have the Windows user rights **`Adjust memory quotas for a process`** and **`Replace a process level token`**. These user rights assignments are part of the local security policy for the Windows computer that hosts the spawner. △

Who Can Manage Users, Groups, and Roles?

Your roles and permissions determine which user management tasks you can perform. In a standard configuration, any member of the SAS Administrators group has the capabilities and permissions to perform almost all user management tasks.

The following table provides basic information:

Table 3.8 Who Can Manage Users, Groups and Roles?

Metadata Server Role	Actions Supported
Unrestricted	Perform all identity management tasks.
User administration	Add, modify, and delete most identities.[1]
None	Update your personal logins in SAS Personal Login Manager.[2]

1 Permission requirements apply. The User Manager capability (which enables you to see the **`User Manager`** plug-in in SAS Management Console) is also required.

2 If you have the User Manager capability, you can perform this task in SAS Management Console.

Here are some additional details:

- A special rule prevents restricted user administrators from updating the unrestricted role.

- In order to change a role's capabilities, restricted user administrators must also have the WriteMetadata permission on the associated software component. In the standard configuration, the SAS Administrators group has this grant.
- To prevent a restricted user administrator from updating a particular identity, deny that user administrator the WriteMetadata permission on that identity's **`Authorization`** tab.
- To delegate management of an existing identity to someone who isn't a user administrator, grant the WriteMetadata permission to the delegated administrator on the target identity's **`Authorization`** tab.
- In the initial configuration in a new deployment, all registered users have the User Manager capability through SASUSERS membership in the **`Management Console: Content Management`** role.

CHAPTER

4

Getting Started With Permissions

Orientation to Working With Permissions

When you work with permissions, it is helpful to keep these three central questions in mind:

- Where is the permission set?
- Who is the permission assigned to?
- What is the effect of the permission?

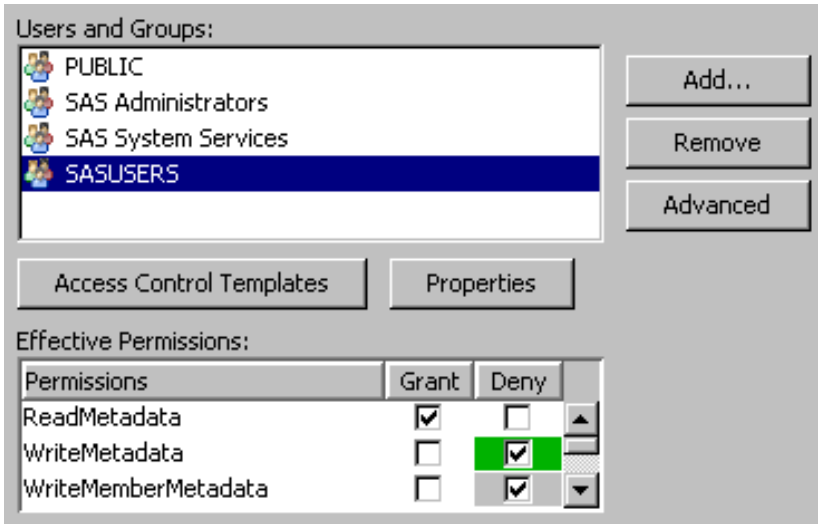
The authorization layer that SAS provides supports very specific settings. For example, *on reportA, give userA a grant of the ReadMetadata permission*. For simplicity, it is preferable to set permissions at a less granular level whenever possible. The recommended approach is to create a folder structure and set permissions so that each folder offers appropriate group-level access to its contents. For example, *on folderA, give groupA a grant of the ReadMetadata permission*. Items within each folder inherit permission settings from that folder.

This chapter demonstrates the mechanics of setting SAS permissions by providing brief exercises that you can safely experiment with in your own deployment. This chapter also introduces the authorization model.

The Authorization Tab

Where Are Permissions Set?

You can set permissions on a folder, a report, an information map, a server definition, a cube definition, a stored process definition, or almost any other metadata item. Each item's permission settings are displayed on that item's **Authorization** tab. To view or set permissions for any item, right-click the item, select **Properties**, and select the item's **Authorization** tab.



On an item's **Authorization** tab, you can set permissions individually by selecting check boxes in the **Effective Permissions** list. These are called explicit settings. You can also set permissions in patterns by clicking the **Access Control Templates** button. These are called ACT settings.

Note: For specialized tasks such as setting repository-level permissions and defining access to subsets of data, see "Granularity and Mechanics" on page 52. △

Who are Permissions Assigned To?

You can assign permissions to individual users or to user groups. Each SAS user has an identity hierarchy that starts with the user's individual SAS identity, can include multiple levels of nested group memberships, and ends with automatic membership in SASUSERS and then PUBLIC. For a depiction, see "Identity Precedence" on page 34.

On an item's **Authorization** tab, the **Users and Groups** list usually includes at least the following groups:

PUBLIC
: automatically includes everyone who can access the metadata server. This is the broadest group.

SASUSERS
: automatically includes those members of PUBLIC who have a well-formed user definition. This is a broad group that represents all registered users.

SAS Administrators
includes metadata administrators. This is a small, highly privileged group.

SAS System Services
includes one or more service identities. This group shouldn't have regular users as members. Usually, the SAS Trusted User is the only member.

Someone who isn't listed on an item's **Authorization** tab has the access of their closest listed group. Each user's closest listed group is determined by that user's group memberships and identity hierarchy. Here are some examples:

- The closest listed group for an administrator is usually SAS Administrators.
- The closest listed group for a regular registered user is often SASUSERS.
- The closest (and only) listed group for an unregistered user is PUBLIC.

To create specialized settings, click **Add** and add users or groups to the list. Or, click **Access Control Templates** and apply a predefined pattern of settings.

The **Advanced** button is available only if you are unrestricted. Use this button to trace an item's inheritance parents or to look up settings for any user.

What Do the Colors Indicate?

The following table explains the significance of the check box colors:

Table 4.1 Significance of Color in the Effective Permissions List

Color	Term	Significance
☑ (clear)[1]	Explicit	The permission is set on the current item and individually assigned to the selected identity.
☑ (green)	ACT	The permission comes from an applied ACT whose pattern explicitly assigns the grant or denial to the selected identity.
☑ (gray)	Indirect	The permission comes from someone else (a group that has an explicit or ACT setting) or somewhere else (a parent item or the repository ACT).[2]

1 Explicit settings are usually white because the background color for the permissions list is usually white.
2 For the WriteMemberMetadata permission, gray can indicate that the setting mirrors the WriteMetadata setting. For an unrestricted user, gray indicates a grant that can't be removed.

What Is the Effect of a Permission Setting?

On each item's **Authorization** tab, the permissions list always includes at least the following basic permissions:

- the ReadMetadata permission (controls the ability to see an item)
- the WriteMetadata permission (controls the ability to update or delete an item)

Other permissions are specialized and affect only certain types of items. For example, the ability to delete most items is controlled by the WriteMetadata permission, not by the Delete permission. For details, see "Use and Enforcement of Each Permission" on page 53.

The effect of a particular permission setting is influenced by any related settings that have higher precedence. For example, if a report inherits a grant from its parent folder

but also has an explicit denial, the inherited grant has lower precedence. The explicit setting determines the outcome, so the result is a denial.

On each item's **Authorization** tab, the check marks that are displayed in the **Effective Permissions** list incorporate all precedence considerations. The displayed effective permissions are a calculation of the net impact of all applicable permission settings in the metadata layer. However, the **Authorization** tab doesn't reflect access in other layers such as the operating system.

Explicit Settings

A permission that is assigned to a particular user or group is an explicit setting for that user or group. To experiment with explicit settings, complete this exercise in SAS Management Console:

1 Log on as someone who has a well-formed user definition.

2 On the **Folders** tab, right-click your **My Folder**  and select **New Folder**. Create a new folder named **test**.

3 Right-click the **test** folder and select **Properties**. On the **test** folder's **Authorization** tab, briefly examine the settings for each identity in the **Users and Groups** list box. Notice that all of the settings are indirect  . These settings come from the *test* folder's parent folder.

Note: You can't remove anyone, because all of the listed identities participate in settings that are defined elsewhere. △

4 To give the SASUSERS group an explicit setting:

a In the **Users and Groups** list box on the **test** folder's **Authorization** tab, select **SASUSERS**. Notice that SASUSERS has an indirect  denial of the ReadMetadata permission.

Note: These instructions assume that your **My Folder**  has standard settings. If this setting is not present, select another identity (such as PUBLIC) that does have an indirect denial of ReadMetadata. △

b Select the opposing check box (grant ReadMetadata). This gives the SASUSERS group an explicit  grant of ReadMetadata permission on the **test** folder.

c Select the grant ReadMetadata check box again. This removes the explicit grant and reveals the underlying indirect denial.

d Select the (already selected) deny ReadMetadata check box. This adds an explicit  denial on top of the indirect  denial.

e Click **OK**. An error message tells you that you can't save these settings. The only explicit setting on the **test** folder is the denial of ReadMetadata permission for SASUSERS. This denial blocks access for all registered users, including you. Click **OK** to close the message box and return to the **Authorization** tab.

Note: If you are unrestricted, you won't see the error message. Go to step 5. △

f To see the impact that the SASUSERS denial has on you, select yourself in the **Users and Groups** list box on the **test** folder's **Authorization** tab. Notice that your previous indirect grant of ReadMetadata permission is now an indirect denial of ReadMetadata permission.

g To restore access for yourself, select the grant ReadMetadata check box. This gives you an explicit grant that offsets the SASUSERS explicit denial. Click **OK**.

Note: An offsetting grant doesn't have to be assigned directly to you; it can be assigned to any group that is closer to you than the group that has the explicit denial. For example, your custom group memberships are closer to you than SASUSERS, and SASUSERS is closer to you than PUBLIC. △

5 To give an explicit setting to someone who is not already listed:

a On the **test** folder's **Authorization** tab, click **Add**. In the Add Users and Groups dialog box, clear the **Show Groups** check box. Move one user (such as the **SAS Demo User**) to the **Selected Identities** list box and click **OK**.

Note: In practice, it is preferable to assign permissions to groups rather than to individual users (for ease of management). △

b On the **Authorization** tab, notice that the user is selected and has an explicit ☑ grant of ReadMetadata permission. An explicit grant of ReadMetadata permission is automatically given to every restricted identity that you add.

Select the opposing check box, deny ReadMetadata permission. This replaces the explicit grant with an explicit denial.

Note: If the selected user has the unrestricted role, you can't change any settings. △

c Click **Remove** and then click **Yes** in the confirmation message box. You can remove this user because this user is named only in explicit settings.

Note: Regular users can't navigate to each other's **MyFolder** because of a denial of ReadMetadata permission to PUBLIC on a parent folder. △

6 To clean up, right-click the **test** folder and select **Delete**.

ACT Settings

Instead of setting every permission explicitly, use access control templates (ACTs). Each ACT consists of a pattern of grants and denials that are assigned to different users and groups. When you apply an ACT to an item, the ACT settings are added to the item's protections. When you want to assign the same settings to several disparate resources, using an ACT is beneficial for these reasons:

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

To learn more, complete this exercise in SAS Management Console:

1 Log on as someone who has a well-formed user definition.

2 On the **Folders** tab, right-click your **My Folder**  and select **New Folder**. Create a new folder named **test2**.

3 Right-click the **test2** folder and select **Properties**. On the folder's **Authorization** tab, briefly examine the settings for each identity in the **Users and Groups** list box. Notice that all of the settings are indirect ☑.

4 To apply an ACT to the **test2** folder:

a Click **Access Control Templates**. In the Add and Remove Access Control Templates dialog box, expand the **Foundation** node  in the **Available** list box and select **Private User Folder ACT**  .

b Before you apply this ACT to the **test2** folder, click **Properties** to verify the settings that this ACT provides. On the **Permission Pattern** tab, notice that this ACT provides denials of ReadMetadata, WriteMetadata, and CheckInMetadata permissions for the PUBLIC group, grants of these permissions for the SAS Administrators group, and a grant of ReadMetadata permission for the SAS System Services group.

Note: Each ACT's pattern consists of only the explicit ☑ settings on that ACT's **Permission Pattern** tab. Settings that are unspecified (blank) on an ACT's pattern have no effect when that ACT is applied to an item. △

Click **Cancel** to return to the list of ACTs that are applied to the **test2** folder.

c In the Add and Remove Access Control Templates dialog box, move **Private User Folder ACT**  to the **Currently Using** list box. This adds that ACT's settings to the access controls for the **test2** folder. Any future changes to this ACT's permission pattern will affect access to this folder.

Note: The **Currently Using** list box includes only applied ACTs; so this list typically does not include the repository ACT (default ACT). △

d Click **OK** to return to the **Authorization** tab. Notice that the PUBLIC denials of ReadMetadata, WriteMetadata, and CheckInMetadata permissions now come from an ACT (those denials are now green ☑). Select **SAS Administrators** and notice the green grants of the same permissions. These ACT settings override and hide the underlying indirect settings.

e Click **OK** to close the Properties dialog box for the **test2** folder.

Note: If you are restricted, an error message indicates that you can't save the settings. Click **OK** to dismiss the message. On the **Authorization** tab, select yourself and add explicit ☑ grants of ReadMetadata and WriteMetadata permissions. Click **OK**. △

5 To clean up, right-click the **test2** folder and select **Delete**.

Several predefined ACTs are provided on the **Plug-ins** tab under **Authorization Manager ▸ Access Control Templates**. You can create additional ACTs in this location.

Inherited Settings

Instead of setting permissions on every individual item, use inherited settings. This approach reduces the number of access controls that you have to manage. For example, rather than adding explicit settings or ACTs to every report, you can set permissions on a folder that contains reports for which those permissions are appropriate.

To learn more, complete this exercise in SAS Management Console:

1 Log on as someone who has a well-formed user definition.

2 On the **Folders** tab, right–click your **My Folder**  and select **New Folder**. Create a new folder named **parent**.

3 Right-click the **parent** folder and create another folder named **child**.

4 Right-click the **child** folder and select **Properties**. On the **Authorization** tab, select **SASUSERS**. Notice that this group has an indirect ☑ denial of the Read permission. Click **Cancel**.

5 Right-click the **parent** folder and select **Properties**. On the **Authorization** tab, select **SASUSERS**, add an explicit ☑ grant of Read permission, and click **OK**.

6 Right-click the **child** folder and select **Properties**. On the **Authorization** tab, select **SASUSERS**. Notice that this group now has an inherited ☑ grant of Read permission.

7 On the **child** folder's **Authorization** tab, add an explicit ☑ grant of Read permission on top of the inherited ☑ grant of Read permission, and click **OK**. This ensures that read access for SASUSERS is preserved even if the setting on the **parent** folder changes.

8 To verify that the explicit setting on the **child** folder is preserved, change the **parent** folder setting for SASUSERS to an explicit ☑ denial of Read permission, and then check the **child** folder settings again. For SASUSERS, the explicit ☑ grant of Read permission is still there. The denial on the **parent** folder is not relevant for the **child** folder because there is an explicit setting on the **child** folder.

9 To clean up, right-click the **parent** folder and select **Delete**.

Using WriteMetadata and WriteMemberMetadata Permissions

The following permissions affect the ability to create, update, and delete metadata.

WriteMetadata (WM)
: Edit, delete, change permissions for, or rename an item. For example, to edit a report, you need WM for the report. To delete a report, you need WM for the report (and WMM for the report's parent folder). For containers other than folders (such as repositories, libraries, and schemas), WM also affects adding and deleting child items. For example, to add an item anywhere in a repository, you need WM at the repository level. For folders, adding and deleting child items is controlled by WMM, not WM.

WriteMemberMetadata (WMM)
: Add an item to a folder or delete an item from a folder. For example, to save a report to a folder, you need WMM for the folder. To remove a report from a folder, you need WMM for the folder (and WM for the report). To enable someone to interact with a folder's contents but with not the folder itself, grant WMM and deny WM.

Note: We recommend that anyone who has a grant of WM is not denied WMM. △

To experiment with WM and WMM, complete this exercise in SAS Management Console:

1 Log on as someone who has a well-formed user definition.

Note: Step 5a assumes that you are restricted and aren't in the SAS Administrators group. To create a temporary restricted user for this exercise, complete steps 2-3 in "Add Administrators" on page 9 (for example, use the name **temp** and log on as temp@saspw). △

2 On the **Folders** tab, right-click your **My Folder** and select **New Folder**. Create a new folder named **learn**.

3 To see how WM influences WMM:

   a Right-click the **learn** folder, select **Properties**, and select the **Authorization** tab.
   b Notice that WMM is in the permissions list. This permission is meaningful only for folders.
   c In the **Users and Groups** list box, select **PUBLIC**. Notice that this group has indirect ☑ denials for both WM and WMM. Add an explicit ☑ grant of WM. Notice that this causes the WMM setting to change to a grant.
   d Select the grant WM check box again. This clears the check box and removes the explicit grant. Notice that the WMM setting also reverts to a denial.
   e Add an explicit ☑ grant of WMM. Notice that this has no effect on the WM setting. The mirroring is one-way. WM influences WMM, but WMM doesn't influence WM. Remove the grant of WMM to revert to the initial settings (indirect ☑ denials of both WM and WMM). Click **OK**.

4 To see how WMM on a folder is conveyed to the items inside the folder:

   a Right-click the **learn** folder and select **New Folder**. Create a new folder named **child**.

   b On the **learn** folder's **Authorization** tab, click **Add**. In the Add Users and Groups dialog box, clear the **Show Groups** check box. Move one restricted user (such as the **SAS Demo User**) to the **Selected Identities** list box and click **OK**.
   c In the permissions list, give the user who you just added an explicit denial of WM and an explicit grant of WMM. Click **OK**.

      Note: If the permissions list is disabled, the selected user is unrestricted (for example, the original SAS Administrator is unrestricted). Add a restricted user to the **Authorization** tab. △

   d On the **child** folder's **Authorization** tab, select the user who you added in step 4b. Notice that the denial of WM on the **learn** folder isn't conveyed to the **child** folder. Instead, the grant of WMM on the **learn** folder is conveyed to the **child** folder as an indirect grant of WM. On the **child** folder, the WMM setting mirrors the WM setting as usual.

5 To see which actions each permission controls:

   a Right-click your **My Folder** . Notice that actions such as **New Folder** and **New Stored Process** are available (because you have WMM) but, if you are a regular user, **Rename** and **Delete** are not (because you don't have WM).

      Note: This is an example of a folder that is under administrative control. Certain users (or groups) can contribute items to the folder, but the folder itself is protected. △

   b Right-click the **learn** folder and examine its pop-up menu. Notice that the **New Folder**, **New Stored Process**, **Delete**, and **Rename** actions are all available (because you have both WM and WMM).

6 To clean up, right-click the **learn** folder and select **Delete**. If you created a temporary user for this exercise, log on with your administrative account, delete

the temporary user (on the **Plug-ins** tab under **User Manager**) and that user's associated folder (at **SAS Folders ▶ Users ▶ <the temporary user>**).

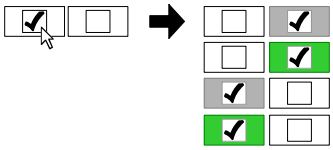
Key Points About Working With Permissions

Here are some key points about using metadata layer permissions:

- These permissions supplement protections from the host environment and other systems. Across authorization layers, protections are cumulative. In order to perform a task, a user must have sufficient access in all applicable layers.
- Setting permissions is an item-centric activity. To define permissions for someone, do not begin by finding that person's user definition. Instead, begin by navigating to an item that you want to protect or make available.
- Only the ReadMetadata and WriteMetadata permissions are universally relevant and enforced. See "Use and Enforcement of Each Permission" on page 53.
- After you add a broad denial, review the impact that this has on everyone else. For example, if the only setting on an item is an explicit PUBLIC denial, that denial blocks access for everyone (other than unrestricted users). To offset the denial, add one or more selective explicit (or ACT) grants.
- Before you deny the ReadMetadata permission on a folder, consider the navigational consequences. Without ReadMetadata permission on a folder, you can't navigate to items beneath that folder. Users need a clear path of grants of ReadMetadata permission in order to navigate to the content that they use.

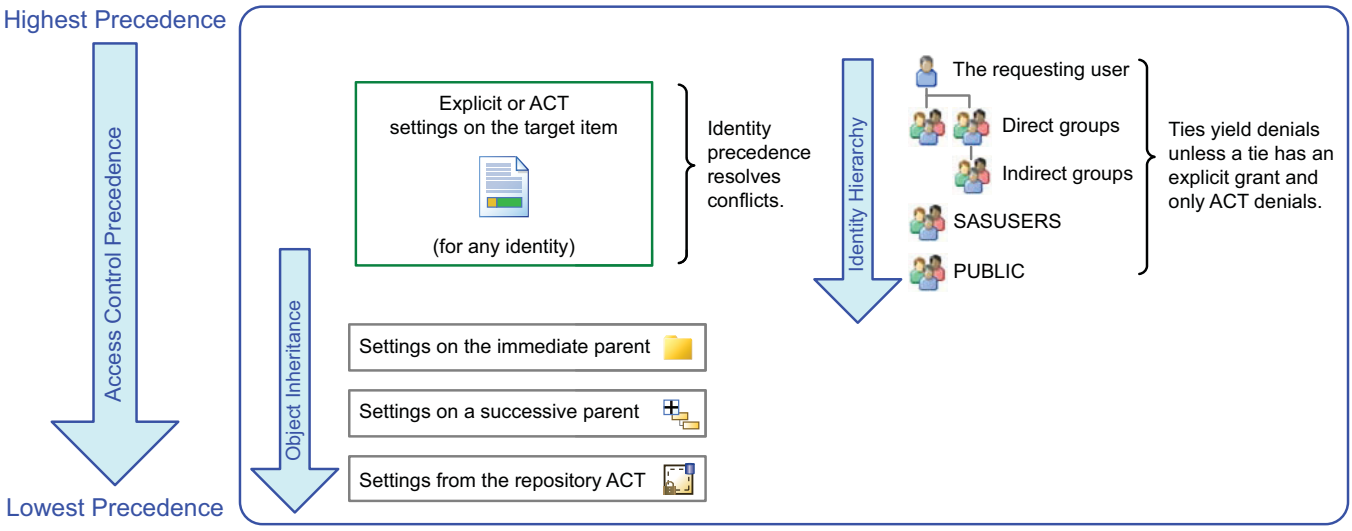
The following table summarizes what happens when you select a check box on the **Authorization** tab. In each row, the pointer indicates an action (a mouse click) that occurs between the before and the after.

Table 4.2 Mechanics of the Effective Permissions List

Before and After	Explanation
	A new explicit setting overrides and hides the opposing indirect (gray) setting.
	A new explicit setting overrides and hides the opposing ACT (green) setting.
	A new explicit setting is added on top of the matching indirect (gray) setting.
	A new explicit setting is added on top of the matching ACT (green) setting.
	A new explicit setting replaces the opposing explicit setting.
	The explicit setting is removed and an underlying indirect (gray) or ACT (green) setting is revealed.

The following figure summarizes the relative priority of access controls based on where they are set and who they are assigned to.

Figure 4.1 Summary: Relative Precedence of Access Controls



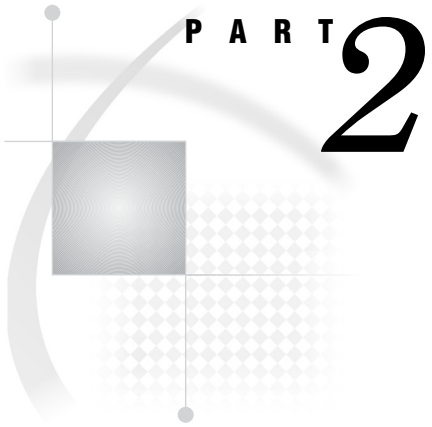
In the preceding figure, notice that explicit and ACT settings on an object (a report in this case) always have priority over settings on the object's parent (a folder in this case). For example, if a report has an explicit denial of ReadMetadata permission for PUBLIC and the report's folder has a grant of ReadMetadata permission for you, you can't see the report. For further discussion and examples, see "Authorization Decisions" on page 62.

See Also

"The Authorization Tab" on page 40

"Orientation to Working With Permissions" on page 39

SAS Management Console: Guide to Users and Permissions



PART 2

Authorization

CHAPTER

5

Authorization Model

Overview of the Metadata Authorization Model

About Metadata-Based Permissions

SAS provides a metadata-based authorization layer that supplements protections from the host environment and other systems. Across authorization layers, protections are cumulative. In order to perform a task, a user must have sufficient access in *all* applicable layers.

CAUTION:
Some clients (such as SAS Data Integration Studio and SAS Enterprise Guide) enable power users to create and run SAS programs that access data directly, bypassing metadata layer controls. It is important to manage physical layer access in addition to metadata layer controls. For example, use host operating system protections to limit access to any sensitive SAS data sets. See "Host Access to SAS Tables" on page 187. △

CAUTION:
Not all permissions are enforced for all items. Enforcement of permissions other than ReadMetadata and WriteMetadata varies by item type and (for data) by the method with which a library is assigned. △

Granularity and Mechanics

You can set permissions at these levels of granularity:

- *Repository-level controls* function as a gateway and as a parent-of-last-resort. Repository-level controls are managed from the **`Permission Pattern`** tab of the repository ACT. All registered users should have ReadMetadata and WriteMetadata permissions for the foundation repository.
- *Resource-level controls* manage access to a specific item such as a report, an information map, a stored process, a table, a column, a cube, or a folder. You can define resource-level controls individually (as explicit settings) or in patterns (by using access control templates).
- *Fine-grained controls* affect access to subsets of data within a resource. To establish fine-grained controls, you add constraints called permission conditions to explicit grants of the Read permission.

Inheritance Paths and Identity Precedence

Permission settings are conveyed across two distinct relationship networks:

- In the resource relationships network, permissions that you set on one item can affect many other items. For example, a report inherits permissions from the folder in which the report is located. This network is a simple folder tree, with these exceptions:
  - The root folder isn't the ultimate parent. This folder inherits from the repository (through the permission pattern of the repository ACT).
  - The root folder isn't a universal parent. Some system resources (such as application servers, identities, and ACTs) are not in the folder tree. For these items, the repository ACT is the immediate and only parent.
  - Inheritance within a table or cube follows the data structure. For example, table columns and cube hierarchies don't have a folder as an immediate parent. Instead, a column inherits from its parent table and a dimension inherits from its parent cube.
- In the identity relationships network, permissions that you assign to one identity can affect many other identities. For example, if you grant a group access to a report, that grant applies to everyone who is a member of the group. This relationship network is governed by a precedence order that starts with a primary (usually individual) identity, can incorporate multiple levels of nested group memberships, and ends with implicit memberships in SASUSERS and then PUBLIC.

Use and Enforcement of Each Permission

Table 5.1 Use and Enforcement of Each Permission

Permission (Abbreviation)	Actions Affected and Limitations on Enforcement
ReadMetadata (RM)	View an item or navigate past a folder. For example, to see an information map you need RM for that information map. To see or traverse a folder you need RM for that folder.
WriteMetadata (WM)	Edit, delete, change permissions for, or rename an item. For example, to edit a report you need WM for the report. To delete a report you need WM for the report (and WMM for the report's parent folder). For containers other than folders (such as repositories, libraries, and schemas), WM also affects adding and deleting child items. For example, to add an item anywhere in a repository you need WM at the repository level. For folders, adding and deleting child items is controlled by WMM, not WM.
WriteMemberMetadata (WMM)	Add an item to a folder or delete an item from a folder. For example, to save a report to a folder you need WMM for the folder. To remove a report from a folder, you need WMM for the folder (and WM for the report). To enable someone to interact with a folder's contents but with not the folder itself, grant WMM and deny WM.[1]
CheckInMetadata (CM)	Check in and check out items in a change-managed area. Applicable only in an optional configuration for SAS Data Integration Studio.[2]
Administer (A)	Monitor an OLAP server. Stop, pause, resume, refresh, or quiesce a server or spawner. For the metadata server, the ability to perform these tasks is managed by the **`Metadata Server: Operation`** role, not by the A permission.
Read (R)	Read data. For example, while you need RM for a cube in order to see that cube, you need R for the cube in order to run a query against it. Enforced for OLAP data, information maps, data that is accessed through the metadata LIBNAME engine, and dashboard objects.
Create (C)	Add data. For example, on a table, C controls adding rows to the table. Enforced for data that is accessed through the metadata LIBNAME engine.
Write (W)	Update data. For example, on a table, W controls updating the rows in the table. Enforced for data that is accessed through the metadata LIBNAME engine, for publishing channels, and for dashboard objects.
Delete (D)	Delete data. For example, D on a library controls deletion of tables from the library. Enforced for data that is accessed through the metadata LIBNAME engine and for dashboard objects.

1 A folder's WMM settings mirror its WM settings unless the folder has explicit ☑ or ACT ☑ (green) settings of WMM. A grant (or deny) of WMM on a folder becomes an inherited grant (or deny) of WM on the items and subfolders in that folder. WMM is not inherited from one folder to another.

2 In any change-managed areas of a foundation repository, change-managed users should have CM (instead of WM and WMM). Change management is a SAS Data Integration Studio feature.

Inheritance Paths

In the metadata layer, parent items convey their effective permissions to child items. Children inherit the net effect of their parents' access controls, not the access controls

themselves. The following figures depict inheritance paths in a foundation repository. The arrows in the figure flow from parent to child. For example, a folder conveys its effective permissions to the items that it contains. The arrows in the second figure flow from child to parent, showing the same relationships from a different perspective.

Figure 5.1 Inheritance Paths (Integrated View)

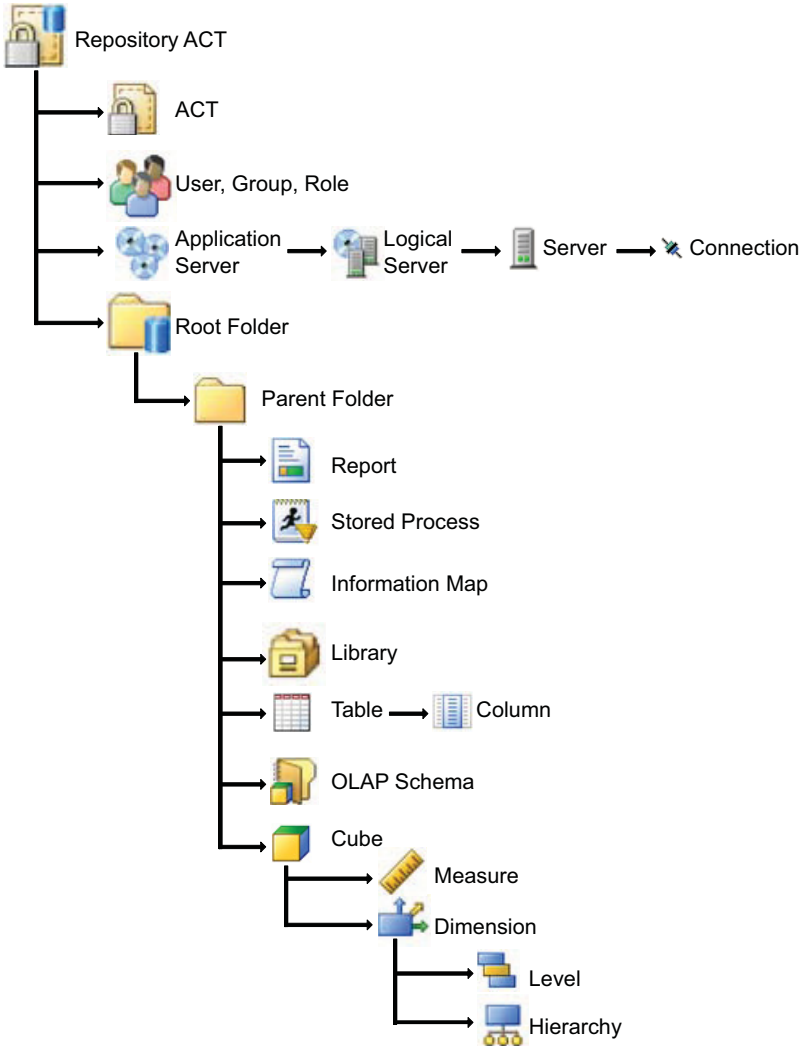
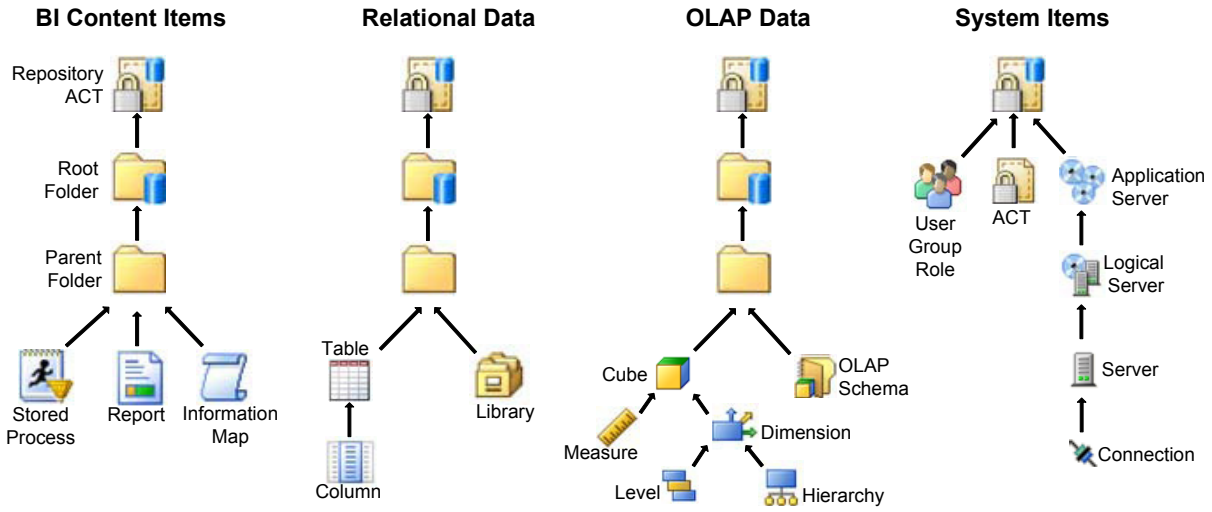


Figure 5.2 Inheritance Paths (Separated View)



Here are some details about the preceding figures:

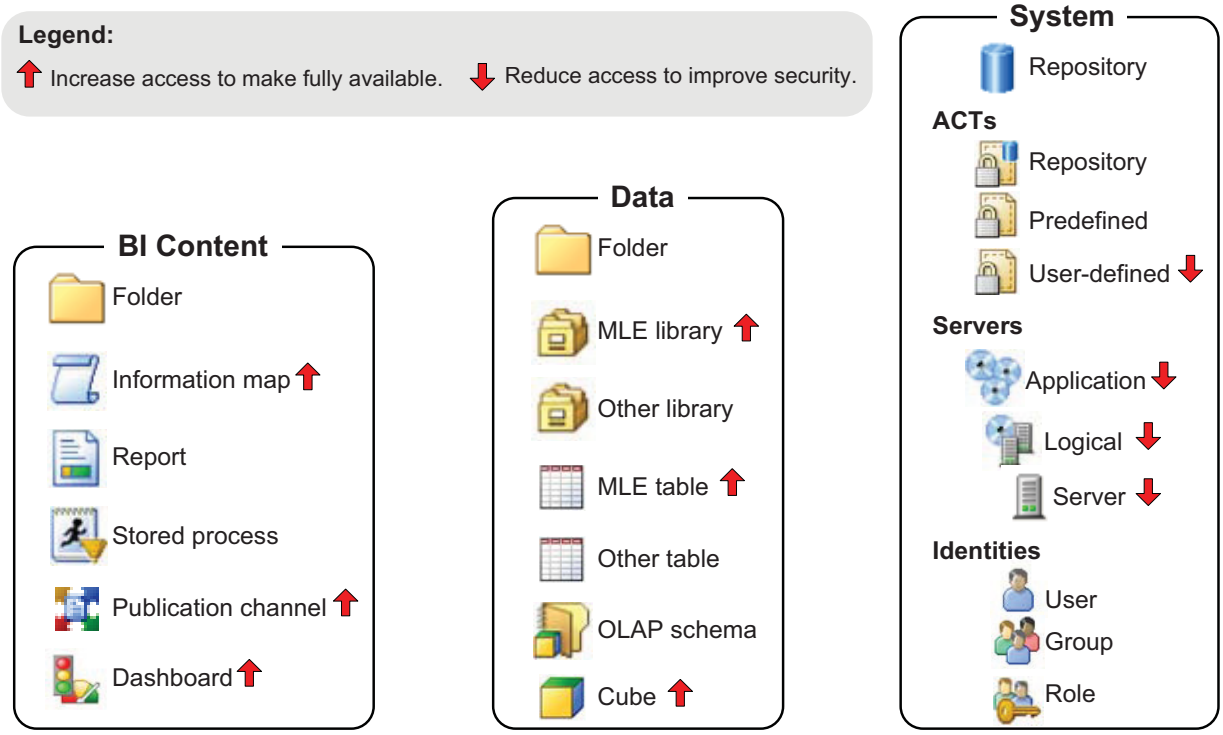
- The depicted folder structure is arbitrary and intended only to show the security relationships between different types of items.
- Not all item types are depicted. To trace inheritance for a particular item, click **`Advanced`** on that item's **`Authorization`** tab. This feature is available to only unrestricted users.
- The root folder represents the top of the folder tree for the foundation repository. It corresponds to the **`SAS Folders`** node on the **`Folders`** tab in SAS Management Console.
- The root folder inherits settings from the **`Permission Pattern`** tab of the repository ACT.
- Any custom repositories are represented as folders (immediate children of the foundation root folder). Although these folders inherit permissions from both the foundation root folder and the repository ACT of the custom repository, access to items within the custom repository branch should be managed from the folder side (except for items that aren't in the folder tree).
- On the **`Folders`** tab in SAS Management Console, your **`My Folder`** is displayed directly below the root folder. This is just a shortcut for accessing your personal content area; this folder is not an immediate child of the root folder.
- The figures show users, groups, and roles inheriting repository-level permissions. The **`Authorization`** tab in a user, group, or role displays default settings that reflect special rules that prevent regular users from modifying or deleting identities. An identity's **`Authorization`** tab can affect access to that identity's definition. An identity's **`Authorization`** tab has no effect on what that identity can do.

Permissions by Item

The following figure depicts common items and uses arrows to indicate items whose permission settings are most likely to need an adjustment. Here are some details about the figure:

- Most of the up arrows indicate a need to extend the ability to read data. In the interest of a more secure initial deployment, read access to data is not granted by default.
- Each down arrow indicates a need to limit the ability to modify or delete a system item. Other system items are protected by default.
- MLE refers to the metadata LIBNAME engine. The MLE items are listed to highlight the elevated permission requirements that apply when data is accessed through this engine.
- In any change-managed areas in a foundation repository, change-managed users should have CheckInMetadata permission (instead of WriteMetadata or WriteMemberMetadata). Change management is a SAS Data Integration Studio feature.
- The intent of the figure is to highlight the adjustments that are most commonly needed. In general, it is preferable to set permissions on a parent (such as a folder) rather than on a specific item (such as a report).

Figure 5.3 Common Access Adjustments By Item



The following tables provide item-specific instructions and tips. The purpose of the tables is to highlight special requirements for common items.

Table 5.2 Permissions Tips: Selected Items in the Folder Tree

 Root folder	The root folder is the first folder on the **Folders** tab. This folder's **Authorization** tab is an important point of control for narrowing WM. Unlike other folders, the root folder does not have the WriteMemberMetadata permission in its permission list. This is because the root folder is a software component object, not a true folder. The root folder can contain only other folders.
 Folder	For folders, follow these guidelines: □ Provide users with a clear path of grants of ReadMetadata permission to all of the content that they access. This is a navigational requirement; to navigate past a folder, you need ReadMetadata permission for that folder. □ To enable users to contribute items to a folder, grant them WriteMemberMetadata permission on that folder. □ On a folder, grant WriteMetadata permission only to users who should be able to delete, move, or rename that folder. □ Don't assume that every permission that is listed on a folder's **Authorization** tab is relevant for every item in that folder. □ If you deny ReadMetadata permission on a folder, make sure that you don't prevent the SAS Trusted User from having that permission on all cubes and schemas within that folder. A reasonable approach is to give the SAS System Services group a grant of ReadMetadata permission on the folder. This preserves necessary access for this privileged service identity. See Chapter 6, "Permissions on Folders," on page 71 and "Using WriteMetadata and WriteMemberMetadata Permissions" on page 45.
 Information map	To access any data through an information map, you need Read permission on that information map. You can manage Read permission globally or selectively. For example: □ A broad approach is to grant Read permission to SASUSERS on the repository ACT's **Permission Pattern** tab or on the **Authorization** tab of the top folder. □ A narrow approach is to grant Read permission to smaller groups on specific subfolders or even specific information maps.
 Report	If a user can't run a report, check these things: □ Does the user have Read permission for the underlying information map? □ Does the user have Read permission for any underlying OLAP cube? □ Does the user have Read permission for any underlying relational data that is accessed via the MLE? □ Does the SAS Trusted User have ReadMetadata permission for any underlying cube? □ Does the account that retrieves any underlying SAS data have physical access to that data? □ Is the underlying information map in a legitimate folder? See the wrs.map.accessibility.check property in the *SAS Intelligence Platform: Web Application Administration Guide*. It is especially important to manage ReadMetadata permission to pregenerated reports, because those reports contain data. Any user who views a pregenerated report sees the same data, regardless of his or her permissions to the underlying tables or cubes.

 Stored process	To run a stored process, you need ReadMetadata permission on the stored process. If you can see a stored process but can't run it, you might not have the necessary grant of Read permission for the underlying data. To register a stored process, you need WriteMetadata permission for the target application server. The method that you use to make a stored process available can affect data retrieval and security. For example, in the standard configuration, a stored process that is assigned to a workspace server and embedded in an information map retrieves SAS data under the pooled workspace server's host identity. However, if a user opens that same stored process directly (for example, as a report in SAS Web Report Studio), the host identity of the requesting user (or group) retrieves the data.
 Channel	In a new deployment, only the SAS Administrators group can add channels, subscribers, and content. To enable all registered users to publish content to a particular channel, navigate on the **Folders** tab to System ▶ Publishing ▶ Channels and grant Write and WriteMetadata permissions to SASUSERS on that channel's **Authorization** tab (WM is required only if a channel has an archive persistent store). To enable all registered users to add channels or subscribers, grant WriteMemberMetadata permission on the relevant parent folder (for example, on the System ▶ Publishing ▶ Subscribers ▶ Content Subscribers folder). See "Using WriteMetadata and WriteMemberMetadata Permissions" on page 45.
 Dashboard	The permissions model for dashboards and their related items (indicators, data models, and ranges) has several unique aspects. See the *SAS Intelligence Platform: Web Application Administration Guide*.
 Schema	To associate an OLAP schema with an application server, you need WriteMetadata permission for the schema and the server. To add cubes to a schema, you need WriteMetadata permission for the schema and WriteMemberMetadata permission for the target folder. The SAS Trusted User must have ReadMetadata permission for all OLAP schemas and cubes. This access is usually granted through the SAS System Services group.
 Cube	To register cubes, you need WriteMetadata permission for the OLAP schema and WriteMemberMetadata permission for the target folder. To associate a cube with a schema, you need WriteMetadata permission for the cube and the schema. Read permission is enforced for cubes. Grant Read permission broadly (as described in the information map row), or add narrower grants of Read permission on folders or individual cubes. The SAS Trusted User must have ReadMetadata permission for all OLAP schemas and cubes. This access is granted through the SAS System Services group.
 Cube component	To access the **Authorization** tab for an OLAP dimension, measure, level, or hierarchy, navigate on the **Plug-ins** tab under Authorization Manager ▶ Resource Management ▶ By Location ▶ <Application Server>. For cube components, Read permission is enforced. If you lack access to a measure that participates in a calculated measure, you can get unintended results. There are navigational requirements for Read permission on cube components. If a user does not have Read permission to a hierarchy, the user can't navigate to the top levels within the hierarchy. If a user does not have Read permission to a particular level in a hierarchy, the user can't navigate to the next level.
 Library	To associate a library with an application server, you need WriteMetadata permission for the server (but not for the library). For a library that is accessed via the metadata LIBNAME engine (MLE), you need the Create permission in order to add tables and the Delete permission in order to delete tables.

 Table	To associate a table with a library, you need WriteMetadata permission for the table and the library. For a table that is accessed via the MLE, you need Read permission in order to access data. The Create, Delete, and Write permissions affect your ability to add, update, or delete data. You can grant these permissions broadly (as described in the information map row) or narrowly (for example, to a small group of users on a particular table's **Authorization** tab).
 Column	To access a column's **Authorization** tab, navigate on the **Plug-ins** tab under Authorization Manager ▶ Resource Management ▶ By Location ▶ <Application Server>. The MLE doesn't support column-level access distinctions for the Read permission. Column-level access distinctions for the ReadMetadata permission are always supported.

Table 5.3 Permissions Tips: Selected Items Outside the Folder Tree

 Repository	On a foundation repository, all participating users should have repository-level ReadMetadata and WriteMetadata permissions. To verify access: **1** On the **Plug-ins** tab, select **Authorization Manager ▶ Access Control Templates** and select the repository ACT. This ACT is usually named **Default ACT** and always has a blue cylinder as part of its icon  . **2** Right-click, select **Properties**, and select the **Permission Pattern** tab. **3** Select the SASUSERS group. On the repository ACT for a foundation repository, SASUSERS is usually granted ReadMetadata and WriteMetadata. Other predefined entries in this pattern provide necessary administrative and service access.
 Application server	To monitor or operate servers other than the metadata server, you need the Administer permission on the server. To verify access: **1** On the **Plug-ins** tab, navigate to **Server Manager** and select a server. **2** Right-click, select **Properties**, and select the **Authorization** tab. **3** Select the SAS Administrators group. Verify that this group is granted ReadMetadata, WriteMetadata, and Administer permissions. To associate a stored process, OLAP schema, or library with an application server, you need WriteMetadata permission for that application server. Certain service identities need ReadMetadata permission to all server definitions. See Chapter 7, "Permissions on Servers," on page 85.
 Logical server	To use a logical server, you need ReadMetadata permission for at least one of that server's connections. This is called server access security. Certain service identities need ReadMetadata permission to logical server definitions. See "Hiding Server Definitions" on page 88.

Identity

User administration capabilities enable you to create, update, and delete users, groups, and roles. You can delegate management of an identity to someone who doesn't have user administration capabilities by adding explicit ☑ or ACT ☑ (green) grants of WriteMetadata permission on the identity's **`Authorization`** tab. An identity's **`Authorization`** tab has no effect on what that identity can do.

ACT

To create an ACT, you need repository-level WriteMetadata permission. Each predefined ACT is protected by settings on its **`Authorization`** tab. ACTs that you create aren't automatically protected. It is important to add protections to any ACTs that you create. To protect an ACT:

1. On the **`Plug-ins`** tab, select the ACT under **Authorization Manager ▸ Access Control Templates** .
2. On the ACT's **`Authorization`** tab, narrow access. For example, if you explicitly ☑ deny WriteMetadata permission to PUBLIC, only unrestricted users can modify or delete the ACT. If you also add an explicit grant for the SAS Administrators group, members of that group can also modify or delete the ACT.

Permissions by Task

The following tables show required metadata layer permissions for selected tasks. This list explains symbols in the tables:

- An asterisk (*) in a column heading indicates that an item is optional. For example, not all reports include a stored process.
- R^{MLE} indicates that the Read permission is required only if data is accessed through the metadata LIBNAME engine.
- $R^{MLE, OLAP}$ indicates that the OLAP data can be also involved (the Read permission is always required for OLAP data).

Table 5.4 Working with Folders: Permissions for Selected Tasks

Task	Repository	Parent Folder	Folder	Item
Add a folder	RM, WM	RM, WMM[1]	-	-
Delete a folder	RM	RM, WMM[1]	RM, WM	-
Rename a folder	RM	RM	RM, WM	-
Set folder permissions	RM	RM	RM, WM	-
Add an item to a folder	RM, WM	RM	RM, WMM	-
Delete an item from a folder	RM	RM	RM, WMM	RM, WM
Copy/export items	RM	RM	RM	RM
Paste/import items	RM, WM	RM	RM, WMM	-

1 If the parent folder is the root folder , you need RM, WM on the root folder.

Table 5.5 Working with Reports: Permissions for Selected Tasks

Task	Repository	Parent Folder	Report	Stored Process*	Information Map*	Data
Create and save a new report	RM, WM	RM, WMM	-	RM	RM, R	RM, $R^{MLE, OLAP}$
Delete a report	RM	RM, WMM	RM, WM	-	-	-
View or refresh a report	RM	RM	RM	RM	RM, R	RM, $R^{MLE, OLAP}$
View a batch report	RM	RM	RM	-	-	-
Edit or rename a report	RM	RM	RM, WM	-	-	-
Set report permissions	RM	RM	RM, WM	-	-	-

Table 5.6 Working with Information Maps: Permissions for Selected Tasks

Task	Repository	Parent Folder	Information Map	Stored Process*	Data
Create and save a new information map	RM, WM	RM, WMM	-	RM	RM, $R^{MLE, OLAP}$
Delete an information map	RM	RM, WMM	RM, WM	-	-
Set information map permissions	RM	RM	RM, WM	-	-
Edit or rename an information map	RM	RM	RM, WM	-	-
Run queries in an information map	RM	RM	RM, R	RM	RM, $R^{MLE, OLAP}$

Table 5.7 Working with Stored Processes: Permissions for Selected Tasks

Task	Repository	Parent Folder	Application Server	Stored Process	Data
Register a stored process	RM, WM	RM, WMM	RM, WM	-	-
Delete a stored process	RM	RM, WMM	RM, WM	RM, WM	-
Set stored process permissions	RM	RM	RM	RM, WM	-
Run a stored process	RM	RM	RM	RM	RM, $R^{MLE, OLAP}$

Table 5.8 Working with Publishing Channels: Permissions for Selected Tasks

Task	Repository	Parent Folder	Channel	Subscriber
Add a channel or subscriber	RM, WM	RM, WMM	-	-
Delete a channel or subscriber	RM	RM, WMM	RM, WM	RM, WM
Edit a channel or subscriber	RM	RM	RM, WM	RM, WM
Publish content to a channel	RM, WM[1]	RM	RM, W, WM[1]	RM[2]

1 WM is required if the channel has an archive persistent store.
2 Content is published to only those subscribers for whom you have RM.

Table 5.9 Working with Tables: Permissions for Selected Tasks

Task	Repository	Application Server	Library	Parent Folder	Table	Column
Register a table	RM, WM	RM	RM, WM	RM, WMM	-	-
Delete a table	RM	RM	RM, WM	RM, WMM	RM, WM	-
Set table permissions	RM	-	RM	RM	RM, WM	-
Access table data	RM	RM	RM	RM	RM, R^{MLE}	RM
Register a library	RM, WM	RM, WM	-	RM, WMM	-	-

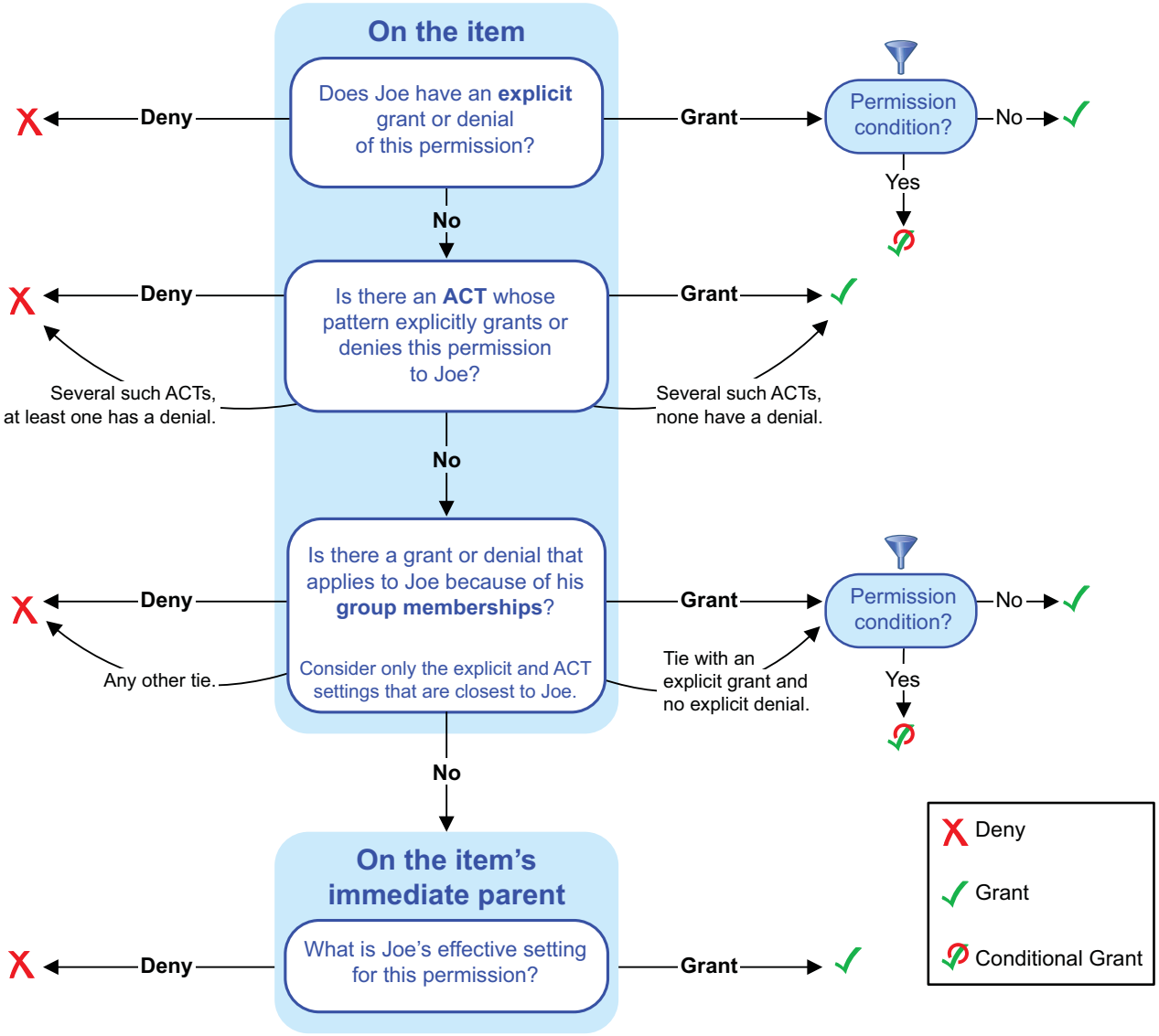
Table 5.10 Working with Cubes: Permissions for Selected Tasks

Task	Repository	Application Server	OLAP Schema	Parent Folder	Cube	Source Data Sets
Register a cube	RM, WM	RM	RM, WM	RM, WMM	-	RM, R^{MLE}
Delete a cube	RM	RM	RM, WM	RM, WMM	RM, WM	-
Rebuild a cube	RM	RM	RM	RM	RM, WM	RM, R^{MLE}
Refresh a cube	RM	RM	RM	RM	RM, R	RM, R^{MLE}
Set cube permissions	RM	-	RM	RM	RM, WM	-
Access cube data	RM	RM	RM	RM	RM, R	-
Register a schema	RM, WM	RM, WM	-	RM, WMM	-	-
Use the OLAP Server Monitor	RM	A	-	-	-	-

Authorization Decisions

The following figure depicts the evaluation process that determines whether a restricted user (Joe) has a particular permission for a particular item.

Figure 5.4 Authorization Decision Flowchart



Here are some details that help you interpret the preceding figure:

explicit settings
: A user has an explicit ☑ setting on an item if the user is individually granted or denied a permission on the item's **`Authorization`** tab. Explicit settings have the highest precedence. However, managing a large number of explicit settings for individual users can be cumbersome. For greater efficiency, we recommend that you set explicit permissions for groups, use ACTs, and rely on inheritance.

ACT settings
: A user has an ACT setting ☑ (green) on an item if an ACT that is applied to the item has a permission pattern that explicitly grants or denies the relevant permission to the user. Each ACT adds its pattern of grants and denials to the settings for each item to which the ACT is applied. A green check box on an item's **`Authorization`** tab indicates that at least one ACT is applied to that item. To determine which ACTs are applied to an item, click the **`Access Control Templates`** button on the item's **`Authorization`** tab.

indirect settings (derived from group memberships)

A user has a group setting ☑ on an item if a group to which the user belongs has an explicit or ACT setting on the item. Group settings matter only if the requesting user has no relevant explicit or ACT settings on the target item. Identity precedence determines which group settings are closest to each user. See "Identity Precedence" on page 34.

If you directly assign a user to multiple groups (or directly assign a group to multiple groups), you can get identity precedence ties. For example, if Joe is a direct member of GroupA and GroupB, and the only settings on an item are conflicting explicit settings for those groups, both of those settings are closest to Joe. These conflicts are resolved as follows:

- Explicit settings have precedence over ACT settings.
- Conflicting explicit settings result in a denial.
- Conflicting ACT settings result in a denial.

indirect settings (inherited from a parent item)

Inherited settings ☑ come from a parent item (such as a folder). Inherited settings matter only if there are no relevant explicit, ACT, or group settings on the target item. When there are no relevant settings on an item, access to the item is the same as access to the item's immediate parent. The controlling setting is usually an explicit or ACT setting that is defined somewhere in the item's inheritance path.

If there are no explicit or ACT settings on any parent (or if the item's only parent is the repository), the repository ACT determines the outcome. If that ACT's pattern neither grants nor denies the permission, then the permission is denied.

Note: If there is no repository ACT, all permissions are granted (you should always have a designated repository ACT). △

Note: If an item has more than one immediate parent, a grant from any inheritance path is sufficient to provide access. Having multiple immediate parents is not a common situation. △

permission conditions

Permission conditions constrain explicit grants of the Read permission on OLAP dimensions (limiting access to members) or information maps (limiting access to rows). On the **`Authorization`** tab, the presence of an **`Edit Condition`** or **`Edit Authorization`** button indicates that a permission condition is assigned to the currently selected user or group. See Table 5.14 on page 69.

The following table summarizes the principles of the authorization decision process:

Table 5.11 Precedence Principles for Authorization

Principle	Example[1]	Outcome
Settings on an item have priority over settings on the item's parents.	LibraryA has an explicit denial for PUBLIC. LibraryA's parent folder has an explicit grant for Joe.	Joe can't see LibraryA.
Conflicting settings on an item are resolved by identity precedence.	LibraryA has an explicit denial for GroupA and an explicit grant for GroupAA. Joe is a direct member of GroupA. GroupA is a direct member of GroupAA.	Joe can't see LibraryA.
In an identity precedence tie, explicit settings have priority over ACT settings.	LibraryA has an ACT denial for GroupA and an explicit grant for GroupB. Joe is a direct member of both GroupA and GroupB.	Joe can see LibraryA.

Principle	Example[1]	Outcome
In an identity precedence tie that isn't resolved by the preceding row, the outcome is a denial.	LibraryA has an explicit denial for GroupA and an explicit grant for GroupB. Joe is a direct member of both GroupA and GroupB.	Joe can't see LibraryA.
A grant from any inheritance path can provide access.	ObjectA has no explicit or ACT settings, one immediate parent that conveys a grant, and another immediate parent that conveys a denial.[2]	Joe can see ObjectA.

1 The settings described in this column are the only explicit or ACT settings for ReadMetadata permission on LibraryA (or ObjectA).

2 Having more than one immediate parent is not a common circumstance.

Fine-Grained Controls for Data

What Are Fine-Grained Controls?

Business requirements often specify that different users should see different portions, or slices, of data. In some cases, the requirement is driven by the sensitive nature of data. For example, company policy might state that each sales person should be able to access only his or her own salary information. In other cases, the requirement is intended to prevent information overload. For example, each regional sales team within a national organization might be interested in only the sales trend information for their region. Row-level access distinctions are frequently based on each user's place in an organizational structure such as a management hierarchy or a product matrix. The visibility of data can depend on a simple, site-specific condition such as a user's security clearance level, or on a more complex condition that consists of multiple filters.

You use fine-grained controls to specify who can access particular rows within a table or members within a cube dimension. These controls often subset data by a user characteristic such as employee ID or organizational unit. For example, a table that contains patient medical information might be protected by row-level permissions that enable each doctor to see only those rows that contain data about that doctor's patients.

Unlike other access controls, fine-grained controls are based on filters and rely on target data that is modeled to work with those filters. When fine–grained controls are used, there are three possible authorization decision outcomes for a request to view data:

Grant	The requesting user can access all data.
Deny	The requesting user can't access any data.
Grant-with-conditions	The requesting user can access only the data that meets specified filtering conditions.

Considerations for Batch Reporting

When you use fine-grained controls, it is essential to understand that only dynamically generated reports display data based on the access that is defined for the requesting user. Static reports display data based on the access that is defined for the user ID that was used to generate the report. For example:

- Manually refreshed reports contain cached data (which can be updated by a user action in the report viewer).

- Pre-generated reports reflect the access of the user ID that was used to generate the report. Identity-specific access distinctions are preserved for pre-generated reports only if you define a separate report job for each user ID.

Identity–Driven Properties

Overview

It is often necessary to make per-person access distinctions for the rows in a table or the members in a dimension. You can make a separate filter for each user (such as **`where name="joe"`**). However, if you have more than a few users, this approach quickly becomes cumbersome. The more efficient alternative is to create a dynamic filter (such as **`where name="&name;"`**) that can discover and insert the correct, user-specific value into the WHERE expression each time access is requested.

To create a dynamic filter, use an identity–driven property as the value against which values in the target data are compared. This list explains how the substitution works:

Note: This discussion is not applicable to the SPD Server, which has its own implementation of identity-based filtering. △

1. Each identity-driven property corresponds to a characteristic (such as name, user ID, or external identity).
2. Each user's values for these characteristics (such as **`joe`**, **`WinXP\joe`**, or **`607189`**) are stored in the metadata.
3. The identity-driven property is aware of the user ID with which a client authenticated and can locate information that is stored in the metadata for that user ID.
4. Each time it receives a request, the identity-driven property substitutes a user-specific value into the filter expression.

These are the most useful identity-driven properties:

SAS.Userid
: returns an authenticated user ID, normalized to the uppercase format USERID or USERID@DOMAIN.

SAS.ExternalIdentity
: returns a site-specific value (for example, employee ID). This property is often useful because its values are likely to match user information in your data. An identity can have more than one external identity value; however, only the first value is returned. Unlike the values for other identity-driven properties, values for this property are not always populated in the metadata. See "External Identities" on page 226.

SAS.IdentityGroups
: returns a list of the groups and roles that this identity belongs to (directly, indirectly, or implicitly). The list contains the group and role names, as displayed in the **`Name`** field on the **`General`** tab for each group or role.

SAS.PersonName
: returns a user name, as displayed in the **`Name`** field on the user's **`General`** tab.

These identity-driven properties are also supported:

SAS.IdentityGroupName
: returns a group name, as displayed in the **`Name`** field on the group's **`General`** tab. If a user logs on with an ID that is stored in a login on a group definition, then the

name of the group that owns that login is returned. If a user logs on with a user ID that is not stored in the metadata, then the PUBLIC group is returned.

This property is useful only in the unusual circumstance where a user logs on with the user ID that is defined for a *group* login. In almost all cases, a user logs on with a user ID that is defined for an *individual* user definition. This property is not supported if client-side pooling is used.

SAS.IdentityName
returns a user name or group name, as displayed in the **`Name`** field on the **`General`** tab for the user or group. This property is a generalization of SAS.PersonName and SAS.IdentityGroupName.

Note: In certain circumstances, a connecting identity might not have a value for the identity-driven property that you are using. This can happen with the ExternalIdentity property (sometimes), the IdentityGroupName property (almost always), or the PersonName property (rarely). When a connecting user doesn't have a value for the property that a query uses, an empty string is returned or the query fails. △

Examples

For example, to enable each user to see only his or her own salary information, you could give the PUBLIC group a filter that is based on the SAS.PersonName property. At run time, the SAS.PersonName value that is associated with the connected user ID is substituted into the filter. In this way, the query is modified as appropriate for each requesting client.

This table contains examples of filters that are based on identity properties, showing representations of both the generic form and how each filter would be modified when executed by a user named Harry Highpoint. Harry is a member of the ETL and Executives groups. The example assumes that the customer has an employee information table named EmpInfo which includes Name, Category, WinID, Department, and EmpID columns.

Table 5.12 Examples of Filters That Use Identity-Driven Properties

As Defined (Generic Form)	As Executed (Resolved Form)
Where EmpInfo.WinID=&SAS.Userid;	Where EmpInfo.WinID="HIGH@WIN"
Where EmpInfo.EmpID=&SAS.ExternalIdentity;	Where EmpInfo.EmpID="123–456–789"
Where EmpInfo.Department IN &SAS.IdentityGroups;	Where EmpInfo.Department IN ('ETL','Executives','PUBLIC','SASUSERS')
Where EmpInfo.Name=&SAS.IdentityName;	Where EmpInfo.Name="Harry Highpoint"
Where EmpInfo.Name=&SAS.PersonName;	Where EmpInfo.Name="Harry Highpoint"
Where EmpInfo.Category=&SAS.IdentityGroupName;	Where EmpInfo.Category=' '[1]

1 Because the user does not log on with a user ID that is stored as part of a group definition, the user has no value for this property. This either returns an empty string (in BI row-level permissions) or causes the query to fail (in other implementations).

What Implementations are Available?

Each of these components offers an implementation of fine-grained controls:

Information Maps (BI Row-Level Permissions)

provides filtering for SAS data sets and third-party relational data accessed through an information map. You define and assign the filters in SAS Information Map Studio. See Chapter 8, "BI Row-Level Permissions," on page 93.

This feature is practical for use with large, dimensionally modeled data marts. BI row-level permissions do not require a specific data model. BI row-level permissions can limit access to data within fact tables without incurring the performance cost of directly filtering those tables. This is accomplished by ensuring that access to a fact table is always subject to an inner join with a filtered dimension (the filtering criteria is usually some type of identity information). This feature enables you to define granular access to third-party data without requiring you to maintain individual user accounts within those database systems.

CAUTION:
Not all SAS clients require that users go through information maps in order to access data. Comprehensive security that incorporates BI row-level permissions requires a specific, high-security configuration of SAS Web Report Studio. △

SAS OLAP Server
provides filtering for SAS OLAP data using MDX expressions. You define and assign the filters in SAS Management Console, SAS OLAP Cube Studio, or SAS Data Integration Studio. See Chapter 9, "OLAP Member-Level Permissions," on page 109.

SAS Scalable Performance Data Server
enables you to define database views that filter rows based on the user ID of the connecting client. This functionality is provided by the @SPDSUSR system variable. The metadata-based identity driven properties are not available in this implementation. See the *SAS Scalable Performance Data Server: Administrator's Guide*.

The following table compares the implementations:

Table 5.13 Comparison of Fine-Grained Control Implementations

Implementation	Secure	Graphical Filter Creation	Standard Authorization UI	Metadata-Aware	Identity-Driven
OLAP member-level	✓	✓	✓	✓	✓
SPD Server row-level	✓				✓
BI row-level	⚠	✓	✓	✓	✓

How are Fine-Grained Controls Assigned?

Note: This discussion is not applicable to the SPD Server. △

Fine-grained controls are assigned to users or groups on the **`Authorization`** tab of the target dimension or table. These filters are available only to constrain an explicit grant of the Read permission. These filters are incorporated into the access control evaluation process as permission conditions.

A permission condition is applied only if it is on the setting that is closest to the requesting user. Other conditions that are relevant because of further-removed group memberships don't provide additional, cumulative access. If there is an identity precedence tie between multiple groups at the highest level of identity precedence, those tied conditions are combined in a Boolean OR expression. If the identity

precedence tie includes an unconditional grant, access is not limited by any conditions. The following table provides examples:

Table 5.14 Precedence for Permission Conditions

Principle	Scenario	Outcome and Explanation
If there are multiple permission conditions that apply to a user because of the user's group memberships, then the highest precedence identity controls the outcome.	A filter on InformationMapA limits Read permission for GroupA. Another filter on InformationMapA limits Read permission for the SASUSERS group. The user is a member of both GroupA and SASUSERS.	The user can see only the rows that GroupA is permitted to see. GroupA has higher identity precedence than SASUSERS, so the filters that are assigned to GroupA define the user's access.
If there are multiple permission conditions at the highest level of identity precedence, then any data that is allowed by any of the tied conditions is returned.	A filter on DimensionA limits Read permission for GroupA. Another filter on DimensionA limits Read permission for GroupB. The user is a first level member of both GroupA and GroupB.	The user can see any member that is permitted for either GroupA or GroupB.

The following example describes the impact of identity precedence when a manager uses an information map that includes both of the following filters for a SALARY table:

- A permission condition that is assigned to the SASUSERS group gives each user access to his or her own salary information.
- A permission condition that is assigned to a Managers group enables each manager to see the salaries of the employees that he or she manages.

When the manager accesses the SALARY table, the filter that is assigned to the Managers group is applied and the filter that is assigned to SASUSERS is ignored. This is because the manager's direct membership in the Managers group has higher identity precedence than the manager's implicit membership in the SASUSERS group. To avoid a situation in which managers can see their employees' salaries but each manager can't see his or her own salary, you can use either of these approaches:

- Assign the filters to two groups that have the same identity precedence. For example, if you assign the first filter to a general purpose user-defined group (rather than to SASUSERS), and you make each manager a direct member of that group, then managers will have an identity precedence tie between that group and the Managers group. This situation causes the two filters to be combined for members of the Managers group, enabling those users to see any row that is permitted by either filter.
- Define the Managers filter in a way that encompasses all of the rows that the managers should be able to see. In other words, combine (OR together) the SASUSERS filter and the Managers filter.

CHAPTER

6

Permissions on Folders

Using Custom Folders to Manage Access

To use the metadata authorization layer to manage access to content (such as reports, data, information maps, and stored processes), create folders and set permissions so that each folder offers appropriate access. Items within each folder inherit permission settings from that folder.

Note: If you aren't familiar with the SAS folder structure, see the chapter "Working with SAS Folders" in the *SAS Intelligence Platform: System Administration Guide*. △

Baseline ACTs

One approach to establishing protections on custom folders is to create a few general-use ACTs and apply one or more of those ACTs on the **Authorization** tab of any folder that you need to secure. To grant access back to a particular group, supplement a folder's baseline ACT settings by adding grants on that folder's **Authorization** tab.

The examples in this chapter use the following baseline ACTs:


Hide — prevents visibility (for users who aren't in the SAS Administrators group).


Protect — prevents updates, deletions, and contributions (by users who aren't in the SAS Administrators group).


LimitData — prevents access to data through the OLAP server, information maps, and the metadata LIBNAME engine (for all restricted users).

Each ACT's name describes the effect of applying that ACT to an item that has no explicit ☑ or ACT ☑ (green) settings. The following tables document the permission pattern for each of these ACTs:

Table 6.1 The Hide ACT

Group	Permission Pattern[1]	
PUBLIC	Denial	ReadMetadata
SAS Administrators	Grant	ReadMetadata
SAS System Services	Grant	ReadMetadata

1 Gives SAS Administrators and service identities exclusive read access to metadata.

Table 6.2 The Protect ACT

Group	Permission Pattern[1]	
PUBLIC	Denial	WriteMetadata, WriteMemberMetadata, CheckInMetadata, Write, Administer
SAS Administrators	Grant	WriteMetadata, WriteMemberMetadata, CheckInMetadata, Write, Administer, ReadMetadata

1 Gives SAS Administrators exclusive write access to metadata.

Table 6.3 The LimitData ACT

Group	Permission Pattern[1]	
PUBLIC	Denial	Read

1 Prevents all restricted users from accessing data (through information maps, the OLAP server, and the metadata LIBNAME engine).

Each baseline ACT reduces a particular type of access down to a minimal level. In the Hide and Protect ACTs, the grants to SAS Administrators preserve standard administrative access so that members of that group can manage all metadata (for alternatives, see "Separated Administration" on page 83). In the Hide ACT, the grant to SAS System Services preserves necessary service access (the SAS Trusted User, who is a member of that group, reads certain metadata on behalf of all users). The LimitData ACT is unusual in that the pattern consists of a single setting. This chapter uses this ACT for consistency and in case at some future point you want to give a restricted user access to all data.

To create the baseline ACTs:

1. Log on to SAS Management Console as a registered user (anyone who has a well-formed user definition). Select the **Plug-ins** tab.
2. Expand **Authorization Manager**, right-click **Access Control Templates**, and select **New Access Control Template**.
3. On the **General** tab, enter the ACT name (**Protect**, **Hide**, or **LimitData**).

   Note: If you previously created the Protect ACT to protect server definitions, just verify that the pattern on that ACT is correct. △

4 On the **Permission Pattern** tab, define the settings this ACT will provide:

   a Click **Add**. In the Add Users and Groups dialog box, clear the **Show Users** check box. Move **PUBLIC** and any other participating identities (**SAS Administrators** and **SAS System Services**) to the **Selected Identities** list box. Click **OK**.

   b On the **Permission Pattern** tab, define explicit ☑ settings as specified in the preceding tables. Remove the automatically created grants of ReadMetadata permission except as specified.

   Note: Make sure you are on the **Permission Pattern** tab and not the **Authorization** tab. △

5 On the **Authorization** tab, protect the ACT that you are creating. Either apply the Protect ACT or add explicit ☑ settings that deny WriteMetadata permission to PUBLIC and grant WriteMetadata permission to the SAS Administrators group.

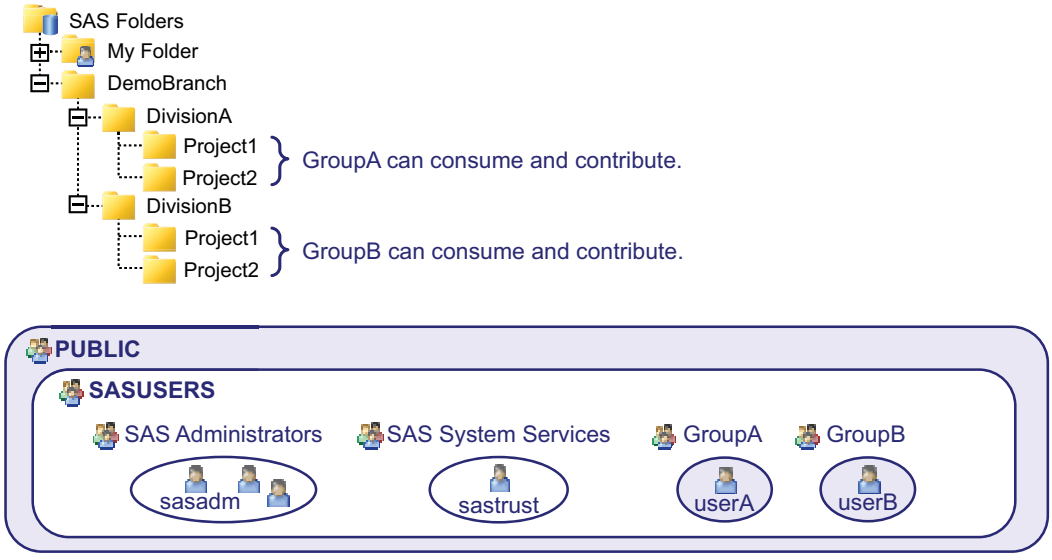
   Note: If the **Users and Groups** list box on the ACT's **Authorization** tab is empty, click **OK** to save the ACT. Then, right-click the new ACT, select **Properties**, and select the **Authorization** tab again. △

6 Click **OK**. Repeat until you have created all three baseline ACTs.

Demonstration: Departmental and Project Separation

This example creates a secured custom branch with mutually exclusive access for two divisions. In this example, the project level folders are for only organizational purposes (there are no access distinctions at the project level). The following figure depicts the goals and group structure.

Figure 6.1 Demonstration: Folder and Group Structure



To set this up, complete these steps:

1 Log on to SAS Management Console as someone who has user administration capabilities and is a member of SAS Administrators (for example, sasadm@saspw). Select the **Plug-ins** tab.

2 Create two temporary groups:

   a Right-click **User Manager** and select **New ▶ Group**.
   b On the **General** tab, enter the name **GroupA**. Click **OK**.
   c Repeat steps 2a-b to create a group named **GroupB**.

3 Create two temporary users:

   a Right-click **User Manager** and select **New ▶ User**.
   b On the General tab, enter the name **userA**.
   c On the **Groups and Roles** tab, move **GroupA** to the **Member of** list box.
   d On the **Accounts** tab, click **Create Internal Account**. In the New Internal Account dialog box, enter and confirm a simple password (for example **123456**). Click **OK** to save the new account.
   e Notice that the new account now appears at the bottom of the **Accounts** tab. Click **OK** to save the new user.
   f Repeat steps 3a-e to create **userB** (who should be in GroupB).

4 Create the folder structure:

   a On the **Folders** tab, right-click the root folder  and select **New ▶ Folder**. Create a new folder named **DemoBranch**.
   b Add subfolders under DemoBranch until it looks like the structure in the preceding display.

5 Protect the top of your custom branch:

   a Right-click **DemoBranch** and select **Properties**.
   b On the **Authorization** tab, click **Access Control Templates**.
   c In the Add and Remove Access Control Templates dialog box, move the **Protect** and **LimitData** ACTs to the **Currently Using** list box (you have to expand the **Foundation** node to get to the ACTs).

      Note: For instructions for creating the ACTs, see "Baseline ACTs" on page 71. △

   d Click **OK** to return to the **Authorization** tab. Review the revised settings. Notice that PUBLIC and SAS Administrators have some green settings ☑. The green settings come from the ACTs that you applied.

6 On the **DivsionA** folder:

   a Apply the Hide ACT.
   b Add GroupA to the **Authorization** tab and give them explicit ☑ grants of ReadMetadata, WriteMemberMetadata, and Read permissions.

7 On each project folder below **DivisionA**:

   a Apply the Protect ACT.
   b Give GroupA an explicit ☑ grant of the WriteMemberMetadata permission.

8 Repeat steps 6 and 7 for the DivisionB folders and subfolders (assigning the grants to GroupB). The following table summarizes the protections for the first four folders:

Table 6.4 Demonstration: Permission Settings

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
DemoBranch	Protect LimitData	
DivisionA	Hide	GroupA: +RM, +WMM, +R
Project1	Protect	GroupA: +WMM
Project2	Protect	GroupA: +WMM

9 Test the protections:

- **a** Log on as userA@saspw. On the **`Folders`** tab, notice which folders in the DemoBranch are visible to you. Right-click each folder and notice where you can add content (where the **`New Folder`** and **`New Stored Process`** actions are available) and where you can't.
- **b** Log on as userB@saspw and repeat the same checks.

10 To clean up, log on with the identity that you used in step 1. On the **`Plug-ins`** tab, delete the temporary groups and users (under **`User Manager`**). On the **`Folders`** tab, delete the **`DemoBranch`** and each user's **`My Folder`** (under **System ▸ Users**).

Here are some key points about this example:

- ReadMetadata permission flows into the DemoBranch from its parent folder. In all of the examples in this chapter, the immediate parent to the DemoBranch is visible to all registered users (SASUSERS). This is a standard setting. Unless you apply the Hide ACT to the top of your custom branch, the SASUSERS grant of ReadMetadata permission flows into your branch.
- The ability to create, manage, and delete content is shut off at the top (by the Protect ACT on the DemoBranch). This constraint flows throughout the tree (except where you add supplemental grants of WriteMemberMetadata permission to specific functional groups on specific folders). Users don't add content or access data in the DemoBranch, so no supplemental grants are needed on that folder.
- Because you want to prevent members of GroupB from seeing the DivisionA branch, you apply the Hide ACT on that branch. You then add supplemental grants for GroupA to restore their access.
- On DivisionA's project folders, you apply the Protect ACT to override GroupA's inherited grant of WriteMetadata permission (GroupA's grant of WriteMemberMetadata permission on DivisionA becomes an inherited grant of WriteMetadata permission on the immediate children of DivisionA). This prevents members of GroupA from renaming, deleting, or changing permissions on each project folder. You then add a supplemental grant of WriteMemberMetadata permission so that members of GroupA can contribute content in the project folders.
- On DivisionA's project folders, you don't apply the Hide ACT, because the ReadMetadata access that flows in from the DivisionA folder is appropriate. In this example, the requirement is that all members of GroupA should be able to access content throughout the DivsionA branch.

□ Notice that it isn't necessary to use separate folders for each type of object.

Note: Any content contributors who register cubes must have WriteMetadata permission on the OLAP schema . By default, the schema is in the **`SAS Folders/ Shared Data/ SASApp - OLAP Schema`** folder. △

Variation 1: Add Subgroups, Designate Content Creators

This example eliminates the project folders and introduces the following requirements:

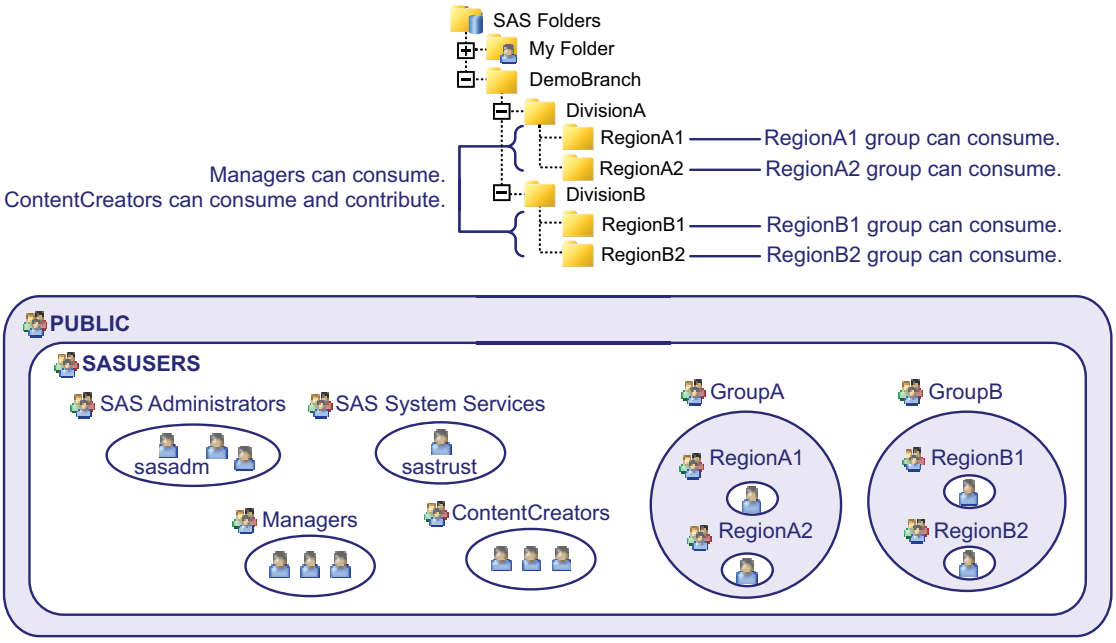
□ Regional employees see only content for their region.

□ A central group of managers see all content.

□ A central group of content creators creates all content.

The following figure depicts the folder and group structure.

Figure 6.2 Variation 1: Folder and Group Structure



The following table lists the protections for the first four folders:

Table 6.5 Variation 1a: Permission Settings

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
DemoBranch	Protect LimitData	
DivisionA	Hide	GroupA: +RM ContentCreators: +RM Managers:+RM
RegionA1	Hide	RegionA1: +RM, +R ContentCreators: +RM, +R, +WMM Managers:+RM, +R
RegionA2	Hide	RegionA2: +RM, +R ContentCreators: +RM, +R, +WMM Managers:+RM, +R

Notice that you are repeating many of the same explicit settings on each region, and that this will be the case throughout the DemoBranch. For greater efficiency and more centralized control, create a custom ACT (called RegionLevel) that provides the supplemental grants for your content creators (RM, R, WMM) and your managers (RM, R). Remember to protect the ACT itself as explained in step five in "Baseline ACTs" on page 71.

The following table lists the protections for the first four folders:

Table 6.6 Variation 1b: Permission Settings (use a supplemental ACT)

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
DemoBranch	Protect LimitData	
DivisionA	Hide	GroupA: +RM ContentCreators: +RM Managers:+RM

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
RegionA1	Hide	RegionA1: +RM, +R RegionLevel
RegionA2	Hide	RegionA2: +RM, +R RegionLevel

If you decide to offer content at the division level and you want that content to be available to only managers, you might make these changes:

- Create a DivisionLevel ACT with grants for Managers (RM, R) and ContentCreators (RM, R, WMM). Apply that ACT to each division folder.

  Note: This is the same pattern that you use for the RegionLevel ACT, so you could instead simply use that ACT. In this example, you choose to create a separate ACT because you anticipate that the requirements for division-level access and region-level access might diverge in the future. △

- Apply the Protect ACT on each region folder (to take away the inherited grant of WriteMetadata permission that content contributors inherit from their division-level grant of WriteMemberMetadata permission).

  Note: If you choose to not do this, members of the content creators group can delete, rename, or change permissions for the region folders. △

The following table lists the protections for the first four folders:

Table 6.7 Variation 1c: Permission Settings (support division-level content)

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
DemoBranch	Protect LimitData	
DivisionA	Hide	GroupA: +RM DivisionLevel

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
RegionA1	Hide Protect	RegionA1: +RM, +R RegionLevel
RegionA2	Hide Protect	RegionA2: +RM, +R RegionLevel

Variation 2: Add Functional Separation

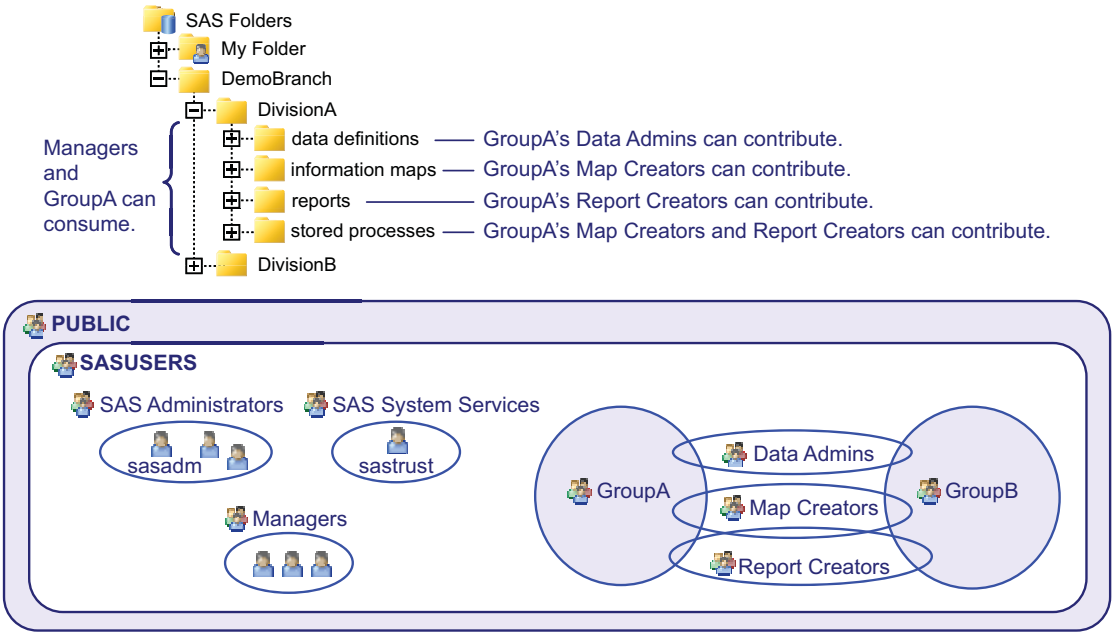
This example includes three custom groups that represent users who have specific job responsibilities (Data Admins, Map Creators, and Report Creators). Each division folder has separate subfolders for different types of content (data definitions, information maps, report definitions, and stored processes). Supplemental write access in each folder is limited to members of the appropriate functional groups as follows:

- Data administrators can work in each division's data definitions folder.
- Information map creators can work in each division's information maps folder.
- Report creators can work in each division's reports folder.
- Information map creators and report creators can work in each division's stored processes folder.

Note: Access for these functional groups is also limited by departmental separations. For example, only DivisionA's report creators (and SAS Administrators) can add, update, and delete items in DivisionA's reports folder. △

The following figure depicts goals and group structure.

Figure 6.3 Variation 2: Folder and Group Structure



The following table lists the protections for the first six folders:

Table 6.8 Variation 2a: Permission Settings (functional separation)

Folder	Protections	
	Baseline ACTs	**Supplemental Grants**
DemoBranch	Protect LimitData	
DivisionA	Hide	Managers: +RM, +R GroupA: +RM, +R
data definitions	Protect*	Data Admins: +WMM
information maps	Protect*	Map Creators: +WMM
reports	Protect*	Report Creators: +WMM
stored processes	Protect*	Map Creators: +WMM Report Creators: +WMM

* You don't strictly need the Protect ACT here, because protections flow through from the DemoBranch folder (and aren't interrupted by any supplemental grants of WM or WMM on higher level folders). However, you do need the supplemental grants of WMM on these folders, so you might choose to also apply the Protect ACT here for clarity.

Here are some details about this example:

- On DivisionA, users who aren't in the Managers group, GroupA, or SAS Administrators are locked out. Such users can't see or traverse DivisionA or any folders in the DivisionA branch. Managers and members of GroupA have supplemental grants of ReadMetadata and Read permissions on DivisionA; these grants flow through the entire DivisionA branch.

  Note: In order to view a DivisionA report, you need ReadMetadata permission and (in most cases) Read permission to the report's underlying components (such as information maps, stored processes, and data). △

- On each content type folder, one or more functional groups has a supplemental grant of WriteMemberMetadata permission. However, functional group access is also constrained by division-level ReadMetadata access. For example, even though all report creators have WriteMemberMetadata permission on the DivisionA reports folder, only those report creators who are members of GroupA can see the folder.
- Nothing in these settings prevents creation of a certain type of item. For example, a map creator can create a report and add that report to the map folder or the stored processes folder. Here are some techniques for increasing control:
  - You can use roles to limit the ability to create reports in SAS Web Report Studio.
  - You should limit the ability to register libraries, stored processes and OLAP schemas by managing WriteMetadata permission on application servers.
  - You can use the map accessibility check configuration option to limit the locations from which SAS Web Report Studio will use relational maps.
- You might choose to put the supplemental grants of WriteMemberMetadata permission in additional ACTs (such as MapCreators ACT, ReportCreators ACT) instead of using explicit settings on each folder. Using ACTs is more centralized, which is beneficial in accommodating later changes to the pattern. Of course, the more divisions you have, the more work it would be to update explicit settings, so the more valuable it would be to centralize settings with ACTs.
- Consider how you would handle an exception requirement. For example, to let a particular data administrator who isn't in GroupA contribute to the DivisionA data definitions folder, you might give that user a clear path of grants of ReadMetadata permission to the folder and supplemental grants of WriteMemberMetadata and Read permissions on the folder.

Key Points About the Baseline ACT Approach

- In general, it isn't necessary to add protection to the predefined folders (such as the root folder, each user's My Folder, and the System folder). These folders are protected by default.
- The preceding examples don't add grants of ReadMetadata permission on the DemoBranch folder because that folder's parent (the SAS Folders node) is visible to all registered users. The grant of ReadMetadata permission to SASUSERS on the **`SAS Folders`** node flows into the DemoBranch.
- To hide a branch, apply the Hide ACT to the relevant folder and grant back ReadMetadata permission to any groups who should have access. Remember that a denial of ReadMetadata permission on a folder prevents navigation to content and folders that are below the hidden folder (and that this can't be mitigated by a grant of ReadMetadata permission on a lower-level folder or other item).

- To enable a group to contribute content to a particular folder, give that group a grant of WriteMemberMetadata permission on that folder and consider applying the Protect ACT to each immediate subfolder.

  Note: This protects lower-level folders that inherit the contributing group's WriteMemberMetadata permission grant (from the parent folder) as a grant of WriteMetadata permission (on the child folder). If you don't protect a child folder, the contributing group can rename, delete, or change permissions on that folder. To enable content contributions to the child folder, grant WriteMemberMetadata permission on that folder (and consider repeating the denial of WriteMetadata permission on any subfolders that are below that folder). △

  Note: Any content contributors who register cubes must have WriteMetadata permission on the OLAP schema. By default, the schema is in the **`SAS Folders/ Shared Data/ SASApp - OLAP Schema`** folder. △

- If you want a group to access data through cubes, information maps, or the metadata LIBNAME engine, give that group a grant of Read permission on the folder that contains the relevant items. For any folder where you add a grant of Read permission, determine whether you want that grant to flow through to the subfolders. Apply the LimitData ACT to any immediate subfolders where you want to prevent data access. The navigational requirement for a clear path of ReadMetadata permission doesn't apply to the Read permission.
- The examples in this chapter ensure that content contributors have Read permission to the resources that they use. If you have a content contributor who isn't also a content consumer, you can choose to not provide Read permission. For example, if you give a user ReadMetadata permission but not Read permission to an information map, that user can still design the map (but can't perform tasks such as testing queries).
- You might be able to save some time by using this technique:
  1. Create one sub-branch (for example, the DivisionA branch in the preceding examples). The folders should have appropriate permission settings but contain no content.
  2. Copy that empty sub-branch to create the other branches (for example, to create the DivisionB folders). Change the folder name on the copy and update the settings as necessary. Often, you only have to remove and add a few supplemental grants.

  Note: If this technique proves useful, consider keeping an empty template branch that is visible only to administrators. You can use the template if you need to add more branches later. △

Further Considerations for Permissions on Folders

Consolidation of ACTs

In general, consolidation (using one pattern in all of the places where it is appropriate) is beneficial, because it simplifies management. However, it might be appropriate to maintain two ACTs that have the similar patterns in circumstances such as these:

- You anticipate that access requirements might diverge. For example, if you think you will eventually separate folder administration from server administration, you might create a SystemProtect ACT for items that aren't in the folder tree.

Note: Another example is that this chapter suggests that you use the baseline Protect ACT to protect servers and folders, but doesn't demonstrate another legitimate use of that ACT, to protect the ACTs themselves. This use was omitted from the introductory information about the baseline ACTs only in the interest of avoiding confusion in the initial discussion. Consider returning to the **`Authorization`** tab of each ACT, removing the explicit settings, and applying the Protect ACT. △

- You want to use a pattern that is similar to but not exactly the same as one of the predefined ACTs. For example, the baseline Hide ACT is not very different from the predefined Private User Folder ACT. We strongly recommend that you do not modify or delete the predefined ACTs, because these ACTs are an integral part of the protections that are set up for you during installation. The usage of each predefined ACT requires certain settings. Modifying the settings on a predefined ACT can compromise the security that that ACT provides.

Separated Administration

If you need to separate administration privileges by department, the approach in this chapter is not granular enough. If you don't want the SAS Administrators group to have universal access, consider creating parallel sets of baseline ACTs.

For example, to separate administration for an East region and a West region, you might create ACTs such as Hide_East, Hide_West. In each baseline ACT pattern, you would replace the SAS Administrators group with a narrower administrative group (for example, East_Admins, West_Admins). The denials to PUBLIC and grants to the SAS System Services group would not change. Any unrestricted users can still access everything.

End Users, Folders, and Permissions

Proper use of the WriteMetadata and WriteMemberMetadata permissions protects a folder structure. Keep in mind that end users can affect access to content as follows:

- A user who can update an item can add settings on that item. You can't prevent this by limiting the availability of SAS Management Console because users can set permissions in other applications (for example, SAS Information Map Studio, SAS OLAP Cube Studio, SAS Data Integration Studio, SAS Enterprise Guide, and the SAS Add-In for Microsoft Office).
- A user who can contribute items to a folder can also add subfolders below that folder. You can't prevent this by limiting the availability of SAS Management Console because users can add folders in other applications (for example, SAS Information Map Studio, SAS OLAP Cube Studio, SAS Data Integration Studio, SAS Enterprise Guide, and the SAS Add-In for Microsoft Office).
- If you give someone CheckInMetadata permission on a folder, that person can update or delete the folder (through change management activities), as well as check in content to that folder. Change management is an optional feature that is available only for SAS Data Integration Studio.

CHAPTER

7

Permissions on Servers

Managing Access to Server Definitions

Server metadata is not automatically protected. We recommend that you limit access to server metadata as follows:

- To protect definitions, use the WriteMetadata permission to limit the ability to update or delete server metadata.
- If you want to limit use of a particular server, use the ReadMetadata permission to hide that server's definition.

Protecting Server Definitions

In the initial configuration, the Server Manager capability is available to only the SAS Administrators group. This provides some protection for server definitions. For greater security, use permissions to define access as follows:

- The SAS Administrators group should be able to administer, update, and delete all server definitions.
- Users who assign libraries, stored processes, or an OLAP schema to an application server must have WriteMetadata permission for that application server.
- No other users should be able to update or delete server definitions.

To efficiently set the permissions, create and use an access control template (ACT). This enables you to avoid individually setting the same grants and denials on multiple pieces of server metadata. If you want to enable regular users to assign libraries, stored processes, or an OLAP schema to a particular application server, supplement the ACT settings with a grant of WriteMetadata permission on that application server. The following instructions are one way to set the permissions:

1. Log on to SAS Management Console as a member of the SAS Administrators group (for example, sasadm@saspw). Select the **Plug-ins** tab.
2. (Optional) To examine the current settings:
   a. Expand **Server Manager**, right-click the **SASApp** application server  and select **Properties**.
   b. On the **Authorization** tab, select **SASUSERS**. Notice that this group (which includes all registered users) has both ReadMetadata and WriteMetadata

permissions. The application server inherits these grants from the standard repository-level settings. Click **OK** to close the dialog box.

3 To create the ACT:

a Expand **Authorization Manager**, right-click **Access Control Templates**, and select **New Access Control Template**.

b On the **General** tab, enter a name such as **Protect**.

Note: If you already have a Protect ACT, verify that the pattern on that ACT's **Permission Pattern** tab is correct and that settings on that ACT's **Authorization** tab protect the ACT itself. Then, proceed to step 4. △

c On the **Permission Pattern** tab, define settings that you want to apply to all server metadata:

i Click **Add**. In the Add Users and Groups dialog box, clear the **Show Users** check box. Move **PUBLIC** and **SAS Administrators** to the **Selected Identities** list box and click **OK**.

ii On the **Permission Pattern** tab:

- Select **PUBLIC** and set the denials in the following table
- Remove the grant of ReadMetadata permission for **PUBLIC** by clicking the already selected grant ReadMetadata check box. This clears that check box so that the pattern neither grants nor denies this permission to PUBLIC.

  Note: A grant of ReadMetadata permission is automatically added for each user or group that you add to a **Permission Pattern** tab. The purpose of the Protect ACT is to limit the ability to update or delete items, not to change the visibility of items, so it is important to undo the automatically created grant of ReadMetadata permission for PUBLIC. Do not instead add a denial of ReadMetadata permission for PUBLIC. That denial would interrupt the grant of this permission for SASUSERS that flows through from the repository ACT. △
- Select **SAS Administrators** and set the grants in the following table. For this group, you can leave the automatic grant of ReadMetadata permission in place.

Table 7.1 The Protect ACT

Group	Permission Pattern[1]	
PUBLIC	Denial	WriteMetadata, WriteMemberMetadata, CheckInMetadata, Write, Administer
SAS Administrators	Grant	WriteMetadata, WriteMemberMetadata, CheckInMetadata, Write, Administer, ReadMetadata

1 Gives SAS Administrators exclusive write access to metadata. To increase reusability, this pattern includes extra permissions.

d On the **Authorization** tab, add explicit ☑ settings to protect the ACT that you are creating:

Note: If the **Users and Groups** list box on the ACT's **Authorization** tab is empty, click **OK** to save the ACT. Then, right-click the new ACT, select **Properties**, and select the **Authorization** tab again. △

- Select **PUBLIC** and deny WriteMetadata permission. Leave the indirect ☑ (gray) ReadMetadata setting in place.

- Select **SAS Administrators** and grant WriteMetadata permission. Leave the indirect ☑ (gray) ReadMetadata setting in place.

e Click **OK**.

4 (Optional) If you want to allow some nonadministrators to assign libraries, stored processes, or an OLAP schema, and you don't already have a group that represents those users, create a new custom group.

Note: As an alternative to creating a group for only this purpose, you can skip this step and instead assign the permissions to specific users in step 5c. △

a Right-click **User Manager** and select **New ▶ Group**.
b On the **General** tab, enter a name such as **Data Admins**.
c On the **Members** tab, move users (or groups) to the **Selected Identities** list box.
d Click **OK** to save the new group. You will grant WriteMetadata permission to this group in step 5c. This group doesn't participate in the ACT's pattern because this group needs WriteMetadata permission on only some of the target objects.

5 To set the protections:

a Expand **Server Manager**, right-click the first application server  (if this is your first pass) and select **Properties**.
b On the **Authorization** tab, click **Access Control Templates**. In the Add and Remove Access Control Templates dialog box, move the **Protect** ACT to the **Currently Using** list box (you have to expand the **Foundation** node to get to the ACT). Click **OK** to return to the **Authorization** tab.

Note: Review the revised settings. Notice that SASUSERS is now denied WriteMetadata permission and that PUBLIC and SAS Administrators have some green settings ☑. The green settings come from the **Protect** ACT. △

c (Optional) If the current item is an application server  to which nonadministrators will assign libraries, OLAP schemas, or stored processes, click **Add**, add one or more identities to the **Authorization** tab, and give each of those identities an explicit ☑ grant of WriteMetadata permission. For example, you might assign the grant to a group (such as GroupA) or to individual users.

Note: Don't extend WriteMetadata access to the **SASMeta** application server, because that server should be used for only a few specialized administrative tasks. △

d Click **OK** to save the settings for this object.
e Repeat steps 5a-d for every immediate child of **Server Manager** (except the table server). Immediate children include objects such as other application servers, third-party servers, the share server, the content server, and spawners.
f Apply the **Protect** ACT to every logical server  that is under an application server  where you added an individual grant of WriteMetadata permission (for SASUSERS or a custom group such as **Data Admins**).

Note: This protects lower-level server metadata that would otherwise inherit the nonadministrator's WriteMetadata grant from the application server. This also prevents nonadministrators who have WriteMetadata permission on the application server from deleting that application server. △

The following table summarizes typical protections.

Table 7.2 Example: Protecting Server Definitions

Object	Added Settings	
	ACT	**Supplemental Grants**
SASApp	Protect	DataAdmins: +WM
Each logical server inside SASApp	Protect	(none)
SASMeta	Protect	(none)
Other immediate children of **`Server Manager`***	Protect	(none)

* Except the table server, which does not need any changes.

Hiding Server Definitions

Initially, all registered users can use almost all servers. In general, the SASUSERS group has the ReadMetadata permission on all server metadata.

Note: The logical workspace server within the **`SASMeta`** application server has limited availability. This avoids the potentially confusing situation in which a regular user is offered the **`SASMeta`** server context in applications such as SAS Enterprise Guide. **`SASMeta`** should be used only as instructed in a few specialized administrative tasks. △

These initial settings are appropriate until you want to allow only certain users to use certain servers. Here are some reasons why you might choose to limit use of a server:

- to create different levels of host access from a particular server component , logical server , or application server . This is one part of the configuration.
- to direct high priority users to a server on hardware that offers superior performance.
- to direct power users to a server with settings that offer advanced options.
- to enable users in SAS Information Map Studio to use a standard workspace server even when a logical pooled workspace server is present.

If you choose to limit use of a server, preserve access as follows:

- Make sure that the SAS System Services group has ReadMetadata permission for server metadata. This enables the SAS Trusted User to see server definitions. This is necessary because the object spawner uses the SAS Trusted User to discover and read all server metadata.

  Note: Users should not be members of the SAS System Services group. This group is for service identities. In the standard configuration, the only member of this group is the SAS Trusted User. △

- Make sure that the SAS General Servers group has ReadMetadata permission for server metadata. This enables the metadata identity of the launched server to see the server definition. This is a requirement for stored process servers and pooled workspace servers. This isn't a requirement for standard workspace servers.

  Note: Users should not be members of the SAS General Servers group. This group is for service identities. In the standard configuration, the only member of this group is the SAS Trusted User. △

- In general, the SAS Administrators group should have ReadMetadata permission for all server metadata.
- In order to use a server-side pooled workspace server, a user must have the ReadMetadata permission for that server and for a standard logical workspace server definition within the same application server. An application server that contains only a server-side pooled workspace server is not accessible to users through SAS clients.
- Any user who will use a particular server needs ReadMetadata permission for that server, with these exceptions:
  - The requirement for ReadMetadata permission doesn't apply to requests to use a client-side pooled workspace server. A user can use a client-side pooled workspace server even if that user can't see that server definition.
  - The requirement for ReadMetadata permission isn't enforced if the **Use Server Access Security** check box on a logical server's **Options** tab is present and not selected. This check box should always be selected.

To efficiently set the permissions, create an access control template (ACT) that includes the core grants and denials that you would use when you hide any server. To enable selected users to use a particular server, supplement the ACT settings with a grant of ReadMetadata permission on that server. The following instructions explain one way you can set these permissions:

1. Log on to SAS Management Console as a member of the SAS Administrators group (for example, sasadm@saspw). Select the **Plug-ins** tab.
2. (Optional) To examine the current settings:
   a. Expand **Server Manager**, right-click the server or server component that you are limiting use of and select **Properties**.
   b. On the **Authorization** tab, select **SASUSERS**. Notice that this group (which includes all registered users) has ReadMetadata permission. The application server inherits the grant from the standard repository-level settings. Click **OK** to close the dialog box.
3. To create the ACT:
   a. Expand **Authorization Manager**, right-click **Access Control Templates**, and select **New Access Control Template**.
   b. On the **General** tab, enter a name such as **HideServer**.
   c. On the **Permission Pattern** tab, define general settings for hiding servers:
      i. Click **Add**. In the Add Users and Groups dialog box, clear the **Show Users** check box. Move **PUBLIC**, **SAS Administrators**, **SAS General Servers**, and **SAS System Services** to the **Selected Identities** list box and click **OK**.
      ii. On the **Permission Pattern** tab:
         - Select **PUBLIC** and add a denial of ReadMetadata permission.
         - Make sure that the other three groups each have a grant of ReadMetadata permission.

Table 7.3 The HideServer ACT

Group	Permission Pattern[1]	
PUBLIC	Denial	ReadMetadata
SAS Administrators	Grant	ReadMetadata
SAS General Servers	Grant	ReadMetadata
SAS System Services	Grant	ReadMetadata

1 Gives SAS Administrators and service identities exclusive read access to metadata.

This pattern, when applied to a standard workspace server, grants a little more access than is strictly necessary. For a standard workspace server, the SAS General Servers group doesn't need ReadMetadata permission. If you want to avoid this, consider omitting the SAS General Servers group from this ACT and remembering to add an explicit grant for this group when you are hiding a stored process server or pooled workspace server.

d On the **Authorization** tab, protect the ACT that you are creating. Either apply the Protect ACT or add explicit ☑ settings that deny WriteMetadata permission to PUBLIC and grant WriteMetadata permission to the SAS Administrators group.

Note: If the **Users and Groups** list box on the ACT's **Authorization** tab is empty, click **OK** to save the ACT. Then, right-click the new ACT, select **Properties**, and select the **Authorization** tab again. △

e Click **OK**.

4 (Optional) If you don't already have a group that represents the users who will use the server, create a new custom group.

a Right-click **User Manager** and select **New ▶ Group**.
b On the **General** tab, enter a name such as *GroupA*.
c On the **Members** tab, move users (or groups) to the **Selected Identities** list box.
d Click **OK** to save the new group. You will grant ReadMetadata permission to this group in step 5c. This group doesn't participate in the general pattern because this group doesn't need ReadMetadata permission on all servers.

Note: As an alternative to creating a group for only this purpose, you can skip this step and instead assign the permissions directly to specific users in step 5c. △

5 To set the permissions:

a Under **Server Manager**, right-click the server that you are limiting use of and select **Properties**.
b On the **Authorization** tab, click **Access Control Templates**. In the Add and Remove Access Control Templates dialog box, move the **HideServer** ACT to the **Currently Using** list box (you have to expand the **Foundation** node to get to the ACT). Click **OK** to return to the **Authorization** tab.

Note: If the **Currently Using** list already includes another ACT (such as the Protect ACT), don't remove that assignment. △

Note: Review the revised settings. Notice that SASUSERS is now denied ReadMetadata permission and that PUBLIC and SAS Administrators have some green settings ☑. The green settings come from the HideServer ACT. △

c Click **Add**, add one or more identities to the **Authorization** tab, and give each of those identities an explicit ☑ grant of ReadMetadata permission. For example, you might assign the grant to a group (such as GroupA) or to individual users.
d Click **OK** to save the settings for this object.

For example, the following table summarizes settings that you might add to provide mutually exclusive access to two server components beneath a standard workspace server that is configured for SAS token authentication:

Table 7.4 Example: Hiding Server Definitions

Object	Added Settings[1]	
	ACT	**Supplemental Grants**
SASApp - ServerA	HideServer	GroupA: +RM
SASApp - ServerB	HideServer	GroupB: +RM

1 These settings don't determine which of the users who can see the server can also update or delete the server. See the preceding topic, "Protecting Server Definitions."

Note: Someone who has ReadMetadata permission for both ServerA and ServerB (for example, members of the SAS Administrators group) uses the first server in the object spawner's list of servers. △

6 If you are limiting use of a logical server or server component, ensure that the **`Use Server Access Security`** check box on the logical server's **`Options`** tab is selected. If the check box is present and not selected, requirements for ReadMetadata permission for that server and its components are not enforced. This option affects only enforcement of the ReadMetadata permission.

Note: User access to a client-pooled workspace server is determined by the user's puddle group memberships, not by permissions on the server definition. △

CHAPTER

8

BI Row-Level Permissions

About BI Row-Level Permissions

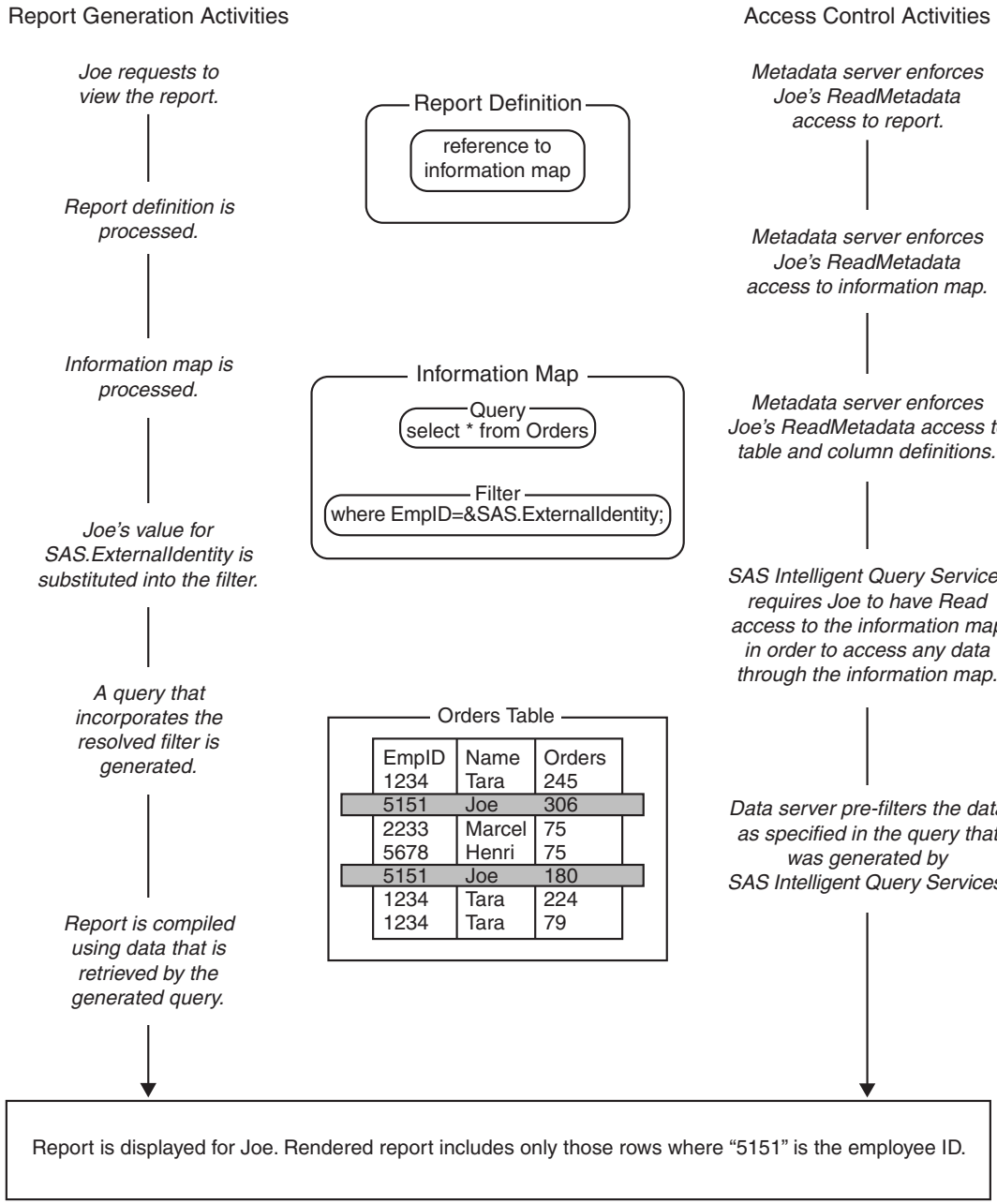
BI row-level permissions limit access to SAS data and third-party relational data when it is accessed through an information map. The initials BI indicate that this is a business intelligence feature. BI row-level permissions are defined in information maps, mediated and enforced by SAS Intelligent Query Services, and surfaced when reports are viewed in applications such as SAS Web Report Studio.

CAUTION:
BI row-level permissions are defined within information maps, so these constraints do not provide comprehensive security for the underlying data sources. A user who accesses the data directly is not subject to the filters that are defined within information maps. BI row-level permissions provide filtering whenever a SAS data set or third-party relational data is accessed through an information map. However, comprehensive security that incorporates this filtering requires a high-security configuration of SAS Web Report Studio. See “Preliminary Tasks” on page 95. △

The following figure depicts how BI row-level permissions are incorporated when a report is generated. In the figure, a user requests access to a report that includes data

for which row-level permissions have been defined dynamically. See “Identity–Driven Properties” on page 66. For each step of the report-generation process, the figure depicts the access control activities in the metadata layer.

Figure 8.1 Report-Generation Process



The overall flow is the same as for any other report: the report definition and underlying information map are processed, a query is generated to retrieve the data, and the report is displayed. These are the row-level security aspects of the process:

- *The information map includes a filter that is assigned to a particular metadata identity.* This example uses an identity-driven property in a filter that is based on each group member's employee ID. The filter is assigned to a group to which Joe

belongs. At run time, SAS Intelligent Query Services uses information from the metadata repository to substitute Joe's employee ID into the filter. The resolved, user-specific form of the filter is incorporated into the generated query. The filter is used to screen the target table before the rest of the generated query runs.

- *The target data includes information that corresponds to the filter.* In this example, the corresponding information consists of user-specific employee ID values in the EmpID column within the Orders table. The data server uses these values to filter the data as specified in the query that was generated by SAS Intelligent Query Services.

Filtering Techniques for BI Row-Level Permissions

To ensure that target data is prescreened by a filter before any other criteria are applied, incorporate that filter into an information map as a prefilter. Use one of these methods:

- To use a filter as an *authorization-based prefilter*, assign it to users or groups as part of an information map's access controls. The filter is evaluated as a permission condition. See "Authorization Decisions" on page 62.
- To use a filter as a *general prefilter*, attach it to an information map as part of the information map's properties. The filter applies to everyone. The filter functions as an additional restriction and operates independently of any access controls that might grant broader access.

Both methods support dynamic filters. See "Identity–Driven Properties" on page 66.

Table 8.1 Summary of BI Row-Level Filtering Techniques

Filter	Method	Typical Usage
Identity-driven	Authorization-based	Per-person access distinctions for every member of a particular group. You create one identity-driven filter and assign it to a group. For example, *each user in GroupA can see his or her own salary information*.
	General	Per-person access distinctions for everyone. You create one identity-driven filter and attach it directly to the information map (not to any particular user or group). For example, *everyone can see his or her own salary information*.
Static	Authorization-based	A few distinct subsets. You create a different filter for each subset and assign each filter to a different group. For example, *each user in GroupA can see sales totals for the West region*; *each user in GroupB can see sales totals for the East region*.
	General	One fixed subset for everyone. This is not a row-level method because it does not yield different results for different users. For example, *everyone can see global sales totals*.

How to Implement BI Row-Level Permissions

Preliminary Tasks

1 If you need comprehensive security, set up the high-security configuration of SAS Web Report Studio. This prevents regular users from circumventing row-level

filters by accessing the target tables directly (without going through the information map that enforces the filters). In the high-security configuration, SAS Web Report Studio uses a privileged account to fetch the data. Regular users have only mediated access to the target tables.

In the following environments, you might not need the high-security configuration:

- prototype environments
- environments that do not have strict security requirements
- environments in which a firewall separates untrusted users
- environments in which untrusted users do not have the tools, knowledge, or operating system privileges to access files and metadata on the server tier

For instructions for setting up the high-security environment, see "Configure a Client-side Pooling Workspace Server to Enforce Row-Level Security" in the chapter "Configuring Client-side Pooling" in the *SAS Intelligence Platform: Application Server Administration Guide*.

2 Make sure that appropriate coarse-grained controls are in place for users.

Table 8.2 Metadata Layer Coarse-Grained Controls

Access	Target Table	Information Map
All rows	Grant Read, ReadMetadata	Grant Read, ReadMetadata
No rows	Deny Read, ReadMetadata	Grant[1] Read, ReadMetadata
Some rows	Grant Read, ReadMetadata	Grant Read[2], ReadMetadata

1 Set these grants if the "No rows" users access other tables through this information map.
2 For authorization-based prefilters, this must be an explicit grant.

Note: In the high-security configuration, only the restricted puddle account and IT staff who test queries in SAS Information Map Studio have physical access to the target table. If you use server-side pooling, only the server launch credential should have physical access to the target table. △

3 If the target data is in a third-party database, enable authentication to that server.

- Make sure that IT staff who test queries against the data from SAS Information Map Studio can authenticate to the DBMS server. For example, you can store individual or group DBMS credentials for those users, or require those users to interactively provide a DBMS user ID and password. See “How to Store Passwords for a Third-Party Server” on page 174.
- Make sure that regular users can access the DBMS server. For example, you can store individual or shared DBMS credentials for users as you did for the IT staff.

  Note: In the high-security configuration, you use a different approach so that physical access for regular users is mediated by a privileged service account. △

4 Plan how you will create the data subsets that will narrow grants of the Read permission for each user. Your choice will be affected by the number and type of access distinctions that you are making, the information that your data already contains, and your plans for enhancing your existing data to support row-level filtering. See “Filtering Techniques for BI Row-Level Permissions” on page 95.

Note: In a situation in which multiple filters (multiple distinct permission conditions) apply to a particular user as a result of the user’s group memberships,

the subset of data that is available to that user is determined by identity precedence. See Table 5.14 on page 69. △

Data Modeling

Overview and Examples

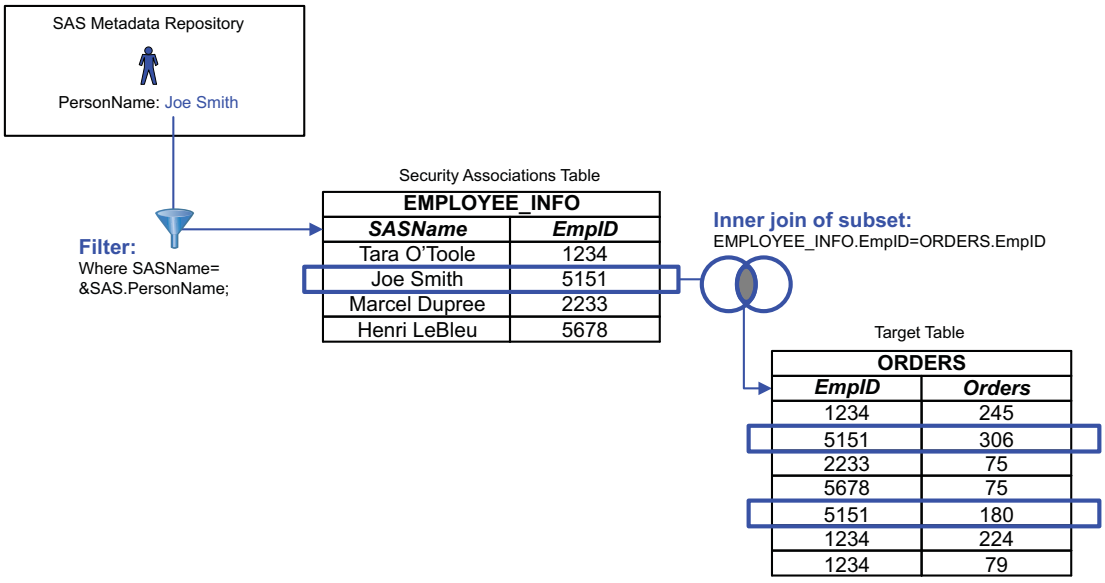
It is usually necessary to enhance existing data to include information that works with the filters that you want to use. For example, consider a four-person company with a flat organizational structure and a business requirement that each employee sees only his or her own order information. The order information is stored in this table:

Figure 8.2 Orders Example: Target Table

ORDERS	
EmpID	***Orders***
1234	245
5151	306
2233	75
5678	75
5151	180
1234	224
1234	79

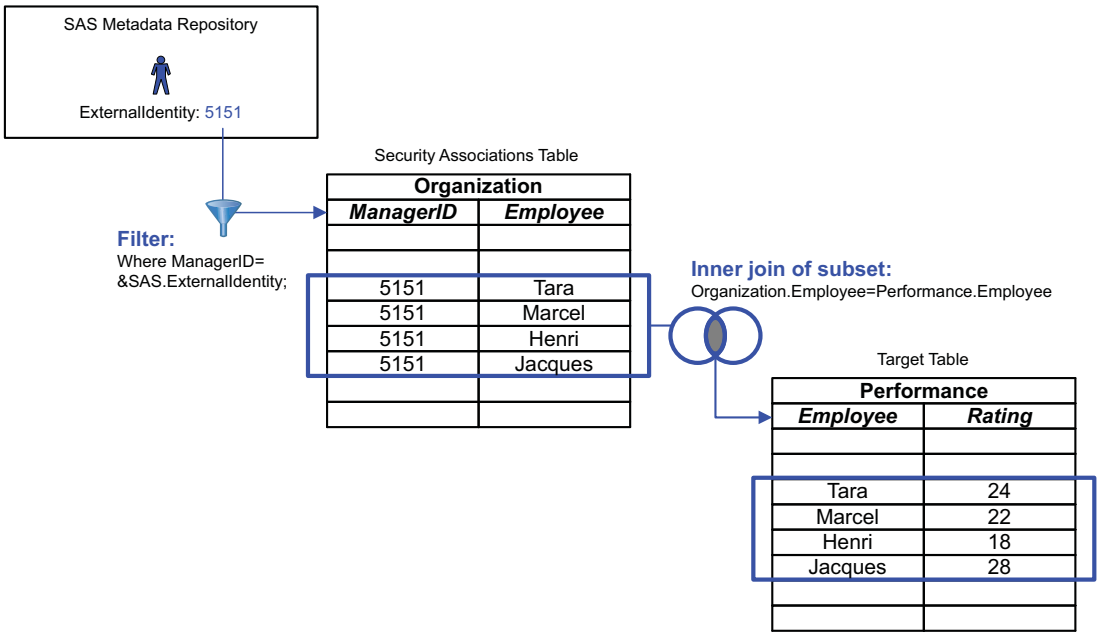
Assume that you didn't import users (so you don't have SAS.ExternalIdentity values in the metadata that correspond the EmpID values in the ORDERS table). To avoid setting up a different filter for each user, you decide to use the SAS.PersonName identity-driven property. Create a table that maps each user's PersonName (from the **`Name`** field on the **`General`** tab of the user's definition) to the user's employee ID. The following figure depicts how that table is used to prescreen the data for each user.

Figure 8.3 Orders Example: Data Model



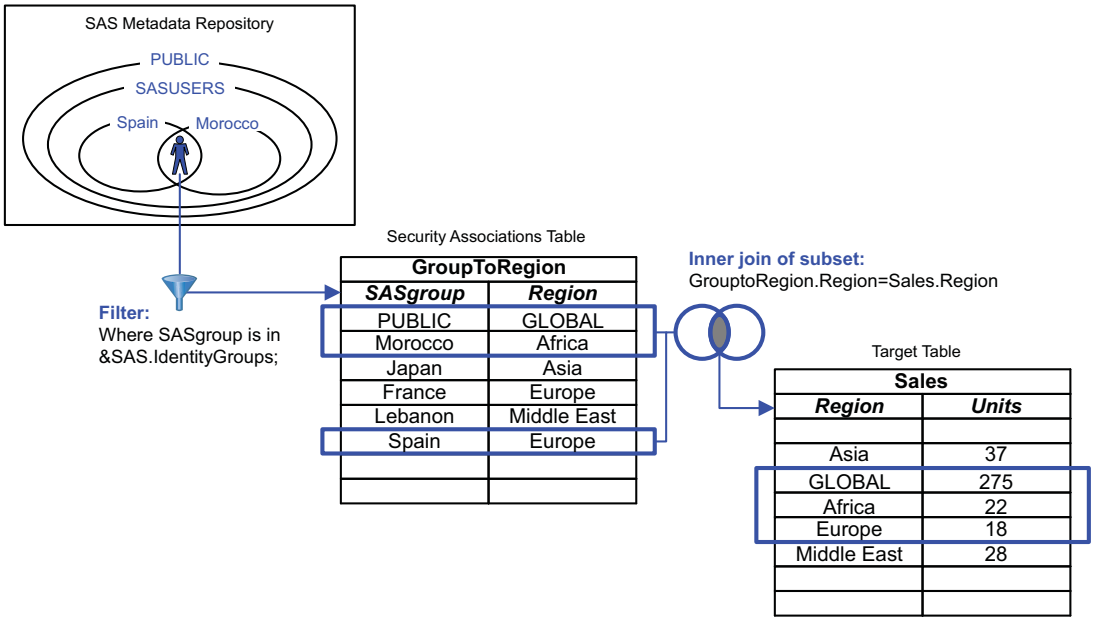
Another simple example is to subset employee performance information based on each manager's external identity value, as depicted in the following figure.

Figure 8.4 Performance Example: Data Model



Another example is to subset sales information by each salesperson's geographic responsibilities. Assume that there is a metadata group for each country and that some employees have responsibilities in multiple continents. The following figure depicts continent-level subsetting based on each salesperson's metadata group memberships.

Figure 8.5 Sales Example: Data Model



This approach provides aggregated retrieval and flattens the group structure. Each user gets all rows that are permitted for any groups that the user belongs to. To enable everyone to see the global totals, the security associations table includes a row that pairs the PUBLIC group with global totals.

Content of a Security Associations Table

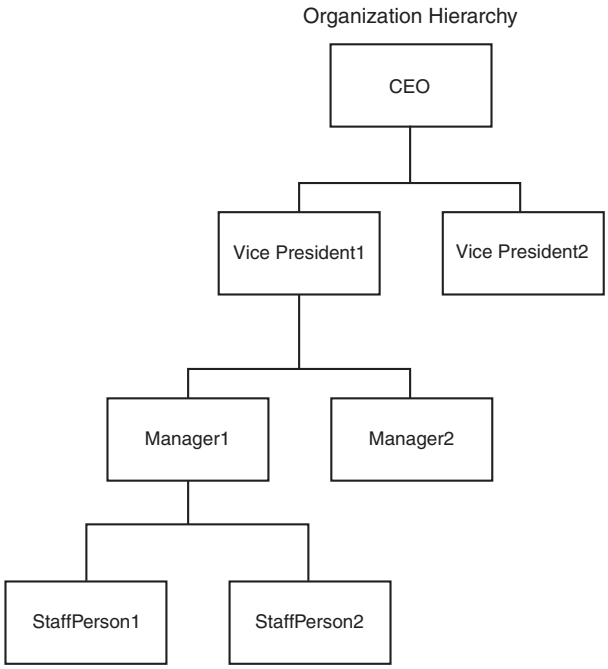
A security associations table is a type of table that documents the relationships between a user and some criterion on which you are making access distinctions. When access distinctions are based on each user's place within an organizational hierarchy, the security associations table must contain a representation of the reporting relationships within the organization. If access distinctions are based on some other criterion (such as each user's project assignments), then the security associations table should reflect that criterion.

Format of a Security Associations Table

BI row-level permissions do not require that the security associations table have a particular format. However, the format of a security associations table can affect filter performance. This topic describes a format that supports efficient hierarchy-based filtering. This format is useful for many common scenarios, because security policies are often hierarchical. For example, a typical business requirement is that a manager can see data for all of the employees that he or she manages either directly or indirectly.

The following figure depicts two ways to structure a security associations table that documents each user's place in a simple organizational hierarchy. The sparse version of the table includes only direct reporting relationships; information about indirect relationships must be derived. The fully articulated (or robust) version explicitly includes indirect reporting relationships along with direct reporting relationships; this is advantageous for query performance.

Figure 8.6 Representations of an Organizational Hierarchy



Security Associations Table (sparse)

Reporting Relationships	
Manager	***Employee***
CEO	Vice President1
CEO	VicePresident2
Vice President1	Manager1
Vice President1	Manager2
Manager1	Staff Person1
Manager1	Staff Person2

Security Associations Table (fully articulated)

Reporting Relationships		
Manager	***Employee***	
CEO	Vice President1	
CEO	VicePresident2	
CEO	Manager1	✓
CEO	Manager2	✓
CEO	Staff Person1	✓
CEO	Staff Person2	✓
Vice President1	Manager1	
Vice President1	Manager2	
Vice President1	Staff Person1	✓
Vice President1	Staff Person2	✓
Manager1	Staff Person1	
Manager1	Staff Person2	

The table that uses the fully articulated format explicitly includes not only the hierarchy's immediate parent-child relationships, but also every other ancestor-descendant association (such as grandparent-child and great grandparent-child). This facilitates simpler queries by eliminating the need to traverse the hierarchy to find all of the descendants of any particular node.

Creation and Maintenance of a Security Associations Table

This topic contains a general discussion about creating and managing a security association table for use with dimensional target data. BI row-level security does not require that target data conform to a particular structure. The description in this topic is for dimensional data, because that is a frequently used structure for query and reporting.

A security associations table is usually created as a new object by traversing an existing sparse table and filling in the indirect relationships to create a fully articulated (or robust) version of the table. If you do not have an existing sparse table, then you must create that object first.

Note: If you want to enhance an existing sparse table rather than creating a new table, you should first review current uses of the sparse table to determine whether the additional rows will negatively affect those uses. △

In most cases it is helpful to have an index on the column in the security associations table that is used for filtering. In some cases, factors such as the size of the security associations table or query optimization features in a particular data source might negate the need for this index.

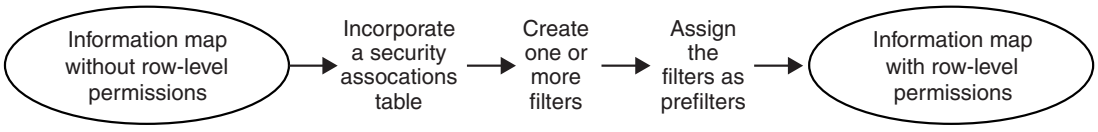
The security associations table must be maintained as security relationships change. This maintenance should be on a schedule that is appropriate for your environment. Typically, this maintenance is accomplished by a batch process (such as a nightly ETL process against the existing tables). In some cases, updates might be entered directly by an administrator.

Information Map Tasks

Overview

The following figure depicts the row-level permission aspects of information map design.

Figure 8.7 Information Map Design for Row-Level Permissions



The following topics provide generic instructions for each of these four tasks.

How to Add a Security Associations Table to an Information Map

In order to make the security relationship information that you added to the data model available for filtering, you must incorporate that information in an information

map. To enhance an existing information map to include a new security associations table:

1. In SAS Management Console, register the new security associations table in the metadata.
2. In SAS Information Map Studio:
   a. Add the table to your information map as a data source.
   b. On the **Relationships** tab, define an inner join that connects an identifier column in the security associations table with a corresponding column in the target table (or in an intermediate dimension).
   c. Make the security associations table a required table by performing these steps:
      i. Select **Edit** ▶ **Properties** ▶ **Information Map**, and then select the **Required Tables** tab in the Information Map Properties dialog box.
      ii. In the **Available tables** list, select the table that you are using as a security associations table. Use the arrow button to move the table to the **Required tables** list. Click **OK**.

Note: We recommend that you do not create data items from columns in the security associations table. Excluding these column references from the information map prevents their values from surfacing when reports are created in SAS Web Report Studio. △

How to Create an Identity-Driven Filter

To create a filter that is based on an identity-driven property, perform these steps in SAS Information Map Studio:

1. Open the information map, save the information map, and select the **Design** tab. Select **Insert** ▶ **New Filter** to open the New Filter dialog box.
2. Enter a name for the filter and select a character data item (or click **Edit Data Item** and use the expression builder to define a character item).
3. In the New Filter dialog box, from the **Enter value(s)** drop-down list, select **Derive identity values (for row-level permissions)**.

   The examples column shows what your values are for each property.
   - The SAS.PersonName value corresponds to the **Name** field on the **General** tab of your user definition. This is not always the same as the value of the **Display Name** field.
   - The SAS.IdentityGroupName value is usually blank and isn't often useful.
   - Although the SAS.IdentityGroups property displays only one value, this property actually returns a list of the groups and roles that you belong to.
   - The SAS.ExternalIdentity value is populated only in certain circumstances. See "External Identities" on page 226.
4. Select the row for the identity-driven property that you want to use in the filter. See "Identity–Driven Properties" on page 66.

   Note: Not all conditions support all identity-driven properties. Make sure that the selection in the **Condition** drop-down list is appropriate. △

   Note: If you use the IdentityGroups property, set the condition to **Is equal to** or **Is not equal to**. For this property, these conditions are converted to an IN (or, NOT IN) statement when the query executes. △
5. Select the **Hide from user** check box at the bottom of the **Definition** tab. This prevents the filter from being surfaced (and potentially removed) when reports are created in SAS Web Report Studio.

6 Click **OK**. The new filter is now available for use in the current information map.

How to Assign a Filter for Row-Level Permissions

To use a filter for security purposes, assign the filter as an authorization-based prefilter or a general prefilter.

To assign a filter as an authorization-based prefilter:

1 Open the information map, save the information map, and select **Tools ▸ Authorization** to open the Authorization dialog box.
2 In the **Users and Groups** list, select (or add) the identity that should be subject to the filter.
3 Make sure that the right user or group is selected. Add an explicit ☑ grant of the Read permission.
4 Click **Add Condition** to open the Row-Level Permission Condition dialog box.
5 In the **Selected filters** list (the right-hand panel), select the table that you are using as a security associations table.
6 In the **Available filters** list, select the filter and then use the arrow button to move the filter to the **Selected filters** list. Click **OK**.
7 In the Authorization dialog box, click **Close**.
8 To make your changes take effect, save the information map.

To assign a filter as a general prefilter:

1 Open the information map, save the information map, and select **Edit ▸ Properties ▸ Information Map**.
2 In the Information Map Properties dialog box, select the **General Prefilters** tab.
3 In the **Selected filters** list (the right-hand panel), select your security associations table.
4 In the **Available filters** list, select the filter and then use the arrow button to move the filter to the **Selected filters** list. Click **OK**.

Verification

In a high security configuration, final verification must be performed from within SAS Web Report Studio. This testing requires that you log on to that application using different accounts.

If you are using server-side pooling, or if you have physical access to the data, you can do some preliminary testing to check your filter logic from within SAS Information Map Studio. Before you test an information map from within SAS Information Map Studio, you should save the information map to ensure that all settings are applied. To test a filter that is based on an identity-driven property, use different accounts to log on to SAS Information Map Studio. To test other filters, temporarily assign the filters to your identity.

Note: If you are using server-side pooling, your results in SAS Information Map Studio should be the same as your results in SAS Web Report Studio. See “Choices in Workspace Server Pooling” on page 192. △

Example: Using BI Row-Level Permissions

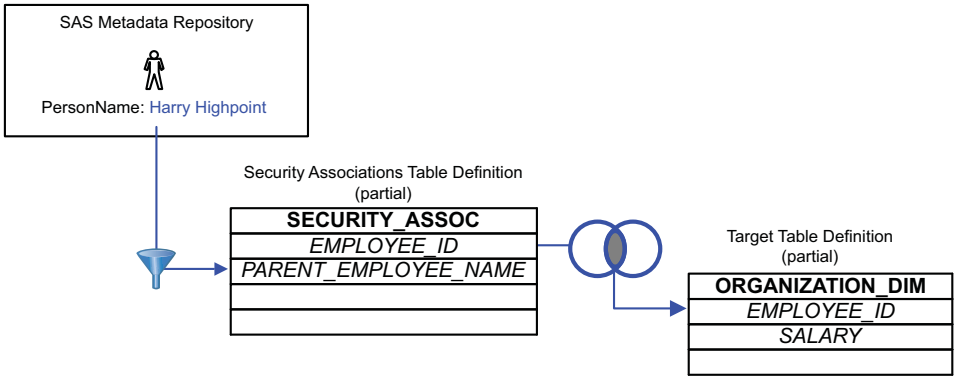
Assumptions and Data Model

This example demonstrates how a company could use row-level permissions to manage access to employee data. The example includes these assumptions:

- The target tables are registered in the metadata repository.
- Except where otherwise noted, users have Read permission for the information maps that they are using.
- The data model is a star schema that contains employee and customer data. The security associations table includes both direct and indirect reporting relationships.
- The company has set up a high-security configuration of SAS Web Report Studio as specified in "Preliminary Tasks" on page 95.

In this example, the business requirement is to enable managers to see salary information for their employees. One way to meet this requirement is to use the SAS.PersonName property. The following figure depicts this process for a requesting user who is a high-level manager in the organization.

Figure 8.8 Salary Example: Data Model



Each requesting user's PersonName is used to filter the security associations table. This yields a subset that includes only those rows with employees who report (directly or indirectly) to the requesting user. That subset is inner joined to the target table to limit retrieval of salary information.

Implementation and Testing

To implement the filtering for this example:

1. Create an information map that includes the necessary data and relationships.
   a. In SAS Information Map Studio, open a new information map.
   b. Insert the target table and the security associations table as data sources. In this example, the target table (ORGANIZATION_DIM) contains salary data, and the security associations table (SECURITY_ASSOC) contains a representation of the company's reporting relationships.

c On the **Design** tab, add the data items that you need from the ORGANIZATION_DIM table (insert the SALARY, EMPLOYEE_ID, and EMPLOYEE_NAME columns).

Note: It is a good practice to not create any data items from the SECURITY_ASSOC table. △

d On the **Relationships** tab, join the two tables on EMPLOYEE_ID.
e Save the new information map to an appropriate folder.
f Make SECURITY_ASSOC a required table:

i Select **Edit** ▶ **Properties** ▶ **Information Map**.
ii In the Information Map Properties dialog box, select the **Required Tables** tab.
iii In the **Available tables** list, select the SECURITY_ASSOC table.
iv Use the arrow button to move the table to the **Required tables** list, and then click **OK**.

2 Create a filter that subsets data by comparing each user's SAS.PersonName value to the PARENT_EMPLOYEE_NAME values in the security associations table.

a Select **Insert** ▶ **New Filter** to open the New Filter dialog box.
b Enter a name such as **byPersonName** for the filter, and then click **Edit Data Item**.
c In the Edit Expression dialog box, select **Character** from the **Type** drop-down list. On the **Data Sources** tab, navigate to **Physical Data** ▶ **SECURITY_ASSOC** ▶ **PARENT_EMPLOYEE_NAME**, and then click **Add to Expression**.
d Click **Validate Expression**, and then click **OK** twice.
e In the New Filter dialog box, from the **Enter value(s)** drop-down list, select **Derive identity values (for row-level permissions)**. A table of identity-driven properties becomes available. See "Identity–Driven Properties" on page 66.

Note: Make sure that the value in the **Condition** drop-down list is **Is equal to**. △

f In the table of properties, select the SAS.PersonName row.
g Click **OK**. The byPersonName filter is now available for use in the information map.

3 Assign the filter as a general prefilter:

a Select **Edit** ▶ **Properties** ▶ **Information Map**.
b In the Information Map Properties dialog box, select the **General Prefilters** tab.
c In the **Selected filters** box, select the SECURITY_ASSOCIATIONS table.
d In the **Available filters** box, select the byPersonName filter.
e Click the right arrow button to assign the byPersonName filter to the SECURITY_ASSOC table, and then click **OK**.

4 Save the information map.

Any administrators who have physical access to the data can test by logging on to SAS Information Map Studio and running test queries. To verify that the filter is working as expected, log on using different accounts. For example:

- For a user who is not included in the security associations table, no salaries should be retrieved.
- For the president of the company, all salaries should be retrieved. Note that by default only 100 rows of data are returned when you test an information map.
- For a mid-level manager, a subset of salaries should be retrieved.

To run a test query from within SAS Information Map Studio:

1. Select **Tools ▸ Run a Test Query** from the main menu.
2. In the Test the Information Map dialog box, use the arrow button to add the **`Salary`** and **`Employee Name`** items to the **`Selected items`** box.
3. Click **`Run Test`** and then examine the data in the Results dialog box.
4. To test using another account, close the information map, and then select **File ▸ Connection Profile** from the main menu.

Note: Because this is a high-security configuration, final verification must be performed from within SAS Web Report Studio. In a different configuration in which you use server-side pooling, your results in SAS Information Map Studio are the same as your results in SAS Web Report Studio. See "Choices in Workspace Server Pooling" on page 192. △

Variation 1: Use External Identity Values for Filtering

This variation describes one way to work with target data that contains employee IDs (instead of employee names). Modify the main implementation steps as follows:

- Verify that users have the external identity values that you expect. If necessary, you can manually add external identity values to the metadata. See "External Identities" on page 226.
- Use the PARENT_EMPLOYEE_ID column instead of the PARENT_EMPLOYEE_NAME column.
- Use the SAS.ExternalIdentity property instead of the SAS.PersonName property.

Variation 2: Apply Different Filtering Logic to Different Groups

This variation addresses these additional business requirements:

- Four people who work in a Human Resources department must be able to view salary information for all employees. You have created a user-defined group in the metadata repository for these users (the HR group).
- Users who do not have individual metadata identities must not be able to see any of the data. These users have the access that has been defined for the PUBLIC group.

This table summarizes the strategy:

Table 8.3 Information Map Controls

Access Class (User Group)	Information Map
All rows (Human Resources)	Grant Read, ReadMetadata
No rows (PUBLIC)	Deny[1] Read, ReadMetadata
Some rows (SASUSERS)	Grant Read[2], ReadMetadata

1 The information map in this example exists only for the purpose of obtaining salary information, so the "No rows" users do not need to be able to see or use this information map.

2 For each member of SASUSERS, this explicit grant is narrowed by the byPersonName filter that you created in the main example. Here, the filter is used as an authorization-based prefilter.

To set these permissions:

1. Prepare the information map by using either of these methods:

- Create a new information map for this variation by completing steps 1 and 2 in the main example.
- Reuse the information map from the main example by saving that map with a different name and deassigning the filter that was assigned on the **General Prefilters** tab.

2 Open the information map and select **Tools ▶ Authorization** to open the Authorization dialog box.

3 In the **Users and Groups** list, select **PUBLIC**. In the **Effective Permissions** list, add explicit ☑ denials for the Read and ReadMetadata permissions.

4 Click **Add**. In the Add Users and Groups dialog box, select the HR and SASUSERS groups and then click **OK**.

5 In the Authorization dialog box, give SASUSERS explicit ☑ grants of the Read and ReadMetadata permissions.

6 To limit the SASUSERS grant of the Read permission, assign the byPersonName filter to that group.

a Click **Add Condition** to open the Row-Level Permission Condition dialog box.

Note: The **Add Condition** button became available when you added the explicit grant of Read permission. △

b In the **Selected filters** list, select the SECURITY_ASSOC table.

c In the **Available filters** list, select the byPersonName filter and then use the arrow button to move that filter to the **Selected filters** list.

Note: Unlike a filter that you assign on the **General Prefilters** tab, this filter applies only to members of the SASUSERS group as evaluated according to the identity hierarchy and access control precedence rules. △

d Click **OK** to close the Row-Level Permission Condition dialog box.

7 In the Authorization dialog box, give the HR group explicit grants of the Read and ReadMetadata permissions. Because you want this group to be able to view all salaries, do not constrain Read access by adding a permission condition.

8 In the Authorization dialog box, click **Close**. To make your changes take effect, save the information map.

With these access controls in place, retrieval is as follows:

- Users who don't have their own SAS identity (PUBLIC-only users) can't see or use the information map.
- Users who have their own SAS identity but aren't listed in the security associations table can see the information map, but retrieve no rows.
- Users who have their own SAS identity, are listed in the security associations table, and are not members of the HR group get only those rows that contain data for their own direct and indirect reports.
- Users who are members of the HR group get all rows.

BI Row-Level Permissions, Identity-Driven Properties, and Missing Values

If a connecting user doesn't have a value for the identity-driven property that a query uses, the generated query uses an empty string as the substituted value for that identity. If the table against which the query filtering is performed includes empty string values in any rows, those rows are returned to the connecting identity.

CAUTION:

Returning data in a missing value situation is a new behavior in this release. In previous releases, a missing value generated an error and prevented any rows from being returned. △

Here are some alternatives for addressing missing values:

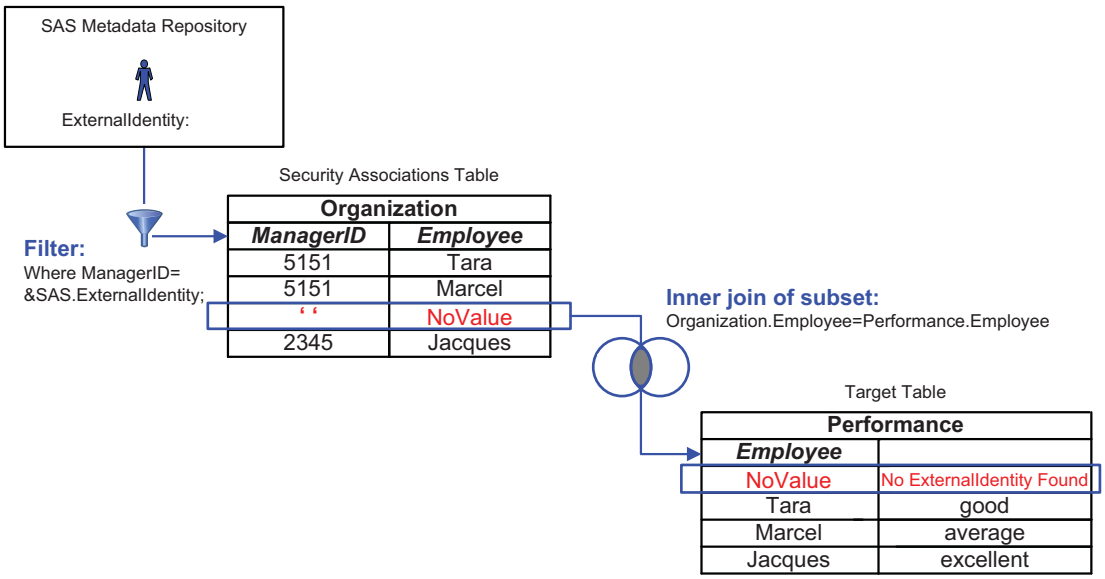
- To ensure that data is returned for only those identities who have a value for the property that you are using, make sure that there aren't any empty string values in the target table's security key column.
- To identify a set of rows that should be returned for identities who don't have a value for the property that you are using, specify an empty string value for those rows in the target table's security key column.

  Note: If the target table is in a DBMS, the extra row must contain an empty (blank) character string (not a NULL value). △
- To identify a situation in which retrieval is empty due to a missing value for the requesting user's identity-driven property, include a mapping for the empty string value (' ')in your security associations table and one extra row in your target table. In that row, use the security key that corresponds to the empty string value and include an appropriate error message. This enables the end user to distinguish between the following situations:
  - an empty result set that is caused by the target table not including any rows that match the user's value for an identity-driven property
  - an empty result set that is caused by the user not having any value for the identity-driven property that a query is using

  Note: If the security associations table is in a DBMS, make sure that the missing value row in that table contains an empty (blank) character string, not a NULL value. △

  The following figure depicts an example:

Figure 8.9 Example: Error Handling for a Missing External Identity Value



CHAPTER

9

OLAP Member-Level Permissions

About OLAP Member-Level Permissions

OLAP member-level permissions enable you to limit access to SAS OLAP data by using filters. Each filter consists of an MDX expression that subsets the data in a dimension as appropriate for a particular user or group. The filters are stored in the metadata as permission conditions. This feature relies on the SAS OLAP Server for enforcement. At query time, the server performs the filtering to determine which dimension members should be returned to each requesting user. The server leverages its knowledge to accomplish the filtering efficiently. This also ensures that the filters are applied every time the data is accessed.

When you use OLAP member-level permissions, it is essential to understand these points:

- With OLAP data, permission conditions can be specified only on dimension objects.
- The SECURITY_SUBSET option can affect results. See the discussion of this option in the topic "Cube Security" in the *SAS OLAP Server: User's Guide*.
- The members that are returned by the MDX expression must all belong to the dimension on which the permission condition is defined. The returned set of members cannot be a union of members from other dimensions.
- A permission condition that filters a non-default hierarchy must include at least one member of the default hierarchy. If a requesting user does not have access to any members in the default hierarchy, then the query fails with a permissions error.

See Also

"Fine-Grained Controls for Data" on page 65

SAS OLAP Server: MDX Guide

SAS OLAP Server: User's Guide

How to Assign an OLAP Permission Condition

1 Access the **Authorization** tab of the dimension  for which you are defining a permission condition.

Note: In SAS Management Console, you can access dimensions on the **Plug-ins** tab under **Authorization Manager ▶ Resource Management ▶ By Location ▶ <your application server> ▶ <your OLAP schema> ▶ <your cube> ▶ Dimensions**. In SAS OLAP Cube Studio, navigate to the dimension on the **Inventory** tab. △

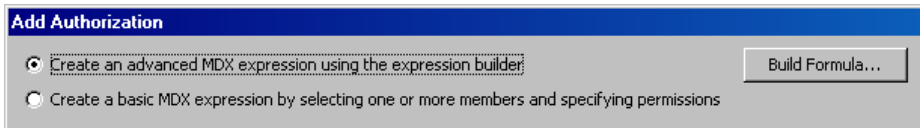
2 Select (or add) the user or group whose access to measures you want to limit.

3 In the permissions list, add an explicit ☑ grant of the Read permission for that user or group.

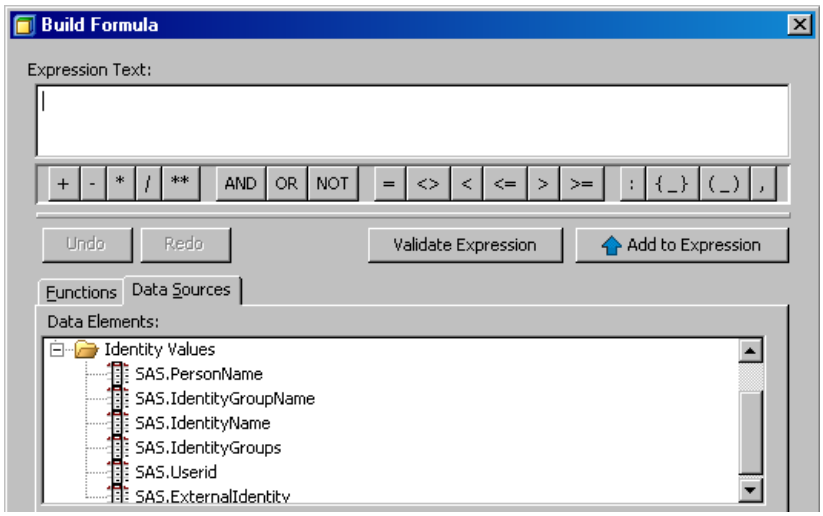
4 Click the **Add Authorization** button.

Note: If the **Edit Authorization** button is displayed, a condition already exists for the selected user or group. △

5 In the Add Authorization dialog box, select the **Create an advanced MDX expression** radio button and click **Build Formula**.



6 Use the Build Formula dialog box to create a filter that limits the current dimension as appropriate for the user or group that you selected on the **Authorization** tab. Identity-driven properties are available on the **Data Sources** tab under the **Identity Values** folder. See "Identity–Driven Properties" on page 66.



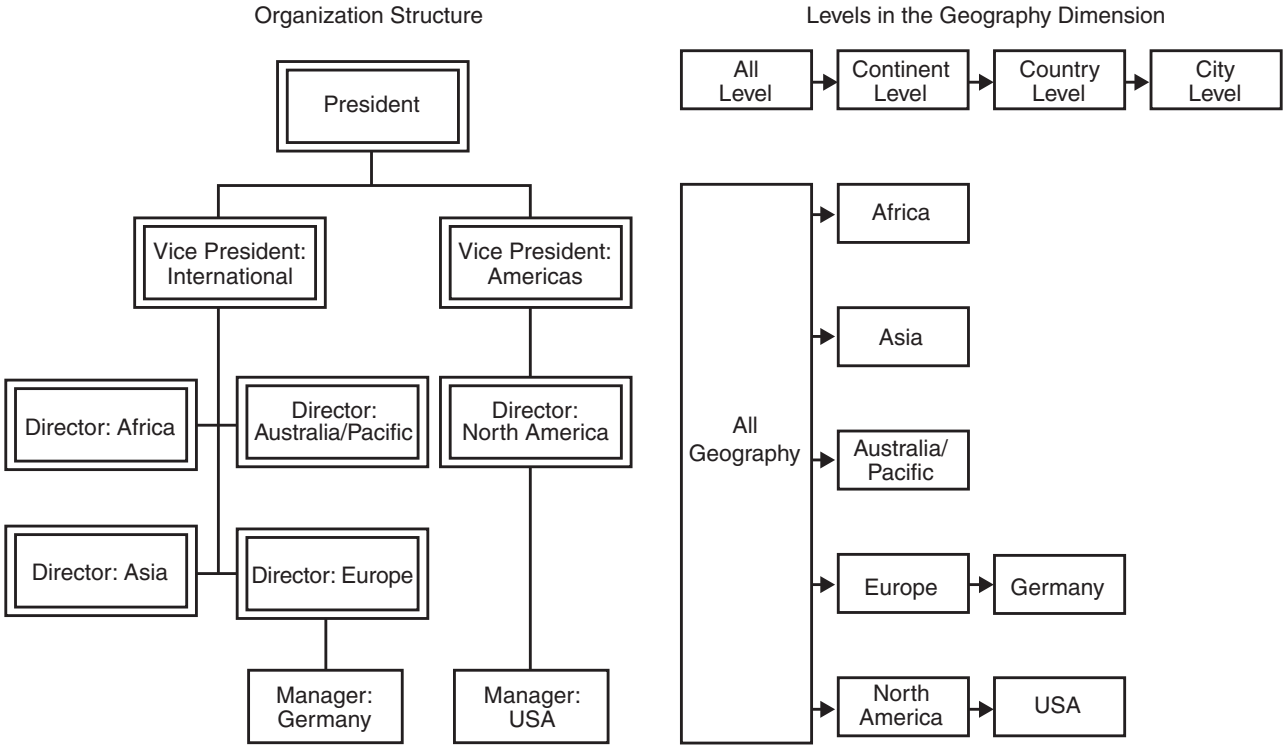
Example: Using Member-Level Permissions

Introduction

This example demonstrates how you can use member-level permissions. These are the security goals in this example:

- enable company executives to access data based on their areas of responsibility
- prevent other employees from accessing data in the cube that is used to generate executive reports

The following figure depicts the relevant parts of a company's organization structure and of the OLAP cube against which the company runs executive reports. Notice that the levels in the cube's Geography dimension closely correspond to the depicted organizational structure.



Implementation Process

1 Set these permissions on the cube:
   - Give the PUBLIC group explicit ☑ denials of the Read and ReadMetadata permissions for the entire cube. The data in the cube is used for executive reports only.
   - Give the company president explicit ☑ grants of the Read and ReadMetadata permissions for the entire cube. This user sees all of the data.

- Give the SAS System Services group an explicit ☑ grant of the ReadMetadata permission. You must always preserve the SAS Trusted User's access to cubes and schemas.
- In most cases, some members of the information technology staff will also need access to the data for administrative purposes. Make sure that any such groups or users have explicit ☑ grants of the Read and ReadMetadata permissions.

2 To assign the permission conditions:

a Right-click the cube's **Geography** dimension  .

Note: In SAS Management Console, you can access the dimensions on the **Plug-ins** tab under **Authorization Manager ▶ Resource Management ▶ By Location ▶ <your application server> ▶ <your OLAP schema> ▶ <your cube> ▶ Dimensions**. In SAS OLAP Cube Studio, navigate to the dimension on the **Inventory** tab. △

b On the **Authorization** tab, select (or add) the Vice President Americas user.
c In the permissions list, add an explicit ☑ grant of the Read permission for the Vice President Americas user. This enables the **Add Authorization** button.
d Click the **Add Authorization** button,
e In the Add Authorization dialog box, select the **Create an advanced MDX expression** radio button and click **Build Formula**.
f In the Build Formula dialog box, paste or build a filter for the Vice President Americas user and click **OK**.
g On the **Authorization** tab, click **OK** to save this permission condition and then return to step 2b to repeat the process for the next executive.

For example, this table contains MDX expressions that you could use to subset the data based on each executive's area of responsibility.

Table 9.1 Example: MDX Expressions

User	MDX Expression	Notes
Vice President Americas	{[Geography].[All Geography], Descendants([Geography].[All Geography].[North America],[Geography].[Continent],SELF_AND_AFTER)}	
Vice President International	Except({[Geography].Members}, {Descendants([Geography].[All Geography].[North America],[Geography].[Continent],SELF_AND_AFTER)})	Use Except to exclude North America.
Director North America	{[Geography].[All Geography],[Geography].[All Geography].[North America],[Geography].[All Geography].[North America].Children}	Use .Children to include countries.
	{[Geography].[All Geography],[Geography].[All Geography].[North America],descendants([Geography].[All Geography].[North America])}	Alternative: use Descendants to include countries and cities.
	{[Geography].[All Geography].[North America], [Geography].[All Geography].[North America].Children}	Alternative: exclude the All level.

User	MDX Expression	Notes
Director Africa	{[Geography].[All Geography].[Africa], [Geography].[All Geography].[Africa].Children}	
Director Asia	{[Geography].[All Geography].[Asia],[Geography].[All Geography].[Asia].Children}	
Director Australia/ Pacific	{[Geography].[All Geography].[Australia/Pacific], [Geography].[All Geography]. [Australia/Pacific].Children}	
Director Europe	{[Geography].[All Geography].[Europe], [Geography].[All Geography].[Europe].Children}	
Manager Germany	{[Geography].[All Geography].[Europe].[Germany], descendants([Geography].[All Geography].[Europe].[Germany])}	Exclude Europe.*
Manager USA	{[Geography].[All Geography].[North America].[USA], descendants([Geography].[All Geography].[North America].[USA])}	Exclude North America.*

* Because this expression excludes a parent level, you should also deny this user the ReadMetadata permission for the level that you are hiding. This is a requirement when OLAP data is accessed through an information map.

CHAPTER
10

Security Report Macros

Overview of Security Reporting

Security reporting creates a snapshot of metadata layer access control settings and uses that snapshot in these ways:

- as a data source for reports about current settings
- as a data point for comparing settings across time

The first task in security reporting is to extract, filter, and format authorization data for a specified set of identities, permissions, and objects. SAS provides macros that help you perform this task. For example, the following code uses the main security report macro, %MDSECDS, to generate authorization data sets for a specified folder and its contents:

```
/* connect to the metadata server */
options metaserver=machine-name metauser="sasadm@saspw" metapass="Pwd123";

/* run the main report macro against a target folder */
%mdsecds(folder="\Shared Data\New Stored Processes");
run;
```

The second task in security reporting is to build reports that run against the authorization data sets. For example, the following code prints some of the data from the preceding example:

```
/* print the WriteMetadata setting information  */
proc print data=work.mdsecds_join noobs width=minimum;
var objname publictype identityname WriteMetadata;
run;
```

The output looks like this:

ObjName	PublicType	identityname	WriteMetadata
New Stored Processes	Folder	PUBLIC	Denied Indirectly
New Stored Processes	Folder	SAS System Services	Denied Indirectly
New Stored Processes	Folder	SAS Administrators	Granted by ACT
Shoe Sales Graphics	StoredProcess	PUBLIC	Denied Explicitly
Shoe Sales Graphics	StoredProcess	SAS System Services	Denied Indirectly
Shoe Sales Graphics	StoredProcess	SAS Administrators	Granted Explicitly
Shoe Sales by Region	StoredProcess	PUBLIC	Denied Indirectly
Shoe Sales by Region	StoredProcess	SAS System Services	Granted Indirectly
Shoe Sales by Region	StoredProcess	SAS Administrators	Granted Indirectly
Shoe Sales by Region	StoredProcess	SASUSERS	Granted Explicitly

Because the output is SAS data sets, you can create sophisticated reports by using SAS reporting techniques such as these:

- create an information map that uses an authorization data set as its data source and then use SAS Web Report Studio to create reports based on that information map.
- write SAS ODS (output display system) code that builds an HTML report against the authorization data sets. For an example, see *SAS-installation-directory*`\SASFoundation\9.2\core\sample\secrpt.sas` (Windows) or *SAS-installation-directory*`/SASFoundation/9.2/samples/base/secrpt.sas` (UNIX).

%MDSECDS

Generates authorization data sets. This is the top-level macro (it calls the underlying macros and should be used on its own).

Used in: Security reporting

Type: Stand-alone

Requirement: Connection to the metadata server

Syntax

%MDSECDS (OUTDATA=*data-set*, <FOLDER="*path*">,
<INCLUDESUBFOLDERS=YES | NO>, <MEMBERTYPES="*list*">,
<MEMBERFILTER="*expression*">, <PERMS="*list*">, <IDENTITYNAMES="*list*">,
<IDENTITYTYPES="*list*">);

OUTDATA
provides a base name for the output. By default, the base name is `work.mdsecds`. See “Authorization Data Sets” on page 118.

The following options define the scope of the extraction:

FOLDER
identifies a starting point folder. By default, the starting point is the server root (the first node on the `Folders` tab in SAS Management Console). If you provide a path (such as `"\Products\SAS Intelligence Platform\Samples"`), the starting point is the last folder in the path.

Note: To avoid having to type a long pathname, copy the path from a child item's `General` tab into your code. △

INCLUDESUBFOLDERS
controls whether the entire subtree is included. By default, all objects in the entire subtree are included. If you specify **NO**, only the immediate contents of the starting point folder are included.

MEMBERTYPES
limits by object type. By default, all public types are included. If you provide a comma-delimited list of types, only those types are included.

You must provide the public type name in its TypeName format. For example, if you access the **Advanced** tab in the properties dialog box for the **Information Map (relational)** object type, you will see that its TypeName is **InformationMap.Relational**. All public types are displayed on the **Folders** tab in SAS Management Console under **System** ▶ **Types**.

Note: If you use this option, examine the output. The log doesn't display errors or warnings for incorrectly specified types. △

MEMBERFILTER
limits by metadata attribute value. By default, no filter is applied. If you provide an expression, only objects that meet the criteria are included. The format for the expression is *@attribute-name comparison-operator 'value'* (for example, **@objID ='A5HDAJSI.B90006Y5'** or **@Name =:'Salary'**).

Comparison operators for character data include **=** (equals), **=:** (begins with), **?** (contains), and **ne** (not equals).

Metadata attributes are associated with an object's metadata type (not public type). Here are two common attributes:

objID — is the object's metadata ID, which is displayed on the object's **Inheritance** tab.

Note: To access an object's **Inheritance** tab, open the object's properties dialog box, select the **Authorization** tab, and then click **Advanced**. This feature is available only to unrestricted users. △

Name — is the object's name, which is displayed in the **Name** field on the object's **General** tab.

To find additional attributes:

1. Determine the MetadataType of the object you are interested in. The type is displayed on the object's **Inheritance** tab. For example, if you look at the **Inheritance** tab for your My Folder  , you will see that the MetadataType for a folder is **Tree**. MetadataType is a low level, internal classification.
2. In the reference documentation for the metadata model, look up that MetadataType to find the names of its attributes.

PERMS
specifies which permissions to include. By default, all permissions that are supported for each object type are included for objects of that type. If you provide a comma-delimited list of permissions, only those permissions are included. Even if the permissions in your user interface are translated, you must specify the English-language permission names (for example, ReadMetadata).

IDENTITYNAMES
specifies which identities to include. By default, only the named participants (the identities that are listed on an object's **Authorization** tab) are included. If you provide a comma-delimited list of identity names, only those identities are included. List identities by their names, not their display names.

If you use this option, you must also use the IDENTITYTYPES option.

IDENTITYTYPES
specifies whether names in the IDENTITYNAMES list correspond to users or to groups. For example, the first name listed in the IDENTITYNAMES parameter must match the first value in the IDENTITYTYPES list. Valid values in this list are **`Person`** and **`IdentityGroup`**.

Examples

This code extracts information about permissions on the objects in the Sales folder but doesn't include objects in subfolders:

```
%mdsecds(folder="\Shared Data\Sales", includesubfolders=no);
```

This code extracts information about permissions on tables and schemas in the Sales folder and its subfolders:

```
%mdsecds(folder="\Shared Data\Sales", membertypes="Library,OLAPSchema");
```

This code extracts information that indicates which libraries a particular user (the SAS Demo User) can see:

```
%mdsecds(identitynames="sasdemo", identitytypes="Person", membertypes="Library",
      perms="ReadMetadata");
```

This code extracts information that indicates which stored processes two users (the SAS Demo User and Tara O'Toole) can see:

```
%mdsecds(identitynames="sasdemo,totoo", identitytypes="Person,Person",
     membertypes="StoredProcess", perms="ReadMetadata");
```

This code extracts information that indicates which reports one user and one group (the SAS Demo User and PUBLIC) can modify:

```
%mdsecds(identitynames="sasdemo,PUBLIC", identitytypes="Person,IdentityGroup",
     membertypes="Report", perms="WriteMetadata");
```

This code extracts information that indicates who can view reports that include the word "Salary" in their names:

```
%mdsecds(membertypes="Report", perms="ReadMetadata", memberfilter="@Name ? 'Salary'");
```

This code extracts permission settings for an item that is referenced by its object ID:

```
%mdsecds(memberfilter="@objID  = 'A5HDAJSI.B90006Y5'");
```

Authorization Data Sets

Table 10.1 work.mdsecds_join (the Primary Table for Reporting; Combines objs, permsw, pconds)

Column Name	Column Contents
IdentityDispName	The display name of this user or group (for example, **`SAS Demo User`**). If there is no display name, this column contains the same value as the IdentityName column.
IdentityName	The name of this user or group (for example, **`sasdemo`**).

Column Name	Column Contents
IdentityType	**Person**, **IdentityGroup**, or **Role**.
Location	A container path for this object (for example, **\Shared Data\Sales**). The path usually consists of folder names but can also include names of other containers.
MetadataCreated	The date when this object was created.
MetadataType	The internal metadata type for this object.
MetadataUpdated	The most recent date that this object was updated.
ObjId	The metadata ID for this object (for example, **A5HDAJSI.B900066J**).
ObjName	The name of this object (for example, **Shoe Sales by Region)**.
ObjUri	The uniform resource identifier for this object (for example, **omsobj:PhysicalTable/ A5XT9KUX.B80000001**).
ParentObjId	The metadata ID for this object's immediate parent. See "Inheritance Paths" on page 53.
Permissions	A list of the permissions to inspect for this object. If the list is blank, then all permissions that are applicable to this type of object are inspected.
(permissions)	A separate column for each permission. Each cell in these columns contains a value that indicates whether this permission is effectively granted or denied for this identity. The type of setting (explicit, ACT, or indirect) is also indicated. For example, **Granted by ACT** or **Denied Explicitly** or **Granted Indirectly**. See Table 4.1 on page 41.
PublicType	The public metadata type for this object.

Table 10.2 work.mdsecds_objs (Contains Folder and Member Information)

Column Name	Column Contents
Desc	Description information for this object. For most objects, this information comes from the **Description** field on the **General** tab in the object's properties dialog box.
Location	A container path for this object. The path usually consists of folder names but can also include names of other containers.
MetadataCreated	The date when this object was created.
MetadataType	The internal metadata type for this object.
MetadataUpdated	The most recent date that this object was updated.
ObjId	The metadata ID for this object.
ObjName	The name of this object.
ObjUri	The uniform resource identifier for this object.
ParentObjId	The metadata ID for this object's immediate parent.
Permissions	A list of the permissions to inspect for this object. If the list is blank, then all permissions that are applicable to this type of object are inspected.
PublicType	The public metadata type for this object.

Table 10.3 work.mdsecds _pconds (Contains Permission Condition Expressions)

Column Name	Column Contents
Condition	The expression that limits this grant.
IdentityName	The name of this user or group.

Column Name	Column Contents
IdentityType	**Person**, **IdentityGroup**, or **Role**.
ObjUri	The uniform resource identifier for this object.
Permission	The name of this permission (for example, **Read**).

Table 10.4 work.mdsecds_permsw (Contains Permissions Data Transformed to Wide Format)

Column Name	Column Contents
IdentityDispName	The display name of this user or group (or the name if there is no display name).
IdentityName	The name of this user or group.
IdentityType	**Person**, **IdentityGroup**, or **Role**.
ObjName	The name of this object.
ObjUri	The uniform resource identifier for this object.
(permissions)	A separate column for each permission. Each cell in these columns contains a value that indicates whether this permission is effectively granted or denied for this identity. The type of setting (explicit, ACT, or indirect) is also indicated.

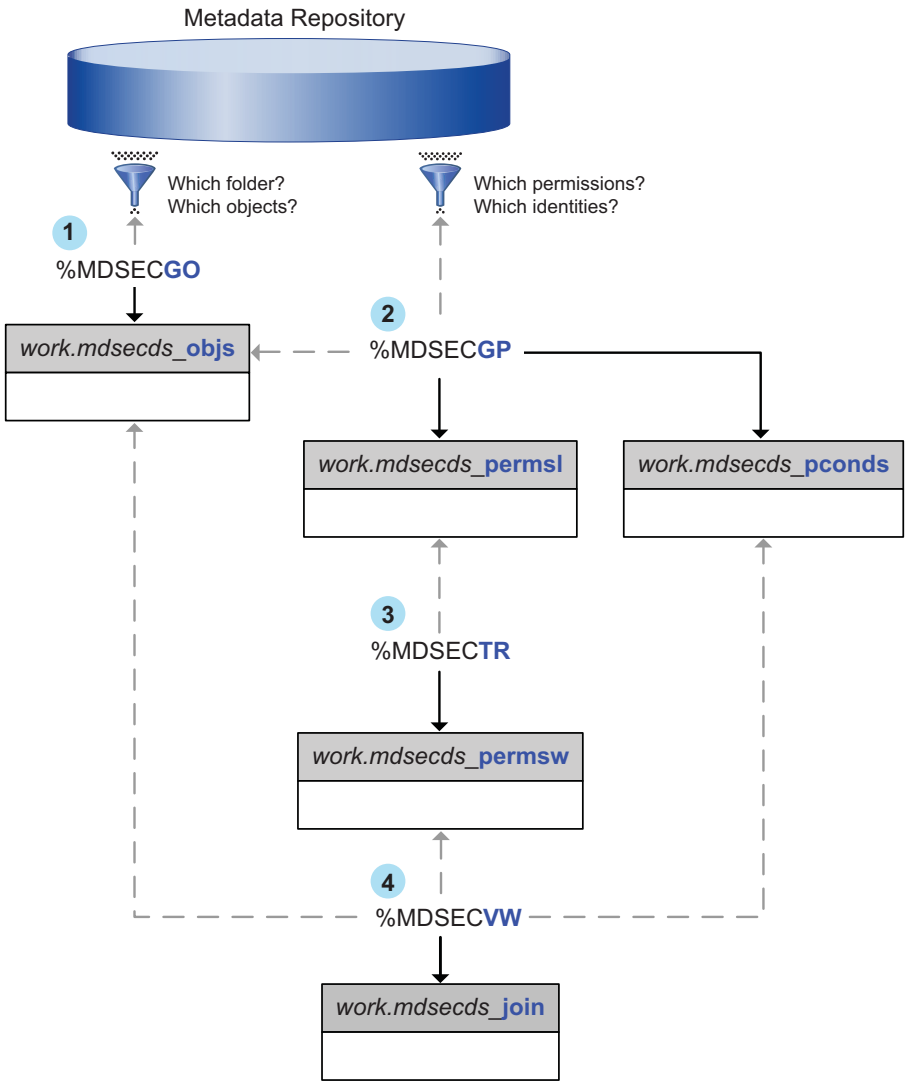
Table 10.5 work.mdsecds _permsl (Contains Permissions Data in Original Long Format)

Column Name	Column Contents
Authorization	A value that indicates where this permission is granted or denied for this identity. The type of setting (explicit, ACT, or indirect) is also indicated.
Conditionflg	A value that indicates whether this permission setting is conditional. The value is either blank (no condition) or * (condition applies and is written to the pconds table).
IdentityDispName	The display name of this user or group (or the name if there is no display name).
IdentityName	The name of this user or group.
IdentityType	**Person**, **IdentityGroup**, or **Role**.
ObjName	The name of this object.
ObjUri	The uniform resource identifier for this object.
Permission	The name of this permission.

Additional Resources for Building Authorization Data Sets

You can choose to directly use the underlying report macros instead of using only %MDSECDS (which is the standard approach to building authorization data sets). However, the underlying macros don't offer any unique inclusion or exclusion parameters. The following figure introduces the underlying macros. In the figure, arrow direction indicates input to and output from each underlying macro.

Figure 10.1 Underlying Security Report Macros



The numbers in the preceding figure correspond to these activities:

1 %MDSECGO extracts information for a specified set of objects. You specify one folder and indicate whether to include subdirectories. You can also provide a list of object types to include and filter the data set by attribute value.

   Note: The same level of control is provided by using %MDSECDS on its own. △

2 For every object in a specified data set, %MDSECGP gets effective permission settings for a specified set of identities and permissions.

   Note: This is the point at which using the underlying macros creates an opportunity for you to define a subset of the objs data set. △

3 %MDSECTR transforms the extracted data set from a long format (a separate row for each permission) to a wide format (all permissions in the same row).

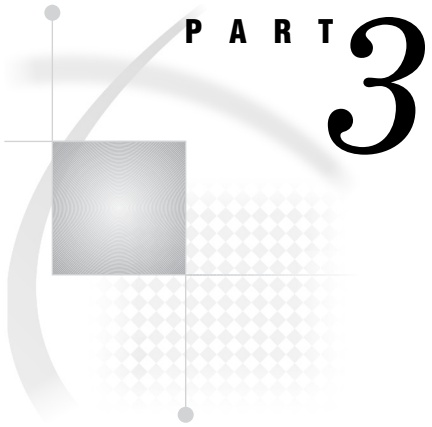
4 %MDSECVW creates a joined view or data set that can be used for reporting.

The following table introduces a few utility macros that can be useful in security reporting:

Table 10.6 Utility Macros for Security Reporting

Utility Macro	Description
%MDUTYPED	Extracts information about top-level metadata objects or locates templates for a particular object type.
%MDUGFLDR	Returns the object ID for a specified folder.
%DEFINEOBJTAB_SQL	Defines the table into which %MDSECGO inserts rows.

The underlying macros and utility macros are in *SAS-installation-directory*`\SASFoundation\9.2\core\sasmacro` (Windows) or *SAS-installation-directory*`/SASFoundation/9.2/sasautos` (UNIX).



PART 3

Authentication

CHAPTER

11

Authentication Model

Introduction to the Authentication Model

There is no single mechanism that is applicable for all authentication events throughout a typical deployment. Instead, each deployment uses some combination of authentication processes, trust relationships, and single sign-on technologies. This helps to balance a range of goals such as these:

- preserve individual identity
- minimize security exposures
- provide a unified user experience
- minimize set up and maintenance efforts
- provide access to disparate systems within an environment
- integrate into a wide variety of general computing environments

This chapter provides an in-depth understanding of authentication and security aspects of SAS server configuration. For basic information about authentication, see “Facilitate Authentication” on page 8.

Authentication to the Metadata Server

How SAS Identity is Determined

When a user launches a SAS client, the following process occurs:

verification phase
ensures that the user is who he or she claims to be. For example, this credential-based host authentication method might be used:

1. The client prompts the user for an ID and password.
2. The user enters credentials that are known to the metadata server's host.
3. The client sends the credentials to the metadata server.
4. The metadata server passes the credentials to its host for authentication.
5. If the host determines that the user has a valid account, the host returns the authenticated user ID to the metadata server.

SAS identity phase
resolves the authenticated user ID to a particular SAS identity. In this phase, SAS examines its copies of user IDs in an attempt to find one that matches the authenticated user ID. One of the following outcomes occurs:

- A matching user ID is found, so a connection is established under the owning identity. The owning identity is the user or group whose definition includes a login with the matching user ID.

  Note: The metadata server's integrity constraints ensure that there won't be more than one owning identity. See "Unique Names and IDs" on page 33. △

- No matching user ID is found, so a connection is established under the generic PUBLIC identity. In the metadata layer, the user is a PUBLIC-only user.

  Note: The matching process expects the SAS copy of the user ID to be qualified (if it is a Windows user ID). See "User ID Formats" on page 32. △

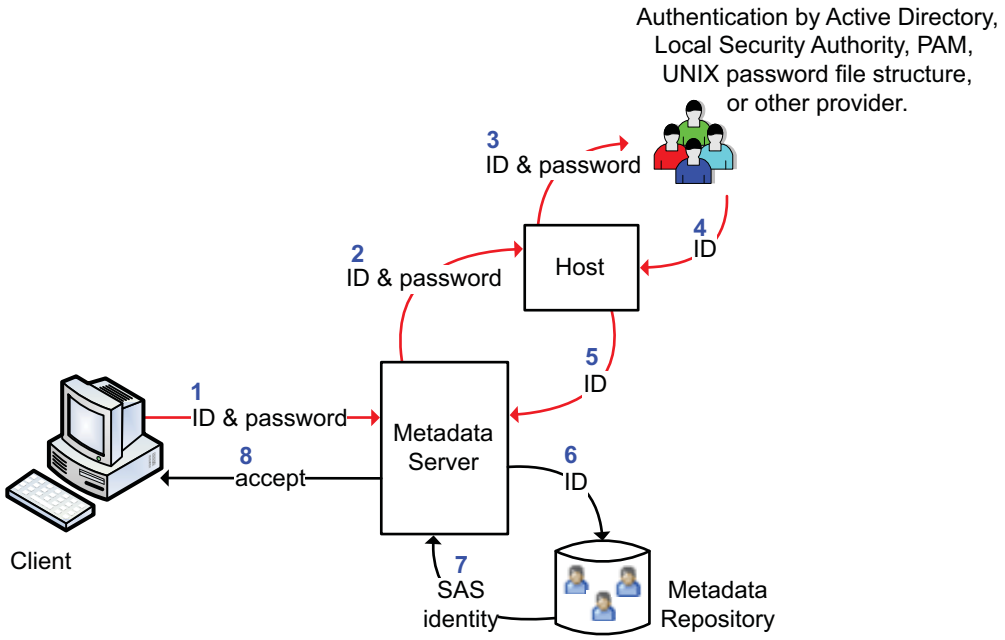
Depictions of the Authentication Process

The following figures introduce authentication methods for the metadata server. For details about any of the following configurations, see Chapter 12, "Authentication Mechanisms," on page 143. In the following figures, notice these points:

- Only the verification phase varies; the SAS identity phase is always the same. With any approach, you need a well-formed user definition for each user who isn't a PUBLIC-only identity.
- Except where internal accounts are used, the process always involves two sets of identity information, one in an external provider and another in the metadata.

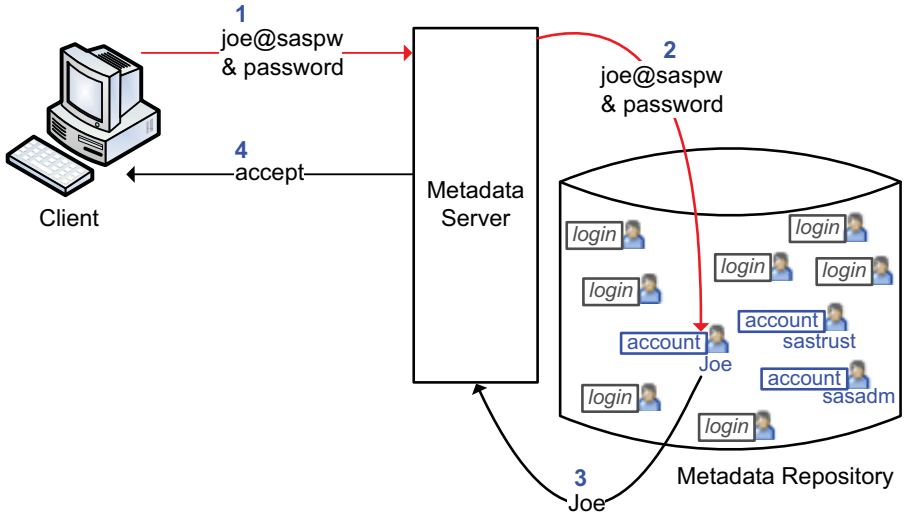
The following figure depicts the basic process.

Figure 11.1 Metadata Server: Host Authentication (Credential-Based)



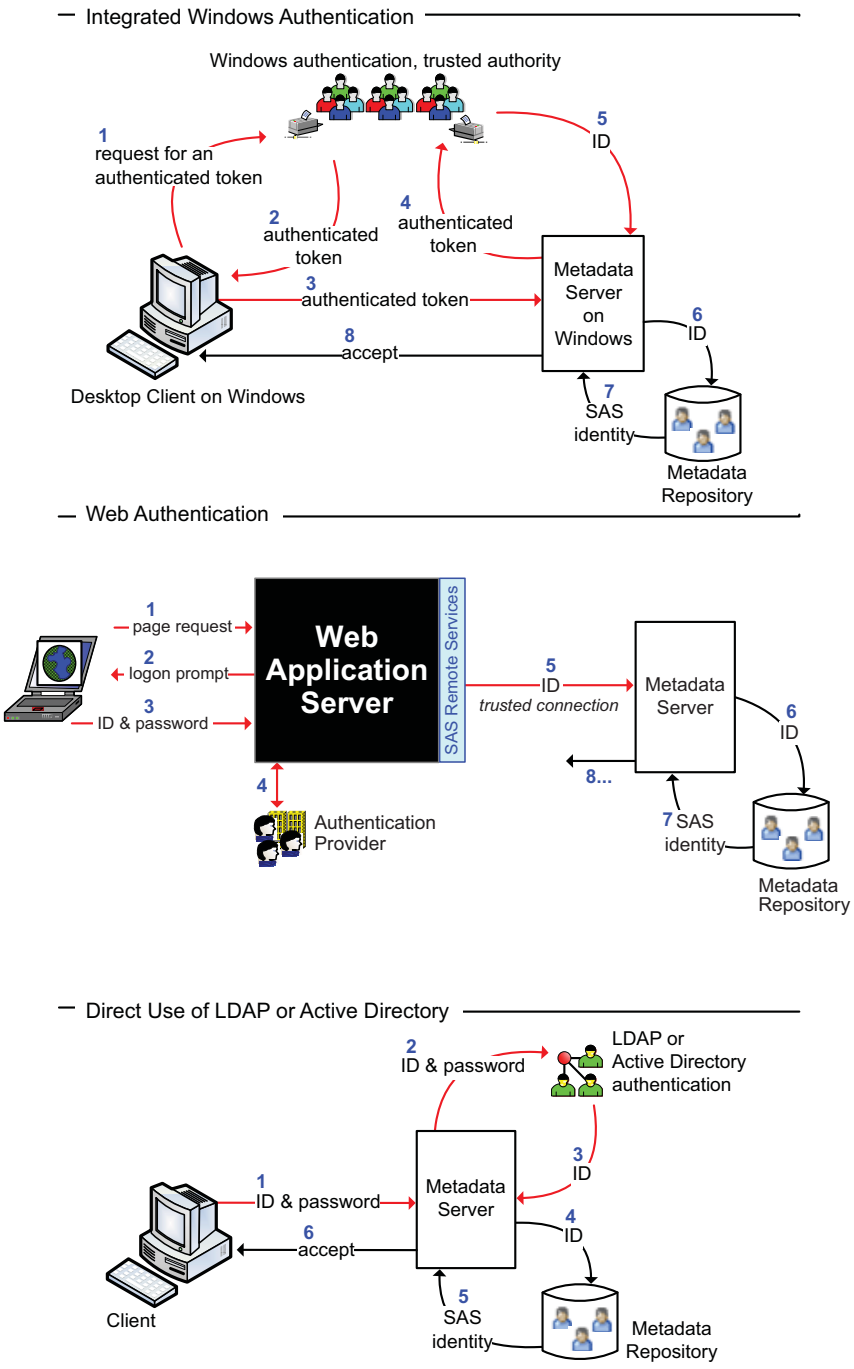
The following figure depicts a special case where a metadata administrator named Joe uses an internal account.

Figure 11.2 Metadata Server: Internal Authentication



The following figure introduces alternate approaches that can help you use accounts that already exist in your environment or provide single sign-on (silent launch of clients) These approaches are documented in Chapter 12, "Authentication Mechanisms," on page 143.

Figure 11.3 Alternate Configurations



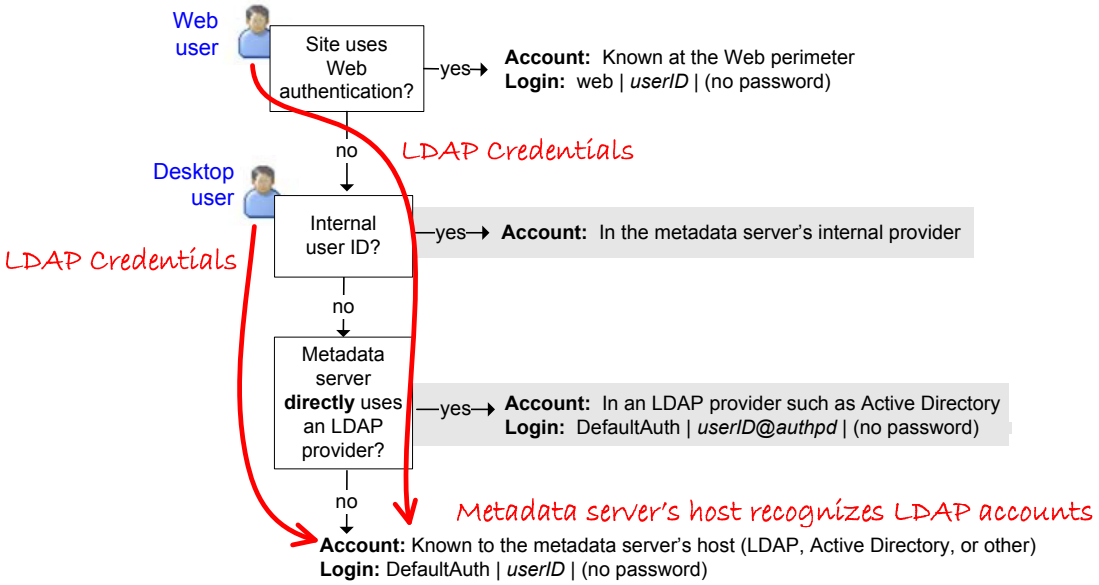
Note: This topic is about authentication of users. For information about how some servers connect to the metadata server, see "Trusted Peer Connections" on page 154 and "Trusted User Connections" on page 155. △

Example: Metadata Server on UNIX

The following annotated template shows how a UNIX site might provide access to the metadata server. Here are some points about this example:

- The UNIX computer that hosts the metadata server recognizes LDAP accounts.
- This site can't use Integrated Windows authentication (IWA) because the SAS implementation of IWA is available only if the metadata server is on Windows. When users at this site launch SAS desktop clients, they provide the user ID and password for their LDAP accounts.
- This site chose to not use Web authentication. When users at this site access a SAS Web application, they provide the user ID and password for their LDAP accounts. Access from one SAS Web application to another is seamless.

Figure 11.4 Example: Metadata Server on UNIX



Note: In the template, the internal account branch is shaded because internal accounts are intended only for metadata administrators and some service identities. The direct LDAP branch is shaded because direct use of an LDAP provider (such as Active Directory) is not a first choice mechanism. △

Example: Metadata Server on Windows

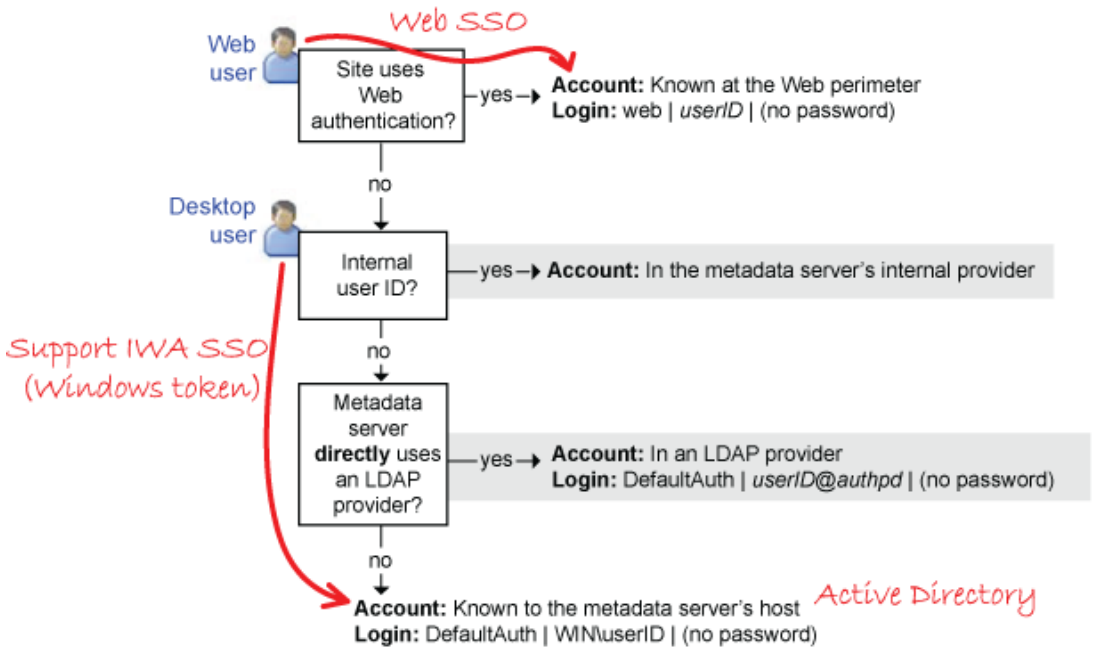
The following annotated template shows how a Windows site might provide access to the metadata server. Here are some points about this example:

- Integrated Windows authentication (IWA) provides silent launch from SAS desktop clients for each user who chooses to use this feature. Users who don't select the IWA setting in their connection profiles are instead authenticated using credential-based authentication.
- Web applications can't use the SAS implementation of IWA. To provide silent launch for these applications, this site configured Web authentication. Everyone who has authenticated at the site's Web perimeter accesses SAS Web applications without interactively providing credentials. Unlike IWA, Web authentication doesn't require (or allow) individual users to opt in or out.
- Notice that the user's login contains a qualified user ID. The user ID in any Windows login must be properly qualified. Someone who has a missing or

improperly specified user ID in his or her login has no individual SAS identity and is a PUBLIC-only identity.

- The figure suggests that someone who uses both Web and desktop applications needs two logins, one in DefaultAuth and one in the web authentication domain. If the same user IDs are valid in both contexts, having a Web login is not a strict requirement (unless the Web applications must provide seamless access to a standard, unpooled workspace server).

Figure 11.5 Example: Metadata Server on Windows



Note: In the template, the internal account branch is shaded because internal accounts are intended only for metadata administrators and some service identities. The direct LDAP branch is shaded because direct use of an LDAP provider (such as Active Directory) is not a first choice mechanism. △

Authentication to Data Servers and Processing Servers

A registered user who has a connection to the metadata server accesses the OLAP server, stored process server, and pooled workspace server seamlessly by SAS token authentication. Authentication to following servers requires coordination:

standard workspace server
: to provide seamless access, see "Coordinate the Workspace Server" on page 8.

third-party data server (for example, Oracle)
: to provide access, select an approach from the following table. For instructions, see "How to Store Passwords for a Third-Party Server" on page 174.

Table 11.1 Authentication to a Third-Party Data Server

Goal	Approach
Provide seamless access and preserve individual identity to the target server.	Store individual user IDs and passwords in the metadata (each on the **Accounts** tab of a different user definition).
Provide seamless access.	Store the user ID and password for one shared account in the metadata (on the **Accounts** tab of a group definition).
Provide seamless access with a few distinct access levels for resources in the target server.	Store a few shared user IDs and passwords in the metadata (each on the **Accounts** tab of a different group definition).[1]
Preserve individual identity.	No configuration required. Users will be prompted for credentials for the target server.[2]

1 For a hybrid approach, use a combination of personal and group logins.
2 Secondary prompting is supported for desktop applications and SAS Web Report Studio.

Note: In general, requesting users can access only those servers for which they have the ReadMetadata permission. An exception is client-side pooling, in which access depends on membership in a puddle group. △

Note: This topic is about metadata-aware connections. Direct connections to a SAS server (for example, from the SAS Add-In for Microsoft Office to the OLAP server) can't use SAS token authentication. Direct connections use client-supplied credentials or, in some cases, Integrated Windows authentication. △

See Also

"SAS Token Authentication" on page 153

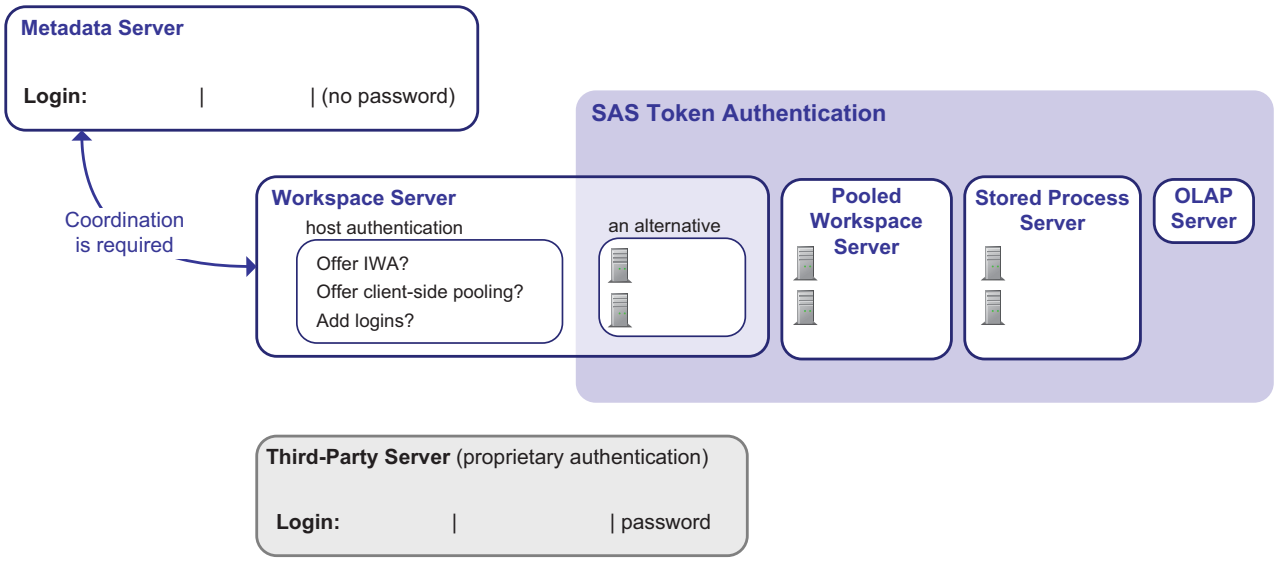
"Choices in Workspace Server Pooling" on page 192

Authentication Scenarios

Introduction and Template

The examples in this section show how different sites might annotate the following template to reflect their environment and choices. Not all deployments use all servers.

Figure 11.6 Template for Authentication Scenarios



Here are some details about the template:

- To facilitate integration with your environment, SAS provides several authentication choices for the metadata server. See "Authentication to the Metadata Server" on page 126.

  Note: Most users should have their own inbound login for the metadata server. Exceptions are users who don't need an individual identity and users who use a SAS internal account. △

  Note: The metadata server also accepts SAS token authentication connections from processing servers. △

- The workspace server usually uses some form host authentication. If you have mixed providers, you might configure the workspace server to use SAS token authentication (instead of using host authentication). See "Coordinate the Workspace Server" on page 8.

  Note: When a user requests a standard workspace server, it is actually the spawner that performs the authentication (in accordance with the workspace server's metadata settings) and then launches the server process. That detail is omitted in the general discussion. △

- Most other SAS servers use SAS token authentication for all metadata-aware requests. See "SAS Token Authentication" on page 153.

- Some types of logical server can contain multiple servers  with each server running under a different launch credential. See "Host Access to SAS Tables" on page 187.

- The logical pooled workspace server  provides server-side pooling. This is not the same as client-side pooling, which can be offered only by a specially configured standard workspace server. See "Choices in Workspace Server Pooling" on page 192.

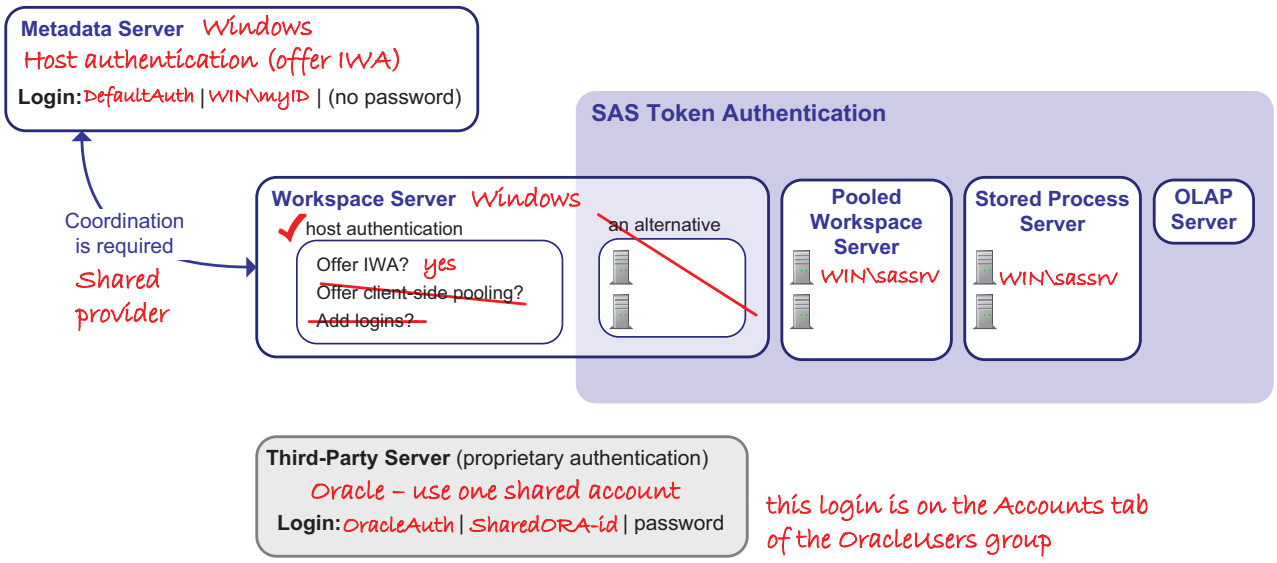
- The OLAP server doesn't have credentials as part of its metadata definition because that server isn't launched by the object spawner.

- In order to provide seamless access to a third-party server that uses its own proprietary user registry, you must store individual or group credentials in the metadata. See Table 11.1 on page 131.

Example: Windows and Shared Access to Oracle

The following annotated template shows how a Windows site might provide access to servers. This example also illustrates one way to incorporate a third-party server (an Oracle server that maintains its own user registry and doesn't accept Windows credentials).

Figure 11.7 Authentication Scenarios: Windows Example



Here are some points about this example:

- This site can offer Integrated Windows authentication (IWA) for the metadata server because that server is on Windows.
- Because the metadata server offers IWA and the workspace server is also on Windows, this site configured the workspace server to also offer IWA. This is necessary in this scenario because when the initial logon is by IWA, there are no cached credentials that a client could reuse for credential-based host authentication to the workspace server.
- If a user doesn't opt in to IWA in their connection profile, the user provides credentials in the initial logon dialog box. Those credentials are added to the user's in-memory credential cache (user context). When the user requests a workspace server, those credentials are reused even though the workspace server is configured for IWA.
- This site chose to provide seamless shared access to Oracle by storing an Oracle user ID and password on the **`Accounts`** tab of a custom group, OracleUsers.
  - If this site also stores individual Oracle credentials for selected members of the OracleUsers group, those members would authenticate with their individual credentials. A personal OracleAuth login would have priority over the group OracleAuth login.

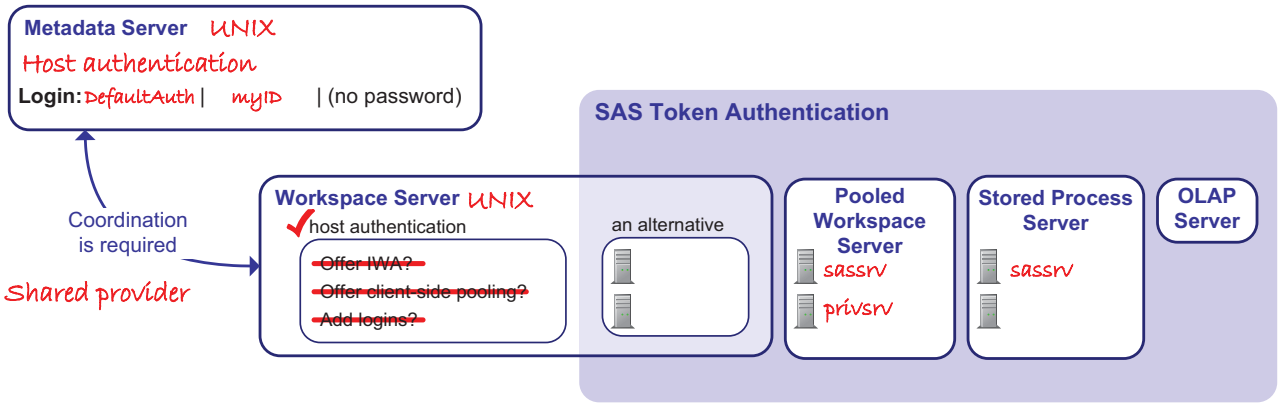
- If this site doesn't store any Oracle passwords, interactive (prompted) access to Oracle is available from SAS desktop applications and in SAS Web Report Studio.

- This site hasn't modified the initial configuration in which all process instances of the stored process server and the pooled workspace server run under the same server launch credential (sassrv). Host layer access from these servers is undifferentiated.

Example: UNIX and Two Levels of Host Access

The following annotated template shows how a UNIX site might provide access to servers.

Figure 11.8 Authentication Scenarios: UNIX Example



Here are some points about this example:

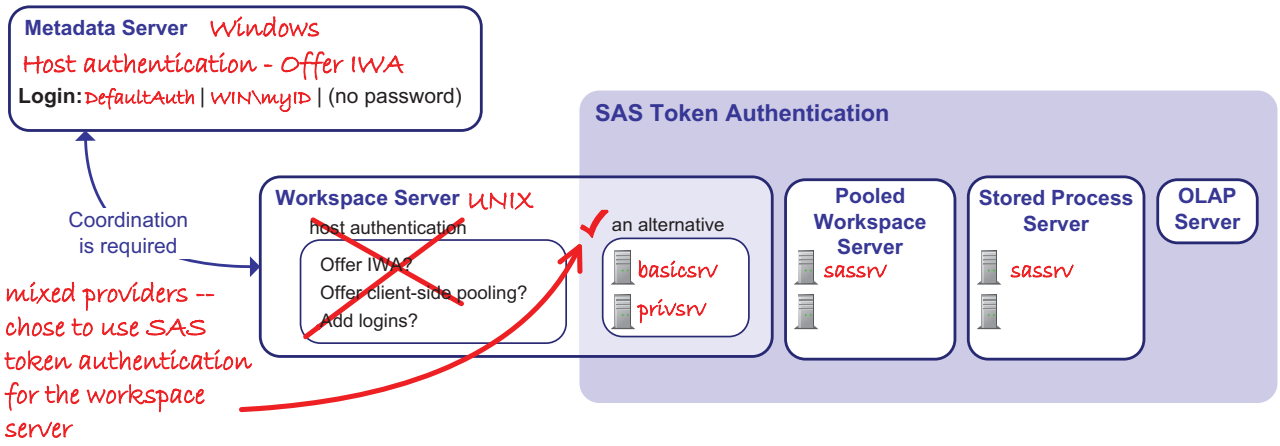
- This site can't offer Integrated Windows authentication (IWA) because the metadata server isn't on Windows.
- This site chose to establish two levels of host access from the pooled workspace server. Notice that there are two servers under that logical server, each with a different launch credential. The site gave each credential different host layer access to SAS data sets. The ReadMetadata permission settings on each server determine who uses that server. See "Hiding Server Definitions" on page 88.

  Note: Other reasons for defining multiple servers below a logical server include using multiple computers or different start-up scripts. △

Example: Mixed Hosts and SAS Token Authentication

The following annotated template shows how a site that has mixed providers might provide access to servers. This site chose the solution of configuring the workspace server to use SAS token authentication. See "Mixed Providers" on page 135.

Figure 11.9 Authentication Scenarios: Mixed Providers Example



A side effect of using SAS token authentication for the workspace server is that host access from the workspace server no longer occurs under the host identity of each requesting user. This site chose to establish two levels of host access from the logical workspace server.

Note: The requesting user's SAS identity is still preserved for metadata layer access control and logging purposes. △

Mixed Providers

About Mixed Providers

If the credentials with which users initially log on aren't also valid for the workspace server, access to the standard workspace server isn't seamless. For example, if you use an Active Directory account in an initial logon to a metadata server on Windows, and you attempt to access a standard workspace server on UNIX, you might be prompted for a user ID and password that are valid for the UNIX host.

Note: In general, Web applications and SAS Information Map Studio use the pooled workspace server, not the standard workspace server. So users who use only these applications might not be affected by a mixed provider situation. See "Choices in Workspace Server Pooling" on page 192. △

In a mixed providers situation, achieving seamless access to the standard workspace server requires a trade-off between configuration effort, maintenance effort, and degree of segregation for host access from that server to SAS data. The following table outlines the choices.

Table 11.2 Choices for Seamless Access In a Mixed Provider Scenario

Priority	Recommendation
Minimize administrative effort.	The preferred approach is to convert the workspace server to use SAS token authentication. You can instead store a single shared password for the workspace server's host.[1]
Provide a few levels of host access with minimal administrative effort.	The preferred approach is to define a few servers that use SAS token authentication. Each server runs under a different account. You can instead store a few shared passwords for the workspace server's host.[1]
Preserve each user's identity.	The preferred approach is to align authentication so that both servers do use the same provider. You can instead store individual passwords for the workspace server's host.

1 The shared password is stored in a login on a group's **Accounts** tab. Members of a group can use that group's logins.

The following topics provide details about each approach.

Solution to Mixed Providers: Use SAS Token Authentication

If you configure the workspace server to use SAS token authentication, access is seamless because the workspace server's host operating system is no longer used to validate users. Instead, users are validated by the metadata server through a single-use SAS identity token.

In a mixed providers situation, using SAS token authentication is preferable to using a group login for these reasons:

- SAS token authentication enables the requesting user's SAS identity to be used for metadata layer evaluations such as library pre-assignment and permissions and auditing.
- SAS token authentication doesn't give users direct access to the server launch credential.
- SAS token authentication involves less transmission of reusable credentials (the client doesn't retrieve credentials from the metadata and send those credentials to the workspace server).

See "How to Configure SAS Token Authentication" on page 161.

Solution to Mixed Providers: Align Authentication

If you can enable the metadata server and the standard workspace server to use the same provider, access is seamless through credential caching and reuse.

If the metadata server is on Windows and the workspace server is on UNIX, you might use PAM to extend the UNIX host authentication process to recognize Windows accounts. See "Pluggable Authentication Modules (PAM)" on page 149. With this approach, the Windows credentials that users supply in their initial logon to the metadata server can be reused for authentication to the workspace server. User identity is preserved from the workspace server to the host, you don't have to store user passwords in the metadata, and access is seamless. However, this approach has these limitations:

- Using PAM isn't compatible with Integrated Windows authentication (IWA) for these reasons:
  - Even though the UNIX server now accepts Windows accounts, the UNIX server still can't do IWA. The SAS implementation of IWA is only for servers on Windows.
  - Because this approach reuses cached Windows credentials to provide access to the workspace server, you can't use IWA for the metadata server either. When IWA is used to get to the metadata server, only a Windows token is passed so there are no available cached credentials. Instruct users to not select the Integrated Windows authentication option in their connection profiles. Consider adding the -nosspi option to the metadata server's sasv9_usermods.cfg file.
- Using PAM to resolve a mixed providers situation requires that you stored two logins for anyone who accesses the standard workspace server. One login contains the user's ID in qualified format (for example, WIN\myID) and the other login contains the same ID in short format (for example, myID). Both logins can be in DefaultAuth. Neither login has to include a password.

If the metadata server is on UNIX (or z/OS) and the workspace server is on Windows, an additional alternative is to make the metadata server itself recognize Windows accounts. See "Direct LDAP Authentication" on page 146.

Note: Direct LDAP isn't supported for the workspace server. The workspace server can use only some form of host authentication or SAS token authentication. △

Solution to Mixed Providers: Store User IDs and Passwords

If you store user or group passwords for the workspace server's host, access is seamless through credential retrieval. In this solution, you treat the workspace server as if it were a third-party server such as Oracle. See "How to Store Passwords for the Workspace Server" on page 173.

Credential Gaps

A credential gap is a situation in which a user doesn't seamlessly access the workspace server for any of these reasons:

- Server configurations are incorrect, incomplete, or incompatible.
- The user's context doesn't include credentials that are known to the workspace server's host.
- The user's context doesn't pair credentials that are known to the workspace server's host with the workspace server's authentication domain.
- The workspace server is on Windows, using credential-based authentication and the user's host account doesn't have the **`Log on as a batch job`** privilege (see "Windows Privileges" on page 36).

The usual symptom of a credential gap is a prompt for a user ID and password after a user makes a request that requires a workspace server (for example, querying relational data). A credential gap can be problematic for these reasons:

- The prompts interrupt the user experience.
- Users have to know credentials that are valid for the workspace server's host and know that those are the correct credentials to provide.

- Not all middle-tier services and Web applications prompt for credentials (and, without a prompt, the user request fails).

You can use the questions in the following table to diagnose and troubleshoot a credential gap.

Table 11.3 Credential Gaps: Causes and Solutions

Is the user logging on with a user ID that ends in @saspw? If so, tell the user that they get the prompt because they are using an internal account. When the user gets the additional prompt, they must enter a user ID and password that are known to the workspace server's host.[1] See "Add Administrators" on page 9.
Is the user logging on with a connection profile that contains a user ID that ends in @saspw? If so, the user's context doesn't pair the credentials from the user's initial logon with the DefaultAuth authentication domain. Tell the user to create a new connection profile with external credentials (and no value in the **`Authentication domain`** field) and try again. To ensure optimal credential reuse, users shouldn't use the same connection profile for both internal and external accounts. See "Credential Management" on page 144.
Is the user using a connection profile that has a value other than DefaultAuth for the authentication domain? If so, the user's context doesn't pair the credentials from the user's initial logon with the DefaultAuth authentication domain. Tell the user to either clear this field or enter the value **`DefaultAuth`** and try again. See "Credential Management" on page 144.
Is the user in SAS Enterprise Guide and accessing a workspace server that is set to prompt? If so, verify the setting on the **`Options`** tab of the logical workspace server. If the setting is intentional, tell the user to supply host credentials.[1] See "About the Workspace Server's Options Tab" on page 177.
Do the metadata server and the workspace server accept different accounts? If so, the credentials that are added to the user's context from the user's initial logon aren't valid for the workspace server. For example, this can happen if the metadata server is on Windows and the workspace server is on UNIX. See "Mixed Providers" on page 135.
Is the workspace server assigned to the wrong authentication domain? If so, credential reuse might be impaired. In most configurations, the workspace server should be in DefaultAuth.[2] To verify (and, if necessary, correct) the workspace server's authentication domain assignment, select the **`Plug-ins`** tab in SAS Management Console, navigate to the server  , select its connection object  , right-click, and select **`Properties`**. The authentication domain assignment is on the **`Options`** tab. See "Credential Management" on page 144.
Is the user using a Web application at a site that has configured Web authentication? If so, the user's initial logon doesn't add a password to the user's context. Make sure that the Web application uses some form of pooling. See "Choices in Workspace Server Pooling" on page 192. If the problem persists, the user is performing one of a few tasks that require a standard workspace server. Either restructure the activity (for example, a stored process in an information map uses pooling but a stored process that is opened directly from SAS Web Report Studio does not) or see "Coordinate the Workspace Server" on page 8.
Is the user connecting to the metadata server with Integrated Windows authentication? If so, the user's initial logon doesn't add a password to the user's context. Configure the workspace server for IWA. See "Integrated Windows Authentication" on page 148. If the problem persists, make sure there is a logical pooled workspace server available to the user. See "Choices in Workspace Server Pooling" on page 192. Or, tell the user to deselect the IWA setting in his or her connection profile and try again. If this resolves the problem, the user must opt out of IWA when performing this task.

1 The host account must correspond to a metadata identity that has ReadMetadata permission for the server definition. On Windows, the host account must have the **`Log on as a batch job`** privilege.

2 Unless you resolved mixed providers by putting the workspace server in its own authentication domain.

How Logins Are Used

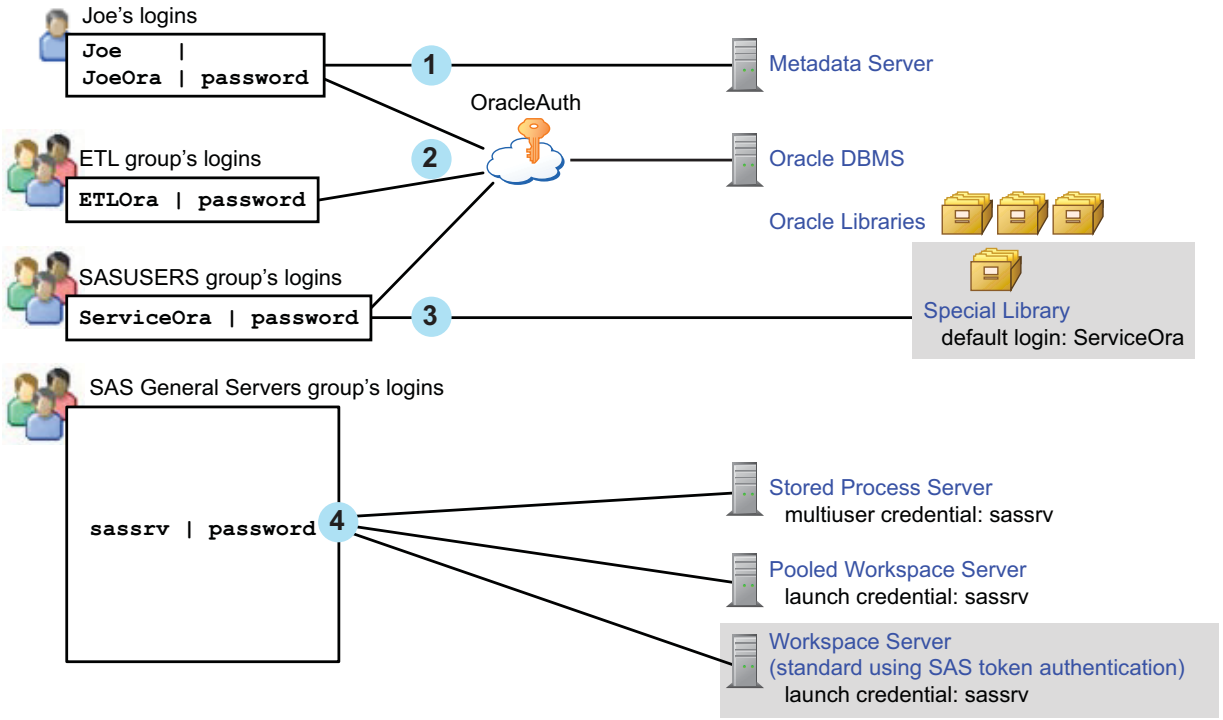
Table 11.4 Summary of How Logins are Used

Purpose	Login Properties[1]
To enable the metadata server to match an incoming user ID with a particular SAS identity (inbound use). See "How SAS Identity is Determined" on page 126.	User ID
To enable clients to seamlessly obtain user credentials for disparate systems (outbound use). See "Credential Management" on page 144.	User ID, password, authentication domain
To designate one account as the preferred account for user access to a particular library and to make that account ID and password available to users. If you assign a login to a library, all users who can see that login use that login to access that library. This is a specialized form of outbound use that is sometimes used for a DBMS library.	User ID, password, authentication domain
To designate one host account as the account under which a particular server runs and to make that account's ID and password available to the spawner (service use). See "Launch Credentials" on page 183.	User ID, password

1 Indicates which properties are involved. Every login should be assigned to an authentication domain.

The following figure depicts examples of login use.

Figure 11.10 Example: Use of Logins



Here are some general points about the figure:

- The workspace server (using host authentication) isn't depicted because access to that server is not usually through stored credentials.

Note: If you choose to store passwords for the workspace server, the relationships would be comparable to the depiction of the Oracle DBMS, OracleAuth authentication domain, and Oracle logins. For example, you might put the workspace server in WorkspaceAuth and create individual and group logins in that authentication domain. See "Mixed Providers" on page 135. △

- The OLAP server isn't depicted because that server isn't accessed using logins.
- The gray shading for the depicted workspace server indicates that this is a specialized configuration. By default the workspace server uses some form of host authentication, not SAS token authentication.

The numbers in the figure correspond to these uses:

1. Joe's first login is only for inbound use to determine his metadata identity. His password is available (cached in the user context, not stored in the metadata) but isn't used to determine his identity. This login should be in DefaultAuth, but that relationship isn't depicted because it isn't used in determining his metadata identity.
2. Joe's second login provides seamless access to Oracle using an individual account. This login includes a password and must be in the Oracle server's authentication domain. The ETL group's login is a shared login for the Oracle server. Joe won't use this login because his personal Oracle login has a higher priority. See "Identity Precedence" on page 34.
3. The SASUSERS login is a designated default login for the Special Library. This login is visible to Joe (through his automatic membership in SASUSERS), so it is used when Joe accesses the Special Library. Assigning a login to a library overrides the usual login priority evaluation (which is based on identity precedence).

   Note: In this example, the ServiceOra login must be in the OracleAuth authentication domain. The list of available default logins for a library consists of only those logins that are in the associated server's authentication domain. The gray shading for the Special Library indicates that this isn't a mainstream use; most libraries don't have a designated login. △

   Note: In an alternate usage, the default library login is part of the user definition for a service identity that provides mediated access to the library. △

4. The designated launch credential for each of the depicted processing servers is stored on the SAS General Servers group definition. In this example, the servers all use the same credential. Logins that contain designated launch credentials are usually in the DefaultAuth authentication domain, because these processing servers are usually in DefaultAuth. However, those logins are directly paired with each server, not looked up by authentication domain. Because the authentication domain assignment for these logins isn't used, the figure doesn't depict that assignment. See "Criteria for a Designated Launch Credential" on page 184.

About PUBLIC Access and Anonymous Access

In general, only users who can authenticate and who have a well-formed user definition should use a SAS deployment. However, in order to accommodate scenarios where more general access is desired, the following specialized configurations are supported:

- PUBLIC access enables unregistered users to participate if they can authenticate to the metadata server (directly or through a trust mechanism). Unregistered users are referred to as PUBLIC-only users because their only SAS identity is that

of the PUBLIC group. A PUBLIC-only user has the logins, permissions, and capabilities of the PUBLIC group. A PUBLIC-only user can't belong to any other groups, or have any personal logins, or have any individual permission settings. See "Provide PUBLIC Access (Optional)" on page 13.

- Anonymous access enables unregistered users to participate without authenticating to the SAS environment. Anonymous access is an optional configuration that is available for only a few applications. Anonymous access is supported only with SAS authentication; anonymous access is not compatible with Web authentication. Anonymous access is supported as follows:
  - For SAS BI Web Services and the SAS Stored Process Web Application, a user who connects through anonymous access uses the SAS Anonymous Web User identity. This is a service identity that functions as a surrogate for users who connect without supplying credentials. For more information, see the *SAS Intelligence Platform: Web Application Administration Guide*.
  - For the SAS Information Delivery Portal (release 4.3 and later), a user who connects through anonymous access uses the Unchallenged Access User identity. This is a service identity that functions as a surrogate for users who connect without supplying credentials. For more information, see the documentation on unchallenged portal access in the *SAS Intelligence Platform: Web Application Administration Guide*.

The following list highlights differences between PUBLIC access and anonymous access:

- In PUBLIC access, each participating user must authenticate. In anonymous access, participating connections don't require user authentication.
- In PUBLIC access, participating users share the PUBLIC group identity. In anonymous access, participating connections share a designated service identity (the surrogate identity is always a member of both the SASUSERS group and the PUBLIC group).
- You can choose to provide wide support for PUBLIC access. You can't extend support for anonymous access beyond the specific applications that can be configured to use it.

CAUTION:

If you choose to offer PUBLIC or anonymous access, you risk users seeing more data and content than you might expect. Carefully review and manage access control for the PUBLIC group. If you offer anonymous access, carefully review and manage access for your surrogate service identity too. △

CHAPTER

12

Authentication Mechanisms

Introduction to Authentication Mechanisms

This chapter describes the following technologies, features, and trust relationships:

- Internal mechanisms unify the SAS realm and provide a degree of independence from your general computing environment. The internal mechanisms are SAS internal authentication and SAS token authentication.
- External mechanisms integrate SAS into your computing environment. External mechanisms include direct LDAP authentication, host authentication (credential-based), Integrated Windows authentication, and Web authentication.
- Supporting features for credential-based authentication include the following:
  - Credential management provides single sign-on through reuse of cached credentials or retrieval of stored passwords.
  - Pluggable authentication modules (PAM) extend UNIX host authentication.
- Trust relationships facilitate communication to the metadata server by permitting one privileged account to connect on behalf of other users (trusted user) or by accepting requests that use a proprietary protocol (trusted peer).

Credential Management

Table 12.1 Credential Management

Summary	A supporting feature in which clients reuse cached credentials or retrieve stored credentials. Clients use authentication domain assignments to determine which credentials are valid for which servers. The target server validates the client-supplied credentials against its authentication provider.
Scope	From clients that are already connected to the metadata server to third-party servers, the Scalable Performance Data Server, and, in some cases, the workspace server.
Benefits	Provides access to servers using individual or shared accounts.
Limits	□ Involves passing user IDs and passwords across the network. □ Can involve maintaining SAS copies of external passwords.
Use	Always available

Credential management techniques populate an in-memory list of credentials for each connected user. Each list is called a user context and includes these entries:

- If the user interactively provides credentials when launching a SAS client, those credentials are added to the list, with these exceptions:
  - If Web authentication is configured, the password from a user's interactive logon to a Web client isn't available to be added to the list.
  - If the user logs on with Integrated Windows authentication (IWA), the user's password isn't available to be added to the list.
- If the user interactively provides credentials at any point in the session, those credentials are added to the list.
- If the user's **Accounts** tab has logins that include passwords, those credentials are added to the list.
- If the user belongs to any groups whose **Accounts** tab has logins that include passwords, those credentials are added to the list.

Note: Credentials from a user or group's **Accounts** tab are not included in the initial list that is created when a user logs on. Instead, such credentials are added to the list dynamically (when and if they are needed in the course of the user's session). △

The following table depicts an example of the contents of a user context:

Table 12.2 Example: Contents of a User Context

User ID	Password	Authentication Domain
myWinID	********	DefaultAuth
GroupDBMSid	********	DBMSauth

Notice that each entry is assigned to an authentication domain. This enables pairing of credentials with the servers for which they are valid. The entries are created as follows:

- The client creates the first entry by caching and inserting the user ID and password that a user submits in an initial logon. This makes the user's

DefaultAuth password available for reuse even though that password isn't stored in the metadata. The client automatically assigns the first entry to the DefaultAuth authentication domain except in these circumstances:

- The user's connection profile contains a user ID with an @saspw suffix.

  Note: Notice that this circumstance describes the user ID in the user's connection profile, not the user ID that the user supplies in the logon dialog box. △

- The **Authentication domain** field in the user's connection profile contains a value other than DefaultAuth (and isn't empty).
- The user is accessing a Web client at a site that has configured Web authentication (or has specified a different authentication domain in the Web configuration for some other reason).

- The client creates the second entry by retrieving information from the metadata (in the preceding example, from the **Accounts** tab of a group that the user belongs to). Such logins are for outbound use, so they must include a password and an appropriate authentication domain assignment. See "How Logins Are Used" on page 139.

When a user requests access to a server that requires credential-based authentication, the client completes these steps:

1. Examine the server's metadata to determine which authentication domain the server belongs to. This information is on the **Options** tab of each of the server's connection objects  in SAS Management Console.
2. Examine the user's context to determine whether it includes any credentials that are assigned to the target server's authentication domain. The process is as follows:
   - If the context includes a cached entry for the target authentication domain, that entry is used.
   - If the user context contains a retrieved entry for that authentication domain, that entry is used. If there is more than one retrieved entry for an authentication domain, the entry that is closest to the user is used. See "Identity Precedence" on page 34.
   - If there is an identity precedence tie among retrieved entries (for example, if a user is a direct member of two groups and both groups have logins in the relevant authentication domain), the same login is used consistently, but you can't control which of the two logins is used.
   - If the user context contains no entries in the target authentication domain, desktop clients will prompt the user for credentials. Web applications can't prompt.

     Note: SAS Web Report Studio has an interactive password management feature. See the *SAS Intelligence Platform: Web Application Administration Guide*. △
3. Present the credentials to the target server for authentication against its provider.

Here are some additional tips:

- Because the credentials that are added to a user context from an initial logon are not likely to be valid for a third-party DBMS, you usually have to store credentials for such servers. See "How to Store Passwords for a Third-Party Server" on page 174.
- Authentication domains have no effect on an initial logon. Authentication domains affect access to secondary servers that perform credential-based authentication.

- To prevent attempts to reuse internal credentials for the workspace server (which doesn't accept internal credentials), users who have an internal account should use a dedicated connection profile for that account. In their internal connection profiles, users should leave the **`Authentication domain`** field blank (or specify an arbitrary value such as InternalAuth).

Direct LDAP Authentication

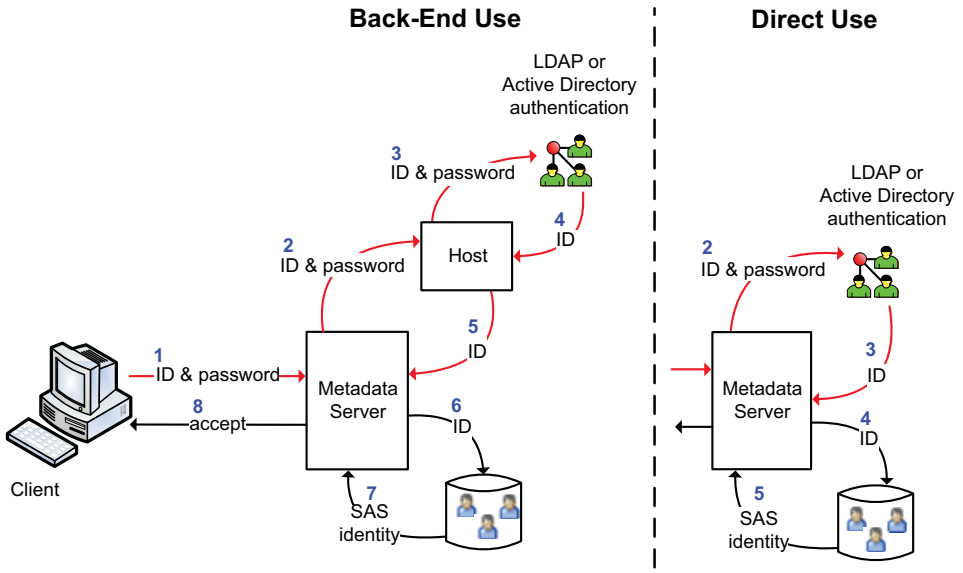
Table 12.3 Direct Use of LDAP Authentication

Summary	The metadata server validates some users against an LDAP provider such as Active Directory.
Scope	□ Primarily used for connections to the metadata server. □ Can also be used for direct connections from a data provider to the OLAP server.
Benefits	Enables users to use their Windows accounts to authenticate to a metadata server that runs on UNIX.
Limits	□ Not an alternative to storing user IDs in the metadata (that requirement applies to all configurations). □ Not supported for workspace servers or stored process servers. □ If you are using external accounts for sasadm and sastrust, requires manual updates to those user IDs in configuration files. □ Can involve appending a special suffix to user IDs that are stored in the metadata.
Use	Optional

1 Direct LDAP enables the metadata server to recognize accounts that aren't known to its host; direct LDAP doesn't modify the host's behavior.

The following figure contrasts back-end use and direct use.

Figure 12.1 Two Ways to Use an LDAP Authentication Provider



Many hosts use an LDAP provider as a back-end authentication mechanism. From the perspective of the SAS server, this is host authentication, so no direct LDAP configuration is needed. For example:

- Active Directory is the standard back-end authentication provider on Windows.
- Some UNIX hosts recognize LDAP accounts (or can be configured to do so). See "Pluggable Authentication Modules (PAM)" on page 149.

See "How to Configure Direct LDAP Authentication" on page 167.

Host Authentication

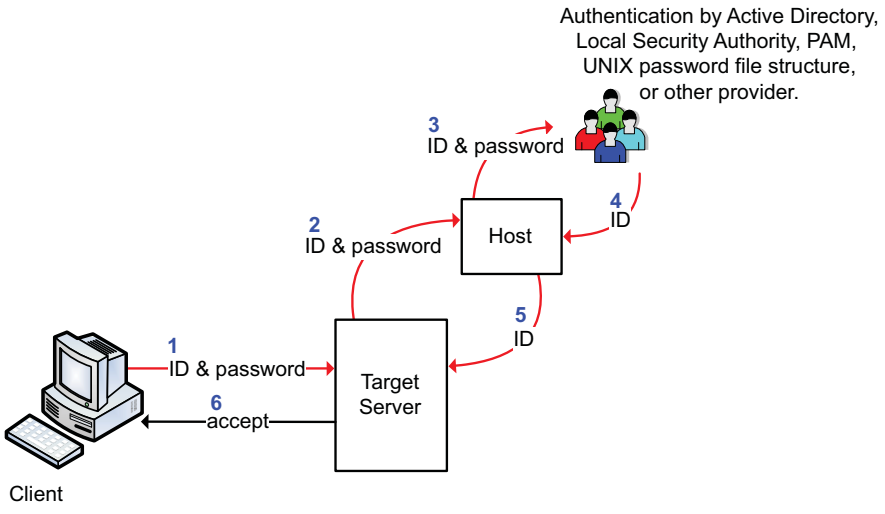
Table 12.4 Host Authentication (Credential-Based)

Summary	A client supplies an external user ID and password to a SAS server. The SAS server passes the credentials to its host for authentication.[1]
Scope	□ Primarily used for direct connections to the metadata server and the OLAP server. □ Not used for metadata-aware connections to the OLAP server or the stored process server. □ Sometimes used for connections to the workspace server.
Benefits	No configuration is required. Can enable users to log on to SAS applications with the same credentials that they use in your general computing environment.
Limits	□ On a workspace server on Windows, requires that users have the **`Log on as a batch job`** privilege. See "Windows Privileges" on page 36. □ Involves passing user IDs and passwords across the network.
Use	Always available

1 Another form of host authentication, Integrated Windows authentication (IWA), is documented separately.

The following figure shows one example of how this mechanism works:

Figure 12.2 Host Authentication (credential-based)



The numbers in the figure correspond to these actions:

1. The client obtains the user's ID and password (interactively or through credential management). The client sends those credentials to the target server.
2. The server passes the credentials to its host for authentication.
3. The host passes the credentials to its authentication provider.
4. After verifying that the user ID and password correspond to a valid account, the host's authentication provider returns the user's ID to the host.
5. The host returns the user's ID to the SAS server.
6. The server accepts the client connection.

Integrated Windows Authentication

Table 12.5 Integrated Windows Authentication (IWA)

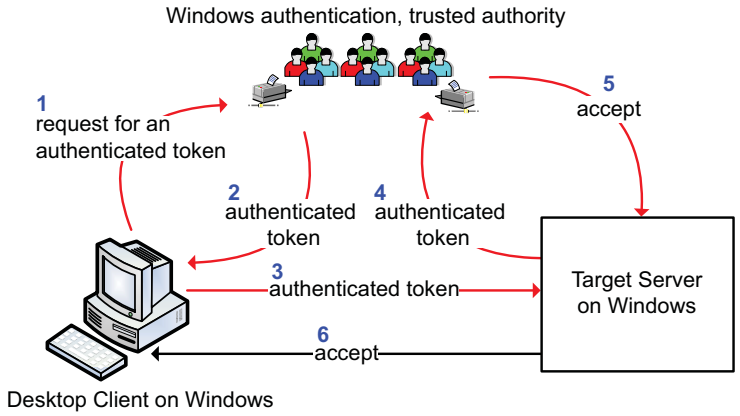
Summary	A Microsoft technology generates and validates Windows identity tokens. This has the effect of causing participating SAS servers to accept users who are authenticated to their Windows desktop. The SAS implementation of IWA supports only desktop clients and servers on Windows.[1]
Scope	□ Primarily used for connections to the metadata server and the workspace server. □ Also supported for direct connections to an OLAP server (for example, from a data provider).
Benefits	□ Bypasses the initial logon prompt. Accommodates logon mechanisms that are not password-based (such as smart cards or biometrics). □ No user credentials are transmitted. Users don't need the **`Log on as a batch job`** privilege.
Limits	□ Clients and servers must be in the same Windows domain or in domains that trust each other.[2] □ If you use IWA for a workspace server that accesses Windows network resources, the Kerberos protocol must be used and the object spawner account must have the **`trusted for delegation`** Windows privilege. □ The SAS implementation of IWA is not supported for Web applications (or analytics platform clients such as SAS Enterprise Miner). □ If you use IWA for the metadata server, there are no cached credentials from an initial logon. For this reason, if the workspace server is also on Windows, it is a good idea to configure IWA for that server also. □ If your SAS servers use DNS aliases, you must manually register those aliases in order to support Kerberos-based IWA connections. See the discussion of custom SPNs in Table 13.6 on page 172.
Use	Optional

1 If you configure Web authentication, Web applications might use IWA. This occurs if your Web environment offers IWA. This isn't part of the SAS implementation of IWA, which is for desktop applications only.

2 The SAS implementation of IWA doesn't support servers on hosts other than Windows. Configuring a server on UNIX to accept Active Directory accounts (by using PAM or configuring direct LDAP) doesn't mitigate this requirement.

The following figure is an abstraction of how this mechanism works.

Figure 12.3 Integrated Windows Authentication



The numbers in the figure correspond to these actions:

1. The client asks Windows for a token that represents the user who is currently logged on to the client computer. This step is initiated when a user launches a desktop client or makes a request for a workspace server.

   Note: The token represents the client who is running the SAS application. If you use the Windows **runas** command to launch a SAS application, the token represents the host account that you chose to use. △

2. Windows provides the token to the client.
3. The client sends the Windows token to the target server. Notice that only the token is sent; the user's password isn't available to the target server.
4. The target server sends the token back to Windows for verification.
5. Windows tells the target server that the token is valid.
6. The target server accepts the connection from the client. The user's ID is provided to the target server in domain-qualified format (for example, WIN\joe).

   Note: If the target server is the metadata server, the SAS identity phase follows. See "How SAS Identity is Determined" on page 126. △

See Also

"How to Configure Integrated Windows Authentication" on page 169

Pluggable Authentication Modules (PAM)

Table 12.6 PAM (Pluggable Authentication Modules)

Summary	A supporting feature that extends UNIX host authentication to recognize an additional provider such as Active Directory. When a SAS server asks its UNIX host to validate a user's credentials, the host sends the user's ID and password to the configured additional provider for verification.[1]
Scope	Affects all SAS servers that run on the UNIX host and rely on the host operating system to authenticate users. Typically, the metadata server and the workspace server use host authentication.
Benefits	Can be used to enable users to use their Windows accounts to authenticate to a metadata server or workspace server that run on UNIX.

Limits	□ Not an alternative to storing user IDs in the metadata (that requirement applies to all configurations). □ You can't use Integrated Windows authentication with this method. The SAS implementation of IWA is only for servers that run on Windows.
Use	Optional

1 PAM extends the host's authentication process to recognize an additional provider; PAM doesn't modify the metadata server's behavior.

This mechanism is useful if both the metadata server and the workspace server are on UNIX and you want users to use Windows accounts to access these servers.

This mechanism can also be useful if one of these servers is on Windows, the other is on UNIX, and you want to avoid credential prompts for the workspace server. However, if you use PAM to resolve a mixed provider situation, users who access the workspace server must have two logins. One login should include the user's ID in its qualified form. The other login should include the same ID in short (unqualified) form. Both logins should be in the DefaultAuth authentication domain. Neither login should include a password. For example, a user's logins might look like this:

```
DefaultAuth | WIN\joe | (no password)
DefaultAuth | joe     | (no password)
```

For configuration instructions, see the *Configuration Guide for SAS 9.2 Foundation for UNIX Environments* at **support.sas.com/installcenter**.

SAS Internal Authentication

Table 12.7 Internal Authentication

Summary	Participating SAS servers validate incoming user IDs that have a special suffix (@saspw) against a list of accounts that exist only in the metadata.
Scope	□ Primarily used for connections to the metadata server. □ Also supported for direct connections to the OLAP server.[1]
Benefits	□ Minimizes the need to create external accounts for service identities. □ Facilitates intermittent use of an administrative role by enabling a user to have a second metadata identity without having a second external account. See "Add Administrators" on page 9. □ Provides independence from the rest of your computing environment. For example, internal accounts don't have to follow your general password change requirements, can't be used to access resources beyond the SAS metadata, and aren't affected by changes in machine names or in your authentication configurations.

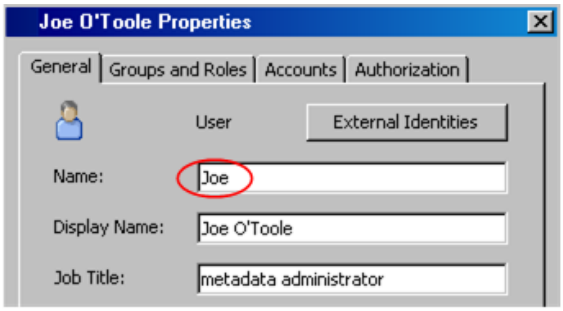
Limits

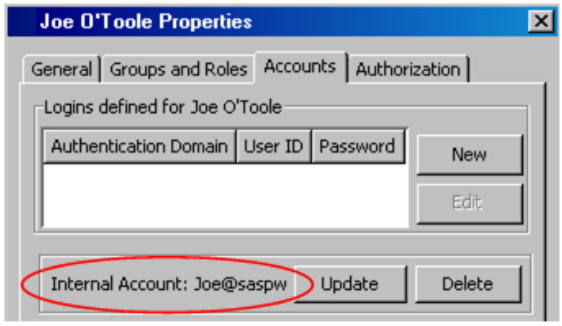
- Internal accounts aren't intended for regular users (they are intended for only metadata administrators and some service identities).
- Someone who has only an internal account can't seamlessly access the workspace server. See "Credential Gaps" on page 137.
- An internal account can't participate in Integrated Windows authentication or Web authentication.
- You can't use an internal account to delete, unregister, add, or initialize a foundation repository.
- You can't run a server under an internal account. For example, the SAS General Server User (sassrv) must be a host account.

Use	Always available

1 This isn't a common use. In the SQL procedure, a connection to the OLAP server can be made using an internal account.

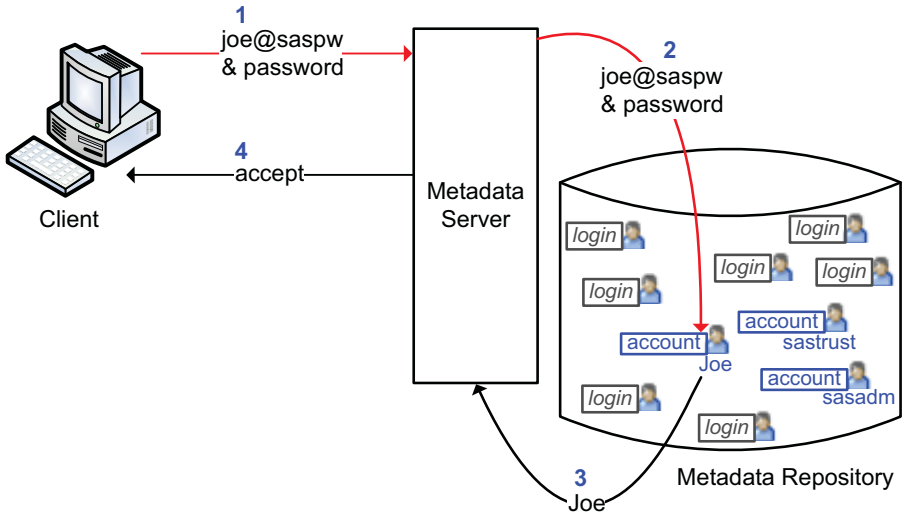
Internal accounts exist only in the metadata and are created in the User Manager plug-in in SAS Management Console. The following displays depict the **`General`** tab and **`Accounts`** tab for an administrator named Joe. Notice that Joe's internal user ID is Joe@saspw. The user ID for an internal account is always in the format *name***`@saspw`**. The name comes from the **`Name`** field on the user's **`General`** tab.





The following figure depicts the internal authentication process:

Figure 12.4 Internal Authentication



The numbers in the preceding figure correspond to these actions:

1 At a logon prompt, Joe enters his internal credentials. The client sends those credentials to the metadata server for verification.
2 The metadata server recognizes that the ID is for an internal account (because the ID has the @saspw suffix), so the metadata server checks the credentials against its list of internal accounts.
3 After validating the ID and password, the metadata server accepts the client connection. The connection is under the identity of the SAS user who owns the account.

Note: Joe can have logins in addition to his internal account, but he doesn't need a login to establish his SAS identity. △

Here are some tips for working with internal accounts:

- There are two distinct expiration settings. Don't confuse the account expiration date with the password expiration period.
- If repeated attempts to log on with an internal account fail, make sure you are including the @saspw suffix in the user ID. Another cause of failure is the account is locked. See "How to Unlock an Internal Account" on page 29.
- If you have both an internal account and an external account, use a dedicated connection profile for your internal account. In that profile, leave the **Authentication domain** field blank. This ensures optimal credential reuse.
- If you don't have user administration capabilities, you can't see internal accounts, unless you are viewing your own definition and you happen to have an internal account.

Because internal accounts exist only in the metadata, they don't automatically follow the policies of other authentication providers. Here are the initial policies for internal accounts:

- Accounts don't expire and aren't suspended due to inactivity.
- Passwords must be at least six characters, don't have to include numbers or mixed case, and don't expire.
- The five most recent passwords can't be reused.
- After three failed attempts to log on, an account is locked for one hour. An administrator can unlock the account by accessing the **Accounts** tab in the user's definition in SAS Management Console.
- A forced password change occurs on first use and after a password is reset. This policy applies only to accounts with passwords that periodically expire. By initial policy, passwords don't expire, so forced password changes don't occur.

You can set all of these policies globally (at the server-level). You can also selectively override many of these policies on a per-account basis. See "How to Change Internal Account Policies" on page 163.

See Also

"How to Configure SAS Internal Authentication" on page 162

"Authentication to the Metadata Server" on page 126

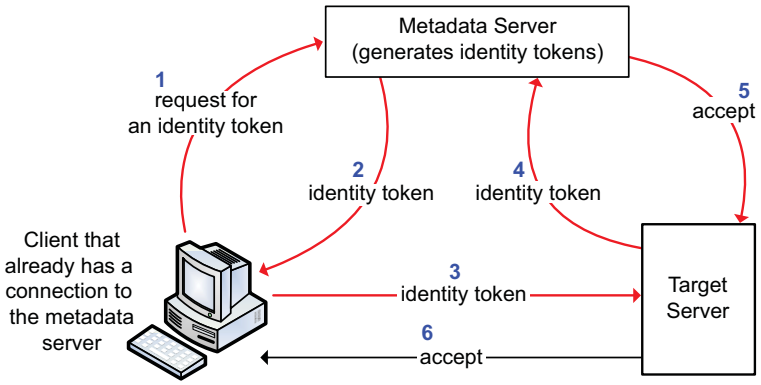
SAS Token Authentication

Table 12.8 SAS Token Authentication

Summary	The metadata server generates and validates a single-use identity token for each authentication event. This has the effect of causing participating SAS servers to accept users who are connected to the metadata server.
Scope	□ Primarily used for metadata-aware connections to the stored process server, the server-side pooled workspace server. the OLAP server, the content server, and (in a specialized configuration) the standard workspace server. □ Also used by launched servers to connect back to the metadata server (for example, from the workspace server to the metadata server for library pre-assignment).
Benefits	□ Preserves client identity for metadata layer access control and auditing purposes. □ No individual external accounts are required, no user passwords are stored in the metadata, and no reusable credentials are transmitted.
Limits	□ On the workspace server, reduces granularity of host access. See “Host Access to SAS Tables” on page 187. □ Supported only for metadata-aware connections (in which the client learns about the target server by reading the server’s metadata definition).
Use	Optional for the workspace server, otherwise mandatory within its scope

The following figure is an abstraction of how this mechanism works.

Figure 12.5 SAS Token Authentication



The numbers in the figure correspond to these actions:

1. Over the user’s existing connection to the metadata server, the client requests an identity token for the target server. This step is initiated by a user request that requires access to the target server (for example, by a request in SAS Enterprise Guide for a cube that is associated with the OLAP server).
2. The metadata server generates the token and sends it to the client.
3. The client provides the token to the target server.

4 The target server sends the token to the metadata server for validation.
5 The metadata server validates the token and returns an acceptance message and a representation of the user to the target server.
6 The target server accepts the connection.

For instructions, see "How to Configure SAS Token Authentication" on page 161.

Trusted Peer Connections

Table 12.9 Trusted Peer

Summary	The metadata server accepts peer SAS sessions and servers that connect using a proprietary protocol (trusting that those connecting identities have already been properly authenticated).
Scope	From any SAS session or SAS IOM server process to the metadata server. The scope is configurable.
Benefits	Enables a SAS/CONNECT server to access the metadata server. Facilitates connections to the metadata server during batch processing.[1]
Limits	It is important to minimize availability of this feature.
Use	Optional[2]

1 In a Windows environment, it is safer to instead use Integrated Windows authentication to support connections back to the metadata server during batch processing.
2 If you use your operating system scheduler to run metadata backup jobs, make sure that trusted peer connections from the host account that runs those jobs are allowed.

In the initial configuration, the metadata server's start command includes TRUSTSASPEER=trustedPeers.xml. The contents of the file trustedPeers.xml (which is located in your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**) make all user IDs and machines eligible to connect to the metadata server using the trusted peer protocol.

For greater security, we recommend that you target this mechanism so the metadata server does not accept every connection that uses the proprietary protocol. You can use either or both of these constraints:

- accept only specified user IDs
- accept only connections that originate from specified machines

You define the constraints in these sections in **`trustedPeers.xml`**:

`TrustedSASPeerClients`
lists eligible client types. **`SAS`** and **`java`** are the valid values. It is recommended that you reject connections from Java clients. Typically, there is only one entry between the **`TrustedSASPeerClients`** tags:

```
<client name="SAS"/>
```

`TrustedSASPeerUsers`
lists eligible user IDs. To represent all users, use an asterisk (*). To represent all users in a Windows domain, use the format *@domain. For Windows accounts, provide domain-qualified (or machine-qualified) IDs. For example, you might insert these three entries between the **`TrustedSASPeerUsers`** tags:

```
<user name="*@winXP"/>
<user name="tara"/>
```

```
<user name="batchjobID"/>
```

`TrustedSASPeerMachines`
lists eligible points of origin. Identify machines by IP address. You can use asterisks (*) as wildcards. For example, you might insert these three entries between the **`TrustedSASPeerMachines`** tags:

```
<machine ip="1.2.3.4"/>
<machine ip="A:B:C:D:E:F:1.2.3.4"/>
<machine ip="*.*.8.9"/>
```

Note: Only connections that meet all specified criteria are accepted. If any of the sections are empty, no trusted peer connections are allowed. △

Note: An additional constraint, TrustedSASDomains, is supported for backwards compatibility but will be deprecated in a future release. △

Trusted User Connections

Table 12.10 Trusted User

Summary	The metadata server allows a privileged account to act on behalf of other users (trusting that those users have already been properly authenticated).
Scope	To the metadata server from the object spawner, the OLAP server, the table server, SAS Web applications (if Web authentication is used), and batch report processes.
Benefits	Supports the optional Web authentication configuration. Enables the OLAP server and the object spawner to impersonate each requesting user on connections to the metadata server. Enables batch reporting processes to connect to the metadata server under their identities.
Limits	It is important to protect this privileged account.
Use	Required

In a standard configuration, the file trustedUsers.txt (which is located in your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**) lists one account (sastrust@saspw) that serves as the trusted user for the entire deployment.

Note: The trusted user is a privileged service identity that acts on behalf of other users. Do not add regular users to this file. △

Web Authentication

Table 12.11 Web Authentication

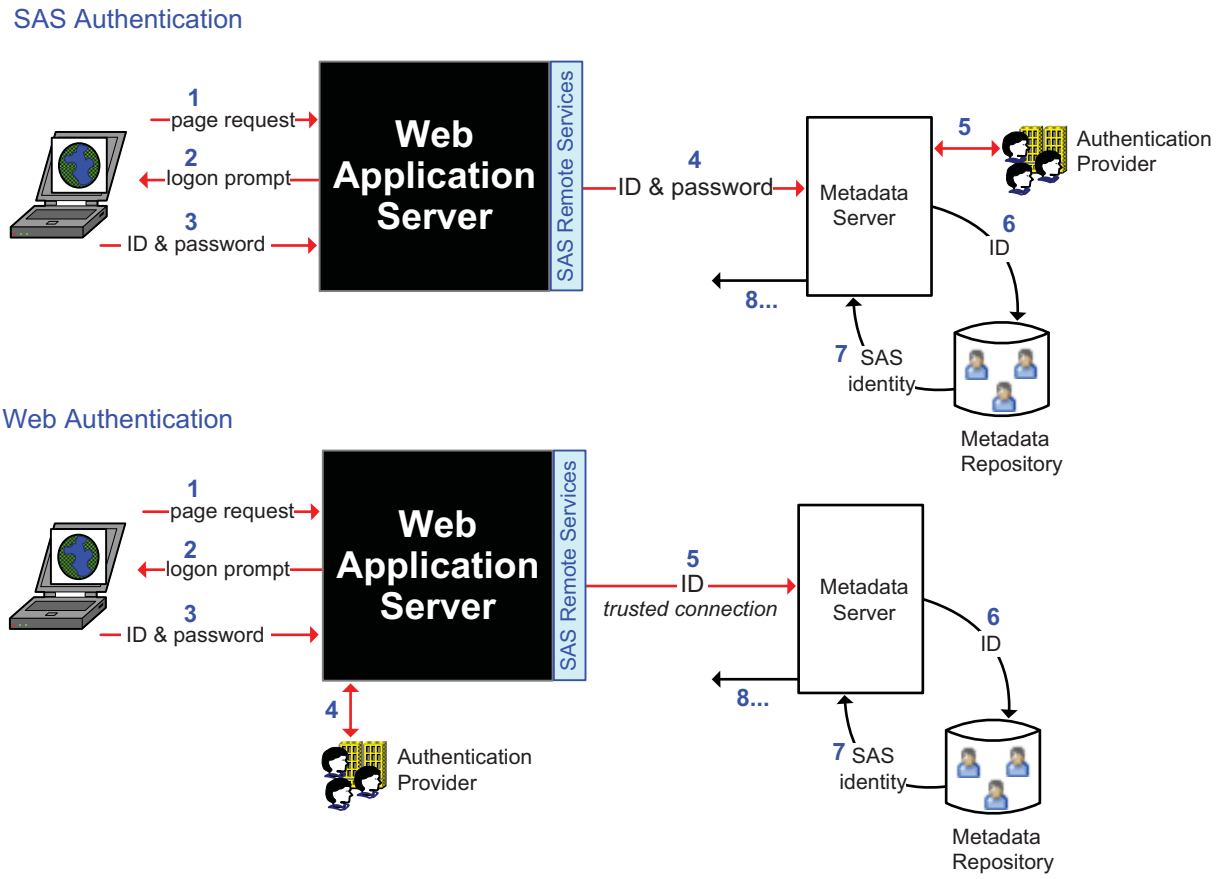
Summary	The metadata server accepts users who are authenticated at the Web perimeter.
Scope	From SAS Web applications to the metadata server.
Benefits	□ Facilitates single sign-on from and across the Web realm. □ Enables SAS Web applications to use whatever authentication scheme you have set up in your Web environment. This can facilitate integration with J2EE servlet containers, products such as SiteMinder or Tivoli Access Manager/WebSEAL, and Web servers. □ Can reduce the number of user accounts that you have to create in the metadata server's authentication provider, because users who use only Web applications no longer need accounts with the metadata server's provider.
Limits	□ When you use Web authentication to access the metadata server, there are no cached credentials from an initial logon. This prevents authentication from Web applications to the workspace server through reuse of credentials.[1] □ Prevents users from logging on to a SAS Web application with a SAS internal account. □ Not compatible with anonymous access. See "About PUBLIC Access and Anonymous Access" on page 140.
Use	Optional

1 This isn't an issue when Web applications use any kind of pooled workspace server.

The following figure depicts the high-level choice in authentication method for SAS Web applications:

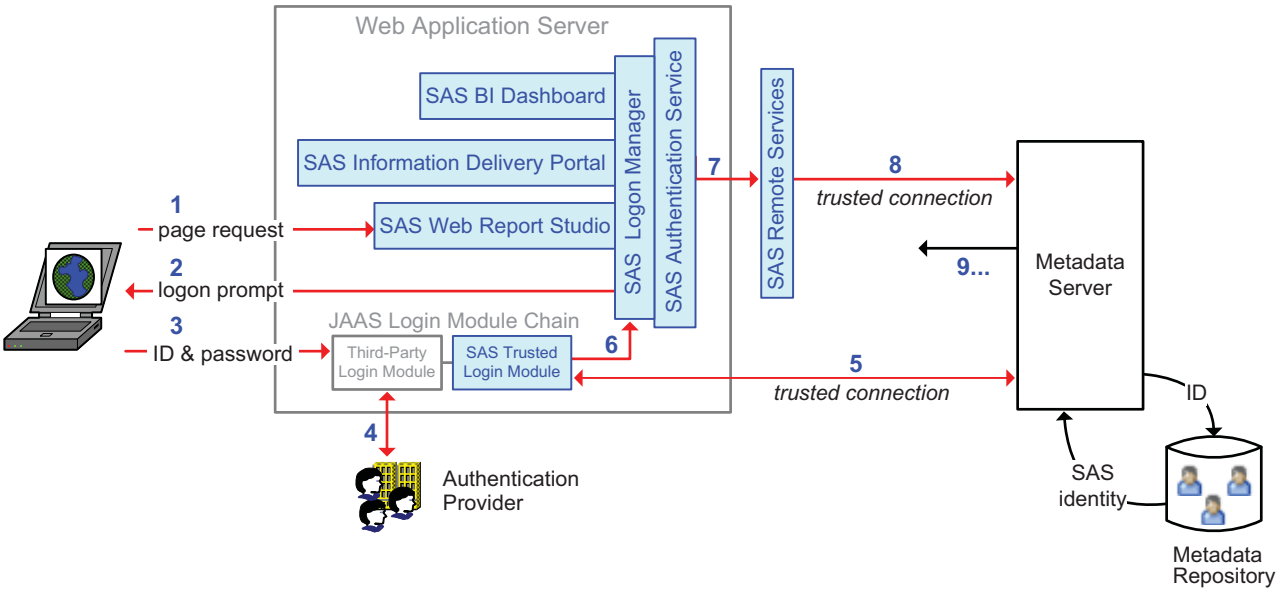
- SAS authentication (any form of authentication in which the metadata server is responsible for requesting verification)
- Web authentication (any form of authentication in which verification occurs in the Web realm and the metadata server trusts that verification)

Figure 12.6 Examples of SAS Authentication and Web Authentication



The preceding figures are simplified in order to highlight the differences between the two configurations. The following figure includes additional detail about Web authentication.

Figure 12.7 Web Authentication: A Closer Look



In the preceding figure, a user who is not already authenticated at the Web perimeter makes a request to access SAS Web Report Studio. The numbers in the figure correspond to these activities:

1. In a Web browser, the user makes the request.
2. A SAS component within the Web application server (the Logon Manager application) prompts the user for credentials. This occurs because, in this example, the user has not already been authenticated at the Web perimeter.
3. The user supplies credentials to the Web container.
4. The Web container's Java Authentication and Authorization Service (JAAS) login module chain directs the container to verify the credentials against its designated authentication provider. JAAS then passes the authenticated ID to the SAS trusted login module.
5. The SAS trusted login module uses a trusted user connection to authenticate to the metadata server and retrieve the user's SAS identity.

   Note: This is actually a two phase process in which the module first asks the metadata server for a one-time-use password for the user and then establishes a connection to the metadata server under the user's ID and the generated one-time-use password. The figure omits these details. △

   The metadata server looks up the user's ID in the metadata repository. As usual, this step doesn't involve password validation and isn't affected by authentication domain assignments. Only the user's ID is being matched (the authentication domain assignment in a login affects only credential reuse).
6. The user's authenticated ID and SAS identity is passed to the SAS Logon Manager and then to the SAS Authentication Service.
7. The authentication service passes the user's identity information to another SAS component (Remote Services), which runs in its own Java virtual machine (JVM), outside of the Web application server.
8. Remote Services initiates a second trusted user connection to the metadata server. The purpose of this connection is to obtain additional information for the user's context.

Note: As in step five, the trusted user connection is actually a two phase process. △

9 The additional user context information is returned.

See "How to Configure Web Authentication" on page 166.

Summary for Single Sign-On

There is no individual mechanism that provides end-to-end single sign-on (SSO). The following authentication processes are transparent:

- Integrated Windows authentication (IWA) is based on previous authentication to your desktop and provides silent launch for SAS desktop applications (and, sometimes, silent access to the workspace server).
- Web authentication is based on previous authentication to your Web realm and provides silent launch for SAS Web applications.
- SAS token authentication requires a connection to the metadata server and provides silent access to most SAS servers.
- Credential reuse and retrieval requires a connection to the metadata server and can provide silent access to any server.

Some configurations can interfere with SSO to back-end servers. This table summarizes the considerations:

Table 12.12 SSO Considerations for Selected Authentication Mechanisms

Feature	Front-end SSO	Back-end SSO	Notes
Internal authentication	⚠	⚠	An internal account can't participate in IWA or Web authentication.
SAS token authentication		✓	Facilitates SSO to most SAS servers.
IWA	✓	⚠	Facilitates silent launch of desktop applications. If not fully configured, prevents SSO to a workspace server on Windows.[1]
Web authentication	✓	⚠	Facilitates silent launch of Web applications. Prevents SSO to a standard (non-pooled) workspace server.[1]
Direct LDAP authentication	⚠	⚠	Not compatible with silent launch. Prevents SSO to a workspace server.[1]
PAM		✓	Can help unify authentication.
Credential Management		✓	Facilitates SSO to third-party servers and (in some configurations) workspace servers.

1 Unless the server is configured for SAS token authentication or accessed using stored credentials.

Summary by Server Type

This table provides a high-level review of support for different authentication mechanisms. In the table, the following symbols indicate the extent to which each server can be accessed using each feature:

● Supported
◒ Supported for only direct connections (not metadata aware connections)
◔ Intended for only administrators and some service identities
⊙ Used only for certain server-to-server communications
○ Not supported

Table 12.13 Summary: How Servers Can be Accessed

Mechanism	Server Type				
	Metadata	**Workspace**	**OLAP**	**Stored Process or Pooled Workspace**	**Client-Pooled Workspace**
Host authentication (credentials)	●	●	◒	○	⊙
IWA	●	●	◒	○	○
Web authentication	●	○	○	○	○
Direct LDAP authentication	●	○	◒	○	○
Internal authentication	◔	○	◔	◔	○
SAS token authentication	⊙	●	●	●	○
Trusted user	⊙	○	○	○	○
Trusted peer	⊙	○	○	○	○

1 For the client-pooled workspace server, user access depends on membership in a puddle group.

CHAPTER

13

Authentication Tasks

How to Configure SAS Token Authentication

For metadata-aware connections to the stored process server, the pooled workspace server, and the OLAP server, SAS token authentication is always used and no configuration is involved. See "SAS Token Authentication" on page 153.

To configure the standard workspace server to use SAS token authentication:

Note: This is not a universally necessary task. Configuring the standard workspace server to use SAS token authentication is one of several solutions to a mixed provider situation. See "Mixed Providers" on page 135. △

1. Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).
2. On the **Plug-ins** tab, expand **Server Manager** and the application server  (for example, **SASApp**). Right-click the logical server  (for example, **SASApp - Logical Workspace Server**) and select **Properties**.
3. On the **Options** tab, select **SAS token authentication**. Click **OK** to save this setting.
4. Expand the logical server  , select the server  , right-click, and select **Properties**.
5. On the **Options** tab, from the **Launch Credentials** drop-down list, select a login and click **OK**. The most basic choice is the SAS Spawned Servers account (the

sassrv login). If you want to use another account, see "Criteria for a Designated Launch Credential" on page 184.

6 To make the changes take effect, refresh the spawner's metadata.

   a Expand the object spawner  , right-click the computer  , and select **Connect**.
   b Right-click the computer again and select **Refresh Spawner**. In the message box, click **Yes**.
   c Right-click the computer a third time and select **Disconnect**.

7 To validate the configuration, select the logical workspace server  , right-click, and select **Validate**.

Metadata layer access from the server (for example, library pre-assignment, PROC OLAP code, and metadata LIBNAME engine use) will still be based on each requesting user's identity. However, all host access from the server will now be under the designated launch credential. See "Example: Multiple Levels of Host Access" on page 190.

How to Configure SAS Internal Authentication

Note: There are no server configuration activities for SAS internal authentication. The metadata server always accepts valid internal account credentials. Internal accounts aren't intended for regular users. See "SAS Internal Authentication" on page 150. △

To create and use an internal account:

1 Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).
2 On the **Plug-ins** tab, select **User Manager**. Make sure you are in the foundation repository.
3 Right-click **User Manager** and select **New ▶ User**. On the **General** tab, enter a short word such as **test** in the **Name** field.
4 On the **Accounts** tab, click **Create Internal Account**. In the New Internal Account dialog box:

   a Notice that the user ID is constructed from the name that you entered on the **General** tab and an @saspw suffix.
   b Enter and confirm a simple initial password such as 123456.

   Note: These instructions assume that the default server-level policies for internal accounts are in place. △

   c Select the **Set a custom password expiration period** check box and the **expires every** radio button. Leave the password expiration period at the default number of days.
   d Click **OK** to save the new internal account.

5 Notice that the new account appears at the bottom of the **Accounts** tab. Click **OK** to save the new user.
6 To log on to SAS Management Console as the new internal user:

   a From the main menu, select **File ▶ Connection Profile**. In the informational message box, click **Yes**.
   b In the Connection Profile dialog box, select **Create a new connection profile** and click OK.

c In the Connection Profile wizard, name the profile **internal**, provide the machine name and port of the metadata server, and enter the internal credentials (for example, **test@saspw** and **123456**). Click **Finish**.

d In the Change Password dialog box, enter the original password again, enter and confirm a new password (such as **223456**), and click **OK**.

Note: You are forced to change the password because this is your first use of the account. This happens only for accounts that have a password expiration period (which you set in step 4c), and can also be affected by server-level policies. △

e In the message box, click **OK**. In the connection profile, enter the new password and click **Finish**.

f If necessary, provide your internal credentials once again to log on.

7 Notice that you have the permissions and capabilities of the SASUSERS and PUBLIC groups (because you didn't make any additional group or role assignments for the test user in step 4).

Note: To clean up, log back on as someone who has user administration capabilities. In **User Manager**, delete the user that you created for this exercise. To delete the test user's home directory and MyFolder, select the **Folders** tab, navigate to **SAS Folders** ▸ **Users**, right-click the test folder, and select **Delete**. △

See Also

"How to Unlock an Internal Account" on page 29

"How to Change Internal Account Policies" on page 163

"How to Create a Dual User" on page 29

How to Change Internal Account Policies

Server-Level Policies

Here are the initial server-level policies for internal accounts:

- Accounts don't expire and aren't suspended due to inactivity.
- Passwords must be at least six characters, don't have to include numbers or mixed case, and don't expire.
- The five most recent passwords can't be reused.
- After three failed attempts to log on, an account is locked for one hour. An administrator can unlock the account by accessing the **Accounts** tab in the user's definition in SAS Management Console.
- A forced password change occurs on first use and after a password is reset. This policy applies only to accounts with passwords that periodically expire. By initial policy, passwords don't expire, so forced password changes don't occur.

To change these settings for all internal accounts (except those that have an overriding per-account setting), edit the metadata server's omaconfig.xml file and restart that server. Here is the syntax:

Note: The following option names are case-sensitive. △

Note: A value of **T** has aliases (**1** or **Y**). A value of **F** has aliases (**0** or **N**). △

ChangeDelayInMinutes="*number*"
specifies the number of minutes that must elapse between password changes. Applies only when you are resetting your own password.

Range: 0–1440

Default: 0

DigitRequired="T | F"
specifies whether passwords must include at least one digit. To enforce this requirement, specify **T**.

Default: F

ExpirationDays="*number*"
specifies the number of days after password is set that the password expires. A value of 0 prevents passwords from expiring.

Range: 0–32767

Default: 0

ExpirePasswordOnReset="T | F"
specifies whether a forced password change occurs on first use and after an administrative password reset. To disable this requirement, specify **F**.

Default: T

Exceptions: This option affects only accounts with passwords that expire and doesn't apply when you reset your own password.

MinLength="*number-of-characters*"
specifies the minimum length for passwords.

Range: 1–32

Default: 6

MixedCase="T | F"
specifies whether passwords must include at least one upper case letter and at least one lower case letter. To enforce this requirement, specify **T**.

Default: F

NumPriorPasswords="*number*"
specifies the number of passwords that are maintained in each account's password history. A user can't reuse a password that is in the user's account history.

Range: 0–5

Default: 5

InactiveDaysToSuspension="*number*"
specifies the number of days after which an unused account is suspended. A value of 0 prevents suspensions due to inactivity.

Range: 0–32767

Default: 0

LockoutDurationInMinutes="*number*"
specifies the number of minutes for which an account is locked following excessive login failures.

Range: $1–2^{31}$

Default: 60

NumFailuresForLockout="*number*"
specifies the number of consecutive unsuccessful logon attempts that cause an account to be locked. We recommend that you do not specify 0, because doing so can make your system vulnerable to password guessing attacks.

Range: 0–100
Default: 3

Per-Account Policies

To override server-level policies on a per-account basis:

1. Log on to SAS Management Console as someone who has user administration capabilities.
2. On the **Plug-ins** tab, select **User Manager** (in the foundation repository).
3. In the display pane, clear the **Show Groups** and **Show Roles** check boxes. Right-click the user definition of the user whose SAS internal account policies you want to change. Select **Properties**.
4. At the bottom of the user's **Accounts** tab, click **Update**.
5. Make changes in the **Custom Settings** box. Not all server-level settings can be modified on a per-account basis.

   Note: There are two distinct expiration settings. Don't confuse the account expiration date with the password expiration period. △

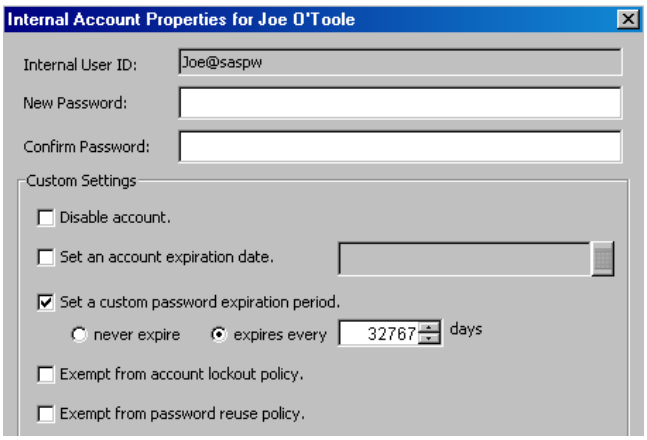
   Note: To minimize administrative maintenance effort for any predefined or service identities that have internal accounts, don't add expiration dates to these accounts or expiration periods to these passwords. △

   The following table maps server-level policies to corresponding account-level policies. Not all policies can be set at both levels.

Table 13.1 Internal Account Policy Mapping

Server-Level Policy	Related Account Level Setting
ExpirationDays	Set a custom password expiration period.
LockoutDurationinMinutes **NumFailuresForLockout**	Exempt from account lockout policy.
NumPriorPasswords	Exempt from password reuse policy.

For example, if you want to force a particular user to change his or her internal password after you create (or reset) the user's internal account, but you don't otherwise want the password to expire, set per-account settings as depicted in the following display.



By using the maximum password expiration period, 32767 days (approximately 89 years) you force a password change on first use but don't require any further password updates in a plausible time frame.

See Also

"SAS Internal Authentication" on page 150

"How to Configure SAS Internal Authentication" on page 162

How to Configure Web Authentication

Overview of Configuring Web Authentication

Note: Before you configure Web authentication, verify that this is an appropriate choice in your environment. See "Web Authentication" on page 156. △

Configuring Web authentication consists of the following high-level tasks:

1. Make configuration changes to SAS components (for example, edit the SAS login.config file to reference the **web** domain, add security information to the SAS Logon Manager application, and adjust the classpath for the Remote Services application).
2. Make configuration changes to your Web environment (for example, add login modules and make SAS JAR files available).
3. Rebuild and redeploy the SAS Web applications. Update and restart the Web application server.
4. Verify or adjust user information in the SAS metadata so that each user who participates in Web authentication has an appropriate login in his or her metadata definition.

Vendor-Specific Instructions for Web Authentication

Many of the implementation details for Web authentication differ by product. For this reason, instructions for setting up Web authentication in JBoss, WebSphere, and WebLogic are available in separate documents from **support.sas.com/thirdpartysupport**.

Logins for Users Who Participate in Web Authentication

If you choose to configure Web authentication, make sure that user metadata definitions include logins as explained in this topic.

Someone who uses only Web applications should have a login in the web authentication domain. For example:

```
web           | joe       | (no password)
```

Someone who uses both Web and desktop applications might need two logins. One login contains the user's authenticated ID after logging on to a desktop application, and the other login contains the user's authenticated ID after logging on to a Web application. For example:

```
DefaultAuth   | WIN\joe   | (no password)
web           | joe       | (no password)
```

In the preceding example, two logins are needed because the format of the authenticated user ID differs in each context as follows:

- When Joe logs on to a desktop application (as joe), SAS obtains his user ID in down-level format (WIN\joe), and that string is matched to the user ID in Joe's DefaultAuth login.
- When Joe logs on to a Web application (as joe), SAS obtains his user ID in short format (joe), and that string is matched to the user ID in Joe's web login.

However, if the authenticated user ID is identical in both contexts, the web login is not needed. If SAS obtains both authenticated user IDs as joe, the web login is not needed. In the following example, the metadata server is not authenticating against Windows accounts and the web login is not needed. When Joe logs on to a Web application, the presence of his DefaultAuth login (which contains the correct user ID) is sufficient for the metadata server to successfully determine his metadata identity.

```
DefaultAuth   | joe  | (no password)
web           | joe  | (no password)
```

Note: If your Web environment uses Integrated Windows authentication, you must pay careful attention to the format in which SAS obtains user IDs from the Web realm. If you find that users of Web applications have only the PUBLIC identity, it is likely that the user ID in each web login is not in the same format as the user ID that SAS obtains from the Web realm. △

Note: This isn't a comprehensive discussion of logins; some users might have additional logins for other purposes. △

See Also

"How Logins Are Used" on page 139

"Add Regular Users" on page 10

"Authentication to the Metadata Server" on page 126

"User ID Formats" on page 32

How to Configure Direct LDAP Authentication

Note: This is not a universally necessary task. Before you use these instructions, make sure that this is an appropriate choice in your environment. See "Direct LDAP Authentication" on page 146. △

To make a metadata server on UNIX directly recognize Active Directory accounts, locate the sasv9_usermods.cfg file that is in your equivalent of **SAS/Config/Lev1/SASMeta/MetadataServer** and add lines such as these:

```
/* Environment variables that describe your AD server */
-set AD_HOST myhost

/* System options that  make AD the primary authentication provider */
-authpd ADIR:company.com  -primpd company.com
```

You can reference only one Active Directory server. You might choose to use a Windows domain name (for example, **ADIR:MyWinDomain** instead of **ADIR:company.com**).

The preceding settings cause these results:

Table 13.2 Example: User ID Formats if -authpd ADIR:company.com -primpd company.com

How a User Logs On	Where the Metadata Server Sends the Credentials	How the User ID Must Be Stored in SAS Management Console[1]
user-ID or *user-ID*@company.com	To Active Directory	*user-ID*@company.com
*WinDomain**user-ID*	To Active Directory	*WinDomain**user-ID* or *user-ID*@*WinDomain*
user-ID@saspw	To its internal provider	No login for an internal account
user-ID@host	To its host	*user-ID*
user-ID@*anything-else*	To its host	*user-ID*@*anything-else*

1 If the ID isn't stored in the correct format, the user can log on but has only the PUBLIC identity. Put the SAS copy of each user's ID in a login on that user's **Accounts** tab. Assign each login to DefaultAuth.

Or, to make a metadata server on UNIX or Windows directly recognize some other LDAP provider, use lines such as these:

```
/* Environment variables that describe your LDAP server */
-set LDAP_HOST myhost
-set LDAP_BASE "ou=emp, o=us"

/* System options that make LDAP the primary authentication provider */
-authpd LDAP:company.com -primpd company.com
```

You can reference only one LDAP server.

The preceding settings cause these results:

Table 13.3 Example: User ID Formats if -authpd LDAP:company.com -primpd company.com

How a User Logs On	Where the Metadata Server Sends the Credentials	How the User ID Must Be Stored in SAS Management Console[1]
user-ID or *user-ID*@company.com	To LDAP	*user-ID*@company.com[2]
user-ID@saspw	To its internal provider	No login for an internal account
user-ID@host	To its host	*user-ID*
user-ID@*anything-else*	To its host	*user-ID*@*anything-else*

1 If the ID isn't stored in the correct format, the user can log on but has only the PUBLIC identity. Put the SAS copy of each user's ID in a login on that user's **Accounts** tab. Assign each login to DefaultAuth.

2 When you use the LDAP version of -authpd, you must append the @*primpd-value* suffix in the SAS copy of each user ID that is authenticated by your LDAP provider.

To make the changes take effect, restart the metadata server. After you complete the configuration, verify that access to the workspace server isn't compromised. If access fails, see "Mixed Providers" on page 135.

Table 13.4 Reference: Environment Variables

AD_HOST	The host name of the machine where Active Directory is running.
AD_PORT	The port number for Active Directory. The default is 389.
LDAP_HOST	The host name of the machine where LDAP is running.
LDAP_PORT	The port number for LDAP. The default is 389.

LDAP_BASE	The base DN to use. For example: o=People, dc=orion, dc=com.
LDAP_IDATTR	(Optional) an alternative LDAP attribute that the SAS server can use to find your DN. The default is uid.
LDAP_PRIV_DN*	The privileged DN that is allowed to search for users. For example, cn=useradmin.
LDAP_PRIV_PW*	The password for LDAP_PRIV_DN. You can use the PWENCODE procedure to provide an encoded password.

* Set this variable only if users connect with a user ID instead of a DN, and the LDAP server does not allow anonymous binds.

Table 13.5 Reference: System Options

AUTHPD	Use this option to register and name your Active Directory provider or other LDAP provider. For complete syntax, see the AUTHPROVIDERDOMAIN SAS system option.
PRIMPD	Use this option to designate your Active Directory server or other LDAP provider as the primary authentication provider for the metadata server. The metadata server directly uses its primary provider when the submitted user ID has an `@`*your-primpd* suffix or no `@` suffix at all. Using this option enables users to log on using their usual user IDs (no special suffix is required at log on time). For complete syntax, see the PRIMARYPROVIDERDOMAIN SAS system option.

Here are some additional details:

- On UNIX, an alternate location for specifying the environment variables is in the MetadataServer.sh shell script. For example:

```
AD_HOST=myhost
export AD_HOST
```

- On z/OS, a TKMVSENV file is used to make a list of pseudo environment variables available. A TKMVSENV PDS is created at installation. To define the environment variables, create a member in the PDS that specifies the necessary variables, and then reference this PDS member in the TKMVSENV DD statement in your started task.
- Don't move inbound logins (logins that provide access to the metadata server) out of the DefaultAuth authentication domain. Doing so can interfere with credential reuse. See "Credential Management" on page 144.

How to Configure Integrated Windows Authentication

Note: This is not a universally necessary task. Before you use these instructions, verify that this is an appropriate choice in your environment. See "Integrated Windows Authentication" on page 148. △

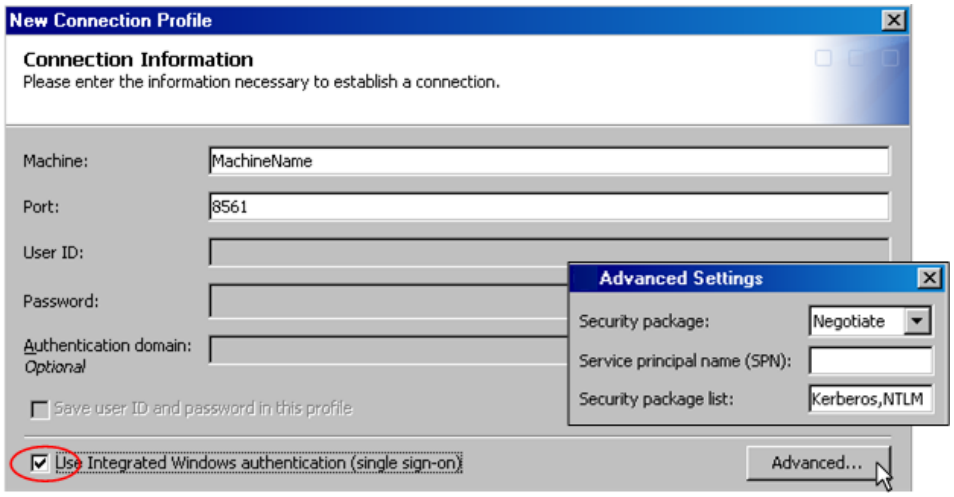
Configuration of Integrated Windows authentication (IWA) can involve three distinct locations:

- Client participation in IWA is determined by a setting in each connection profile. If IWA isn't selected by a client, it isn't used for that client.
- Server participation in IWA is affected by invocation commands. For example, the metadata server can't use IWA if that server's start command includes -nosspi.

□ For the workspace server, participation in IWA also requires certain settings in that server's metadata definition.

To configure IWA:

1 In a Java desktop application such as SAS Management Console, edit your connection profile. In the **Connection Information** panel, select the **Use Integrated Windows authentication** check box. Click **Advanced** and verify that the advanced IWA settings in your profile are as depicted in the following display.



Click **Finish** and then click **OK**. If you are now logged on to SAS Management Console, IWA to the metadata server is working. If the connection fails, verify that the metadata server's startup command includes -sspi (this is the standard setting).

Note: If you don't have a well-formed user definition that includes your Windows account ID, the status bar in SAS Management Console indicates that you are a PUBLIC-only user. In order to log back on as an administrator, you must edit your connection profile again or create a new profile. △

2 If the workspace server is on Windows, complete these steps:

a Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).

b On the **Plug-ins** tab, expand **Server Manager** and the application server (for example, **SASApp**). Right-click the logical server (for example, **SASApp - Logical Workspace Server**) and select **Properties**.

c On the **Options** tab, define settings as follows:

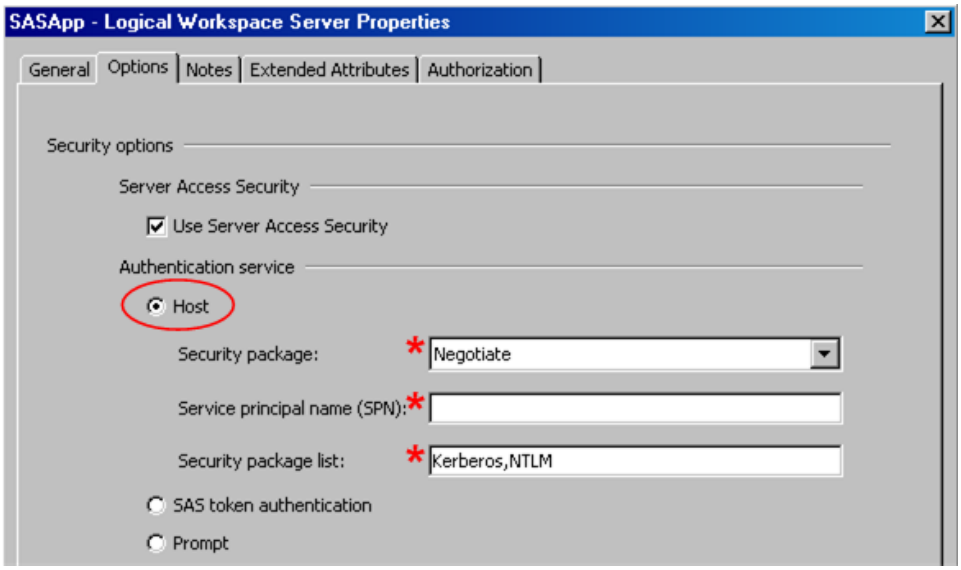
i Select the **Host** radio button. IWA is a form of host authentication.

ii Select the **Negotiate** security package.

iii Leave the **Service principal name** field empty. In a standard configuration, servers register a default SPN and clients know how to construct that value. Entering a value here or on the client side interferes with this default process.

iv If the workspace server doesn't require access to network resources (such as UNC pathnames), leave the default value of **Kerberos,NTLM** in the security package list. Or, if you want to ensure that the workspace server can access network resources, force that server to use only Kerberos. See "How to Force Use of Kerberos" on page 172.

Here is an example of IWA settings:



Note: If a user accesses the workspace server seamlessly but the spawner log indicates that credential-based authentication occurred (instead of IWA), the user's context includes credentials for the workspace server's host. This can happen if the user didn't use IWA to get to the metadata server or if the user's DefaultAuth login includes a password. Even when IWA is properly configured, any available credentials are used. △

3 Tell users that they can select the IWA check box when they log on to desktop applications such as SAS Information Map Studio, SAS Data Integration Studio, SAS OLAP Cube Studio, SAS Management Console, and SAS Enterprise Guide. In general, users shouldn't make changes to the advanced settings that are depicted in step 1.

IWA requires agreement between client and server about which security protocol to use when exchanging authentication packets. The following table provides details:

Table 13.6 Integrated Windows Authentication Settings

Server Setting	Associated Requirements
Negotiate security package	□ The client must select the Negotiate security package. To verify client-side settings, click **Advanced** in the connection profile. □ The server must have a security package list. By default, servers have a security package list that offers the Kerberos and NTLM protocols. □ At least one of the protocols in the server's security package list must be offered by the client. □ At least one of the protocols that are offered by both parties must actually be supported by both parties. □ The Kerberos protocol can be used only if the client knows the server's SPN, as explained in the following row.
Kerberos security package	□ The client must select the Kerberos security package. □ Both client and server must actually support the Kerberos protocol. □ The client must know the server's service principal name (SPN). In a standard configuration, this is transparent. SAS servers run as services under the Local System account and automatically register their SPN as **SAS/***machine-name:port*. Clients can construct the default SPN (because they know the format, machine name, and port) so you don't have to explicitly provide the SPN anywhere. If you need to use a custom SPN: **1** In Windows, use the **setspn** command. For example: `setspn -A sas/customValue myServer` This code registers **sas/customValue** as the SPN for all servers that run as services under the Local System account on a machine that is named **myServer**. You must be a Windows domain administrator in order to use the **setspn** command. **2** Update all client-side connection profiles and the logical workspace server definition (if applicable) to include **sas/customValue** in the SPN field.
NTLM security package	□ The client must select the NTLM security package. □ Both client and server must actually support the NTLM protocol.

Note: If your SAS servers use DNS aliases, you must manually register those aliases (as custom SPNs) in order to support Kerberos-based IWA connections. △

How to Force Use of Kerberos

If you choose to use the SAS implementation of Integrated Windows authentication (IWA), and you need to ensure that the Kerberos protocol is always used, complete the following instructions. These instructions assume that you have already completed the steps in "How to Configure Integrated Windows Authentication" on page 169.

Note: You can't use local accounts with this configuration, because local accounts can't use Kerberos. △

1 Specify **-secpackagelist "Kerberos"** in your equivalent of the following locations:

- **SAS\Config\Lev1\SASMeta\MetadataServer\sasv9_usermods.cfg** (for the metadata server)
- **SAS\Config\Lev1\SASApp\OLAPServer\sasv9_usermods.cfg** (if you need to support direct IWA connections to an OLAP server on Windows)
- **SAS\Config\Lev1\ObjectSpawner\ObjectSpawner.bat** (if the object spawner is on Windows). If the spawner runs as a service, complete these steps:

  a From the Windows **Start** menu, select **Programs ▶ SAS ▶ SAS Configuration ▶ <Level> ▶ Object Spawner – Stop**.

  b From the object spawner's configuration directory, type:

  ```
  ObjectSpawner.bat -remove
  ```

  c Add the **-secpackagelist "Kerberos"** setting to the **Set CMD_OPTIONS=** line of ObjectSpawner.bat. Also, make sure that the **-sspi** setting is present.

  d To reinstall the spawner service, type:

  ```
  ObjectSpawner.bat -install
  ```

2 If the workspace server is on Windows, make sure that its metadata definition includes only **Kerberos** in the **Security package list** field. This setting is located in SAS Management Console, on the **Plug-ins** tab under **Server Manager**. The setting is on the **Options** tab of the logical workspace server definition.

Note: If the workspace server accesses network resources (such as UNC pathnames), you must also mark the account under which the spawner runs as **trusted for delegation**. See "Windows Privileges" on page 36. △

3 Restart the metadata server.

In general, it is not necessary to also change the default IWA setting in client-side connection profiles. If a server accepts only Kerberos, then clients with the default setting of Negotiate (and both Kerberos and NTLM in the security package list) use Kerberos. However, in some circumstances, the client's Windows system chooses to initiate communication using NTLM and is unable to comply with the server requirement by switching to Kerberos. For example, if the client and server are on the same machine, the client chooses NTLM. In these circumstances, you must adjust the client-side settings to specify only Kerberos.

Note: In their 4.2 and earlier releases, SAS Enterprise Guide and the SAS Add-In for Microsoft Office don't expose the advanced IWA settings. △

Note: If your SAS servers use DNS aliases, you must manually register those aliases in order to support Kerberos-based IWA connections. See the discussion of custom SPNs in Table 13.6 on page 172. △

How to Store Passwords for the Workspace Server

Note: This is not a universally necessary task. This is one of several solutions to a mixed provider situation. See "Mixed Providers" on page 135. △

1 Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw).

2 On the **Plug-ins** tab, expand **Server Manager**, the application server  (for example, **SASApp**) and the logical server  (for example, **SASApp – Logical Workspace Server**).

(Optional) Right-click the logical server, select Properties, and select the Options tab. Make sure that the **Authentication service** is set to use **Host** with **Username/Password**. Click **OK** to close the dialog box and return to Server Manager.

3 Below the logical server  , select the server  (for example, **SASApp – Workspace Server**).

4 In the display panel, right-click the server's connection object  and select **Properties**.

5 On the **Options** tab, notice the value in the **Authentication domain** drop-down list. If the value is something other than DefaultAuth, proceed to step 6. Otherwise, complete these steps:

   a Next to the **Authentication domain** drop-down list, click **New**.
   b In the New Authentication Domain dialog box, enter a name such as **UNIXAuth** or **WorkspaceAuth** (you can use any name that is meaningful to you). Click **OK**.
   c On the **Options** tab, make sure the new authentication domain is selected in the **Authentication domain** drop-down list. Click **OK**.

6 Create a SAS copy of credentials that are known to the workspace server's host operating system. In most cases, you will store shared credentials on a group's **Accounts**. You can also store a unique set of individual credentials on each user's **Accounts** tab. Each login must be assigned to the workspace server's authentication domain. Each login must include both a user ID and a password.

   For example, for a workspace server on UNIX:

   ```
   UNIXAuth | myID | mypassword
   ```

   For example, for a workspace server on Windows:

   ```
   WINAuth  | Win\myID | myWINpassword
   ```

   If you store credentials for a workspace server that runs on Windows, give users the **Log on as a batch job** privilege. See "Windows Privileges" on page 36.

   Note: If you don't store the passwords, users are prompted for such credentials when they make a request that requires access to the workspace server. Only desktop applications and SAS Web Report Studio provide such secondary logon prompts. △

Note: Do not instead leave the workspace server in DefaultAuth and move inbound logins to some other authentication domain. Failure to follow this recommendation won't affect the initial logon process, but it will interfere with access to the workspace server. By default, all clients insert the credentials that a user submits at the logon prompt into that user's context as a DefaultAuth entry. This cached DefaultAuth entry has priority over any DefaultAuth credentials that are retrieved from logins in the metadata. △

See Also

"Credential Management" on page 144
"How Logins Are Used" on page 139

How to Store Passwords for a Third-Party Server

Note: This is not a universally necessary task. Use these instructions to provide seamless access to a third-party server that uses a proprietary authentication provider

(for example, Oracle). These instructions associate the database logins with a user or group, not directly with a database library. △

1 Verify that the third-party server is registered in the metadata and is in its own authentication domain.

   a Select the third-party server's definition under **Server Manager** on the **Plug-ins** tab in SAS Management Console.

   b In the display panel, right-click the server's connection object  and select **Properties**. The server's authentication domain assignment is on the **Options** tab.

2 In the server's authentication provider, identify or create accounts. Use any of the following approaches (here, Oracle is used as an example):

   - Create an individual Oracle account for each user. This provides the greatest accountability, but can also necessitate storing many Oracle user IDs and passwords in the metadata.
   - Create one Oracle account that all users will share. This greatly reduces the need to store Oracle user IDs and passwords, but also results in a loss of individual accountability.
   - Create a few Oracle accounts, each of which will be shared by several users. This middle-of-the-road approach enables you to make some access distinctions in Oracle and store only a few Oracle user IDs and passwords in the metadata.

3 In the metadata, store the user IDs and passwords for each account as follows (here, Oracle is used as an example):

   - If you created individual accounts on the Oracle server, add an Oracle login to each user definition.
   - If you created one shared account on the Oracle server, identify or create a group that contains the users who will access the Oracle server. Give that group a login that includes the user ID and password for the Oracle shared account.

     Note: If you want to provide access for all registered users, give the login to the SASUSERS group. △

     Note: If you want to provide access for all users (including users who do not have an individual SAS identity), give the login to the PUBLIC group. △

   - If you created several shared accounts on the Oracle server, identify or create a user group in the metadata for each shared account. Give each group a login for the Oracle server, and assign each user who connects to Oracle to one of the groups.

   Assign these logins to the third-party server's authentication domain. Store both an ID and a password in each login.

   For example, for an Oracle server:

   ```
   OracleAuth | myORAid | myORApassword
   ```

   Note: If you don't store the passwords, users of desktop applications are prompted for such credentials when they make a request that requires access to the server. SAS Web Report Studio has an interactive password management feature. Other Web applications don't support interactive logons to secondary servers. △

See Also

"Credential Management" on page 144

"How Logins Are Used" on page 139

How to Reduce Exposure of the SASTRUST Password

Note: This is not a universally necessary task. This is an optional configuration that can reduce the exposure of a privileged service account. Consider using these instructions if the standard protections (host protection of configuration files and encryption of the sastrust password in those files) is not sufficient for your environment. On hosts other than Windows, these instructions might not provide a clear security benefit. △

In the standard configuration, the OLAP server and the object spawner read the sastrust ID and password from their configuration files during initialization. These components use the sastrust credentials to connect to the metadata server and read the metadata that they use to perform their tasks. For example:

- The spawner uses the SAS Trusted User identity to read metadata about the servers that it launches.
- The OLAP server uses the SAS Trusted User to discover schemas and cubes and to impersonate each requesting user.

The primary reason to remove the sastrust credential from configuration files is to enhance security in a Windows environment. For sites that don't use Integrated Windows authentication, removing the sastrust credentials makes sense only if you prefer the exposure of using the trusted peer protocol to the exposure of storing the credentials in two host-protected files.

Note: Another reason might be to avoid having to update the sastrust password in the configuration file. However, maintenance isn't usually a concern because sastrust is usually an internal account with a password that doesn't expire. △

To remove the sastrust credentials from configuration files:

1. Locate the sasv9_meta.cfg file that is in your equivalent of **SAS/Config/Lev1/**. Make a backup copy of the file. In the original file, delete your version of these lines:

```
-metauser "sastrust@saspw"
-metapass "{sas002}3CD4EA1E35CA49324AOC4D63"
```

2. Locate the file metadataConfig.xml file that is in your equivalent of **SAS/Config/Lev1/ObjectSpawner**. Make a backup copy of the file. In the original file, delete your version of this entry:

```
<Login Name="Metadata Login" UserId="sastrust@saspw"
 Password="{sas002}3CD4EA1E35CA49324AOC4D63"/>
```

3. Determine which account (or accounts) are used for the connections and, on Windows, what format you should use for the account ID in the following steps. For example:
   - On UNIX, the spawner and the OLAP server run under a designated account that is identified during installation.
   - On Windows, if these components run under the local system account, use the format system@*machine-name* (if you are using trusted peer) or *machine-name*$@*Windows-domain* (if you are using IWA). Otherwise, use the usual Windows qualified format (such as *user-ID*@*Windows-domain*).
4. To enable the spawner and the OLAP server to read the same metadata that sastrust reads, add a login such as this to the SAS Trusted User's **Accounts** tab in SAS Management Console:

```
DefaultAuth | connecting-account-ID | (no password)
```

If the components use different accounts, add one login for each account. For example:

```
DefaultAuth | myMachine$@myWinDomain | (no password)
DefaultAuth | invoker                | (no password)
```

You can ignore any messages about multiple logins within an authentication domain.

5 To make the connecting accounts trusted users, add their account IDs to your equivalent of the **SAS/Config/Lev1/SASMeta/MetadataServer/trustedUsers.txt** file. Use the same format for the user ID that you used in step 3. For example, the updated contents might look like this:

```
sastrust@saspw
myMachine$@myWinDomain
invoker
```

Note: The OLAP server uses its trusted user status to impersonate clients for metadata access control evaluations. The spawner uses its trusted user status to determine which servers each requesting user is permitted to use. △

6 Restart the metadata server. Verify that the spawner and the OLAP server can still contact the metadata server. In the metadata server log, you should see a separate connection from each component. The connections will be either IWA (Integrated Windows authentication) or trusted peer.

Note: In most configurations, IWA happens if the metadata server and the connecting component are on Windows and trusted peer happens on other hosts. If you expect IWA but instead see trusted peer, make sure the metadata server has -sspi in its invocation command. If you expect trusted peer but the connection fails, make sure the metadata server has the trustsaspeer object server parameter specified. △

See Also

"Integrated Windows Authentication" on page 148

"Trusted Peer Connections" on page 154

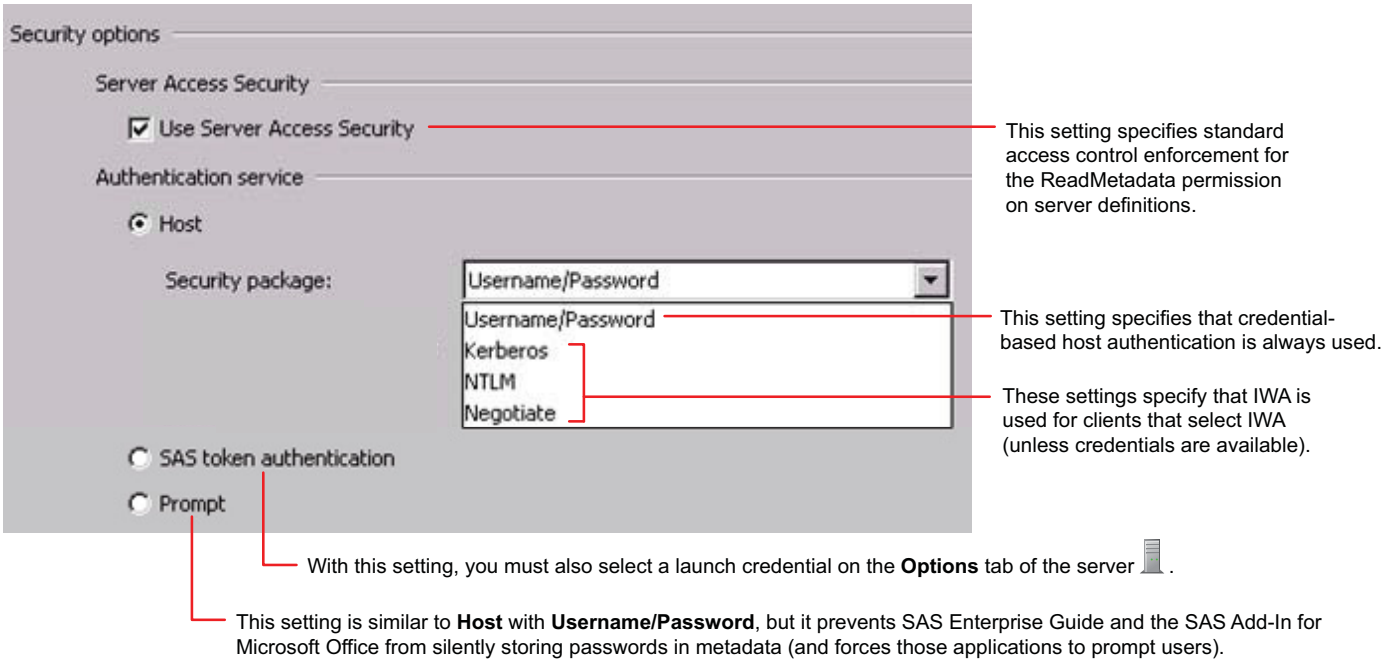
"Trusted User Connections" on page 155

"Passwords That Are Managed By the SAS Deployment Manager" on page 215

About the Workspace Server's Options Tab

The following figure depicts the **Options** tab for a logical workspace server  .

Figure 13.1 The Options Tab for a Logical Workspace Server



Here are some additional details about this tab:

- Because these settings are in the metadata, they can affect only metadata-aware connections.
- The **`Host`** with **`Username/Password`** setting can cause SAS Enterprise Guide and the SAS Add-In for Microsoft Office to silently store user passwords in metadata. The storage occurs only if the workspace server is in its own authentication domain (for example, ServerAuth) and users interactively provide credentials when they access that server. If both of these circumstances apply to your deployment, consider selecting **`Prompt`** instead of **`Host`** with **`Username/Password`**.
- The **`Host`** with **`Username/Password`** setting doesn't eliminate prompting for credentials. With this setting, all desktop applications prompt users to interactively supply credentials in any circumstance where credentials are needed and are not otherwise available.
- The **`Prompt`** setting is the same as the **`Host`** with **`Username/Password`** setting, except that it has the following additional effects in SAS Enterprise Guide and the SAS Add-In for Microsoft Office:
  - prevents silent storage of user passwords in metadata
  - increases interactive prompts for user credentials, preventing single sign-on from those applications to the workspace server
- If a workspace server is converted to client-side pooling, this tab is disabled and might not accurately depict settings that apply if the server is accessed as a standard workspace server. To avoid confusion, don't use the same workspace server for both client-side pooling and standard use. Instead, if you choose to set up client-side pooling, use a dedicated workspace server (inside a separate application server).
- Configuring a workspace server to use SAS token authentication can be useful in a multi-host environment. See "Mixed Providers" on page 135.
- Don't turn off enforcement of the requirement for ReadMetadata permission. This setting is provided only in case a site needs it for backwards compatibility.

- Choosing one of the Integrated Windows authentication (IWA) packages doesn't guarantee that IWA will always be used and doesn't turn off credential-based host authentication. For example:
  - Users who don't choose IWA in their client-side connection profiles don't use IWA. The client-side profile setting affects whether the initial connection to the metadata server uses IWA. In a session in which that initial connection doesn't use IWA, IWA can't be used to launch a workspace server.
  - If a user's stored login includes a password for the workspace server's authentication domain, those credentials are added to the user's context and credential-based host authentication occurs, even if IWA is chosen by both server and client.

    Note: This information is provided to assist in troubleshooting when you expect IWA to occur but it does not. In most cases, a user's login doesn't have to include a password. When you view a user's logins (on the **`Accounts`** tab of his or her user definition), the password field always displays eight asterisks, regardless of whether a password is stored. △
- Changes that you make on this tab take effect after you refresh the object spawner.

See Also

"Integrated Windows Authentication" on page 148

"SAS Token Authentication" on page 153

CHAPTER

14

Server Configuration, Data Retrieval, and Risk

About This Chapter

This chapter explains how SAS servers retrieve SAS data for requesting users. For higher security environments, especially sites that need to provide secure access to SAS tables (data sets), this is essential information.

The central point of the chapter is that different servers retrieve SAS files from the operating system under different host identities. If you understand the underlying factors and relationships, you can make informed choices about data retrieval and risk. The discussion is organized as follows:

- "Identity Passing" on page 182 introduces the concept of preservation of user identity and outlines the behavior of each type of server.
- "Launch Credentials" on page 183 explains how a server's host identity is specified and how you can assign a different account to a server.
- "Host Access to SAS Tables" on page 187 explains the risk of direct access that bypasses metadata controls and the concept of mediated access.
- "Choices in Workspace Server Pooling" on page 192 describes benefits and risks of server configurations that are used when relational information maps are processed.

Identity Passing

About Identity Passing

When a request is passed from one system to another, it is often preferable that the requesting user identity is passed along with the request. This provides individualized access control evaluations and individual accountability in the receiving system. Here are a few examples:

- When a stored process server is launched, it has to use some SAS identity to contact the metadata server in order to discover any pre-assigned libraries.
- When a user opens a SAS folder that contains cubes, the OLAP server has to use some SAS identity to contact the metadata server to determine which cubes to show to that user.
- When a workspace server fetches SAS data from the operating system, it has to use some host identity to authenticate to the operating system and request the SAS data set files.

In each example, the identity that is used affects the outcome (for example, which libraries are available, which cubes are visible, or which data files are returned).

How SAS Servers Preserve Identity

The following table provides details about how SAS servers preserve identity.

Table 14.1 Preservation of User Identity

Server	SAS Identity for Metadata Layer Evaluations	Host Identity for Host Layer Evaluations
OLAP server	The SAS Trusted User. However, user identity is preserved because this privileged service identity impersonates each requesting user.[1]	The server's host identity. This affects drilling to detail if the underlying data is in SAS tables.
Stored process server	Each requesting user's SAS identity. The exception is for library pre-assignment, which happens under the server identity.[2]	The server's host identity (the multi-user credential).
Workspace server (using any form of host authentication)	The SAS identity of the server. This is the same as the SAS identity of the requesting user or group.	The server's host identity (the host identity of the requesting user or group).[3]
Workspace server (using SAS token authentication)	Each requesting user's SAS identity.	The server's host identity (the launch credential).

Server	SAS Identity for Metadata Layer Evaluations	Host Identity for Host Layer Evaluations
Pooled workspace server	Each requesting user's SAS identity.	The server's host identity (the launch credential).
Client-pooled workspace server	The SAS identity of the server. However, SAS Web Report Studio pre-processes requests based on each requesting user's SAS identity.	The server's host identity (the puddle login).

1 Except for retrieval of DBMS credentials, which happens under the SAS Trusted User's identity. This affects drilling to detail if the underlying data is in a third-party DBMS.
2 The requesting user isn't known at the time that library pre-assignment happens (during server initialization).
3 Each server process instance is launched on-demand and runs under the host identity of the requesting user (or group, if authentication is via stored group logins).

Note: You can directly specify which identity to use on a connection to the metadata server (for example, with METAUSER and METAPASS). You can directly specify which identity to use on a connection to a host (for example, by providing explicit credentials in your code). △

In the preceding table, notice that although metadata layer evaluations are almost always individualized (for example, *Does Joe have metadata layer permission to see this stored process?*), physical layer evaluations are often generalized (for example, *Does the sassrv account have host layer permissions for this data?*).

In particular, requests from metadata-aware clients for SAS data often aren't evaluated under the host identity of each requesting user. The host doesn't know the user's SAS identity, so the host can't evaluate the user's operating system permissions. Instead, the host checks the permissions of the service account that is fetching the SAS data on behalf of the requesting user. These service accounts are referred to as launch credentials.

Launch Credentials

About Launch Credentials

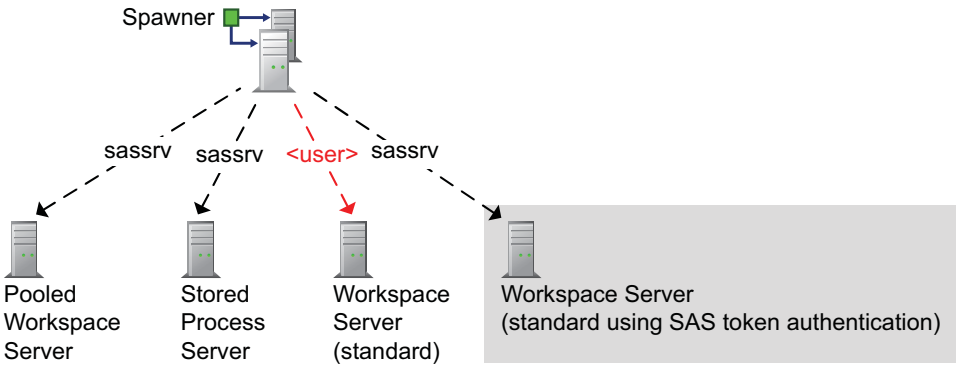
Workspace servers and stored process servers use different process instances (SAS sessions) for different requests. Each process instance is launched by a spawner under a host account as follows:

- For the stored process server, the pooled workspace server, and a standard workspace server that uses SAS token authentication, the spawner launches process instances under a designated service account. That account is called a launch credential (or, sometimes, a multi-user credential) and is specified as part of the server's metadata definition. For example, in the initial configuration, the SAS Spawned Servers account (sassrv) is the only designated launch credential.
- For a standard workspace server that uses host authentication, the spawner launches a new process instance on-demand for each requesting user. The server's metadata definition doesn't include a designated launch credential. Instead, each requesting user's personal host account is used to launch that user's process instances. Or, if the server is accessed using group logins, a group account might

be used. As a result, only users who can authenticate to the workspace server's host can get a workspace server.

The following figure depicts a deployment where the sassrv account is the launch credential for all servers that run under a designated account. The last server in the figure is shaded because it depicts a specialized configuration and can't coexist within an application server that includes another standard workspace server.

Figure 14.1 Example: Launch Credentials for Processing Servers



Note: If you use client-side pooling, the puddle login is comparable to the launch credential. △

Criteria for a Designated Launch Credential

The **`Launch credentials`** (or **`Multiuser credentials`**) setting is available on the **`Options`** tab for a standard workspace server that uses SAS token authentication, the stored process server, and the pooled workspace server.

Display 14.1 Choosing a Launch Credential

Note: In the preceding display, the values within parentheses indicate the authentication domain of the credential. These values aren't part of the service account ID. For example, the selected account ID is **`WIN\sassrv`**, not **`WIN\sassrv (DefaultAuth)`**. △

You should carefully select each server's launch credential for these reasons:

- Not all of the choices are viable without additional configuration.
- Each server retrieves SAS data from the operating system under the host identity of its designated launch credential. Anyone who can use a particular server can potentially access all of the data that is available to that server's launch credential.

The following list explains the choices in the launch credential list:

`(None)`

prevents proper functioning of the server. If the **Multiuser credentials** or **Launch credentials** drop-down list is present and enabled on the **Options** tab for a server  , that server is not functional when **(None)** is selected.

Use spawner identity
can introduce the following complications:

- On Windows, using the spawner's host identity as a launch credential causes launched processes to run with restricted access. For example, even if the spawner runs under the local system account, the host identity of the launched process isn't a member of Windows groups such as Administrators and Power Users. The downgrade is necessary in order to avoid security exposures. The downgrade can interfere with legitimate operation of the server.

  Note: In the initial configuration, a spawner on Windows runs as a service under the local system account. You can reconfigure the spawner to run under some other account. △

- In the standard configuration, the spawner's SAS identity (PUBLIC) can't launch a pooled workspace server or a stored process server. As a PUBLIC-only identity, the spawner lacks the necessary grant of the ReadMetadata permission on the server definition.

  Note: One solution is to create an individual SAS identity for the spawner account. For example, create a user definition named SPAWNER and store the user ID of the spawner account as a login on the **Accounts** tab of that user definition. In the initial configuration, all registered users (SASUSERS) have ReadMetadata permission for server definitions. If you narrow access, you might have to add a grant of ReadMetadata permission on the server's **Authorization** tab for the SPAWNER user. △

Note: Ensure that the server's launch credential has any necessary host access to the SAS Web Report Studio query cache library (wrstemp) and distribution library (wrsdist). See the *SAS Intelligence Platform: Web Application Administration Guide*. △

A listed login (for example, **WIN\sassrv**) or a login that you access by selecting **More logins**
is the standard choice. In order to successfully function as a launch credential, a login must meet all of the following criteria:

- The login that must be visible to the spawner. All of the logins on the **Accounts** tab of the SAS General Servers group are visible to the spawner.

  Note: The SAS Trusted User is a member of the SAS General Servers group. The spawner uses the SAS Trusted User to gather the metadata information needed to launch servers. △

- The login must include the user ID and password for an account that is known to server's host. It doesn't matter which authentication domain the login is in.

  Note: If the server is on Windows, the user ID in the login must be in a qualified format (for example, WIN\serviceID). △

- The login must be owned by a SAS identity that has the ReadMetadata permission for the server definition. In the standard configuration, this requirement is met because the SAS General Servers group, like all other registered identities, has ReadMetadata permission for all server definitions.

However, if you choose to limit ReadMetadata permission on a server definition, you must preserve access for the SAS General Servers group.

- The login must reference an account that has host layer access to any SAS data that it retrieves.
- If the server is on Windows, the login must reference an account that has the **Log on as a batch job** privilege.

Note: The login should reference a service account. This login shouldn't correspond to a real person. △

Note: Ensure that the server's launch credential has any necessary host access to the SAS Web Report Studio query cache and distribution libraries. See the *SAS Intelligence Platform: Web Application Administration Guide*. △

The preceding discussion assumes that your deployment uses the standard predefined metadata groups and users. You can choose to configure variations (for example, create a group other than the SAS General Servers group to hold logins for launch credentials). In general, such variations aren't recommended because they unnecessarily increase complexity and reduce consistency.

How to Create and Designate a New Launch Credential

1. Identify or create the host account.
   - If the server will retrieve SAS data, make sure the host account has appropriate access to those files.
   - If the server is on Windows, assign the **Log on as a batch job** privilege to the account.
2. Log on to SAS Management Console as someone who has user administration capabilities (for example, sasadm@saspw). Other users can't add logins to a group definition or see logins that they don't own.
3. In **User Manager**, create a login on the **Accounts** tab of the SAS General Servers group. This login must include both a user ID and a password.

   Note: If the server is on Windows, the user ID in the login must be in a qualified format (for example, WIN\serviceID). △

   Note: You can ignore any warnings about having multiple logins in the same authentication domain. These logins are directly associated with servers. These logins aren't looked up by authentication domain. △

   Note: You can't reuse a login that already exists on the **Accounts** tab for some other user or group. The metadata server's integrity constraints prevent you from assigning the same user ID to two different SAS identities. △

   Note: You can use the same login as a launch credential for more than one server. △

   Note: Users should not be members of the SAS General Servers group. This group is for service identities only. Users don't retrieve these credentials, so users shouldn't have access to these logins. In a standard configuration, these logins are available only to the privileged service identity that the spawner uses to read all server metadata (the SAS Trusted User). △
4. In **Server Manager**, on the server's **Options** tab, select the login as the server's launch credential (or multiuser credential).

   Note: The launch credential is designated at the level of the server component  , not at the level of the logical server  . △

Note: To locate a login that isn't displayed in the drop-down list, select the **More logins** entry. Although all logins are displayed in the secondary dialog box, any login that isn't visible to the SAS Trusted User isn't a viable choice. △

5 To make the changes take effect, re-initialize the spawner.

   a Expand the object spawner  , right-click the computer  , and select **Connect**.
   b Right-click the computer again and select **Refresh Spawner**. In the message box, click **Yes**.
   c Right-click the computer a third time and select **Disconnect**.

6 Validate the server.

Who Can Launch a Standard Workspace Server?

In order to launch a standard workspace server that uses any form of host authentication, a user must have access to an individual or group account that meets all of the following criteria:

- The account must be known to the workspace server's host. An internal account (such as sasadm@saspw) doesn't meet this requirement.
- The account must have host layer access to any SAS data that it retrieves.
- If the server is on Windows, and the user doesn't connect through Integrated Windows authentication, the account must have the **Log on as a batch job** privilege.
- If the user supplies credentials interactively (for example, at a secondary logon prompt), the account must correspond to a SAS identity that has the ReadMetadata permission for the workspace server definition. In the initial configuration, all registered users meet this requirement, but a PUBLIC-only identity (someone who doesn't have a well-formed user definition) doesn't meet this requirement.

  Note: If the **Use Server Access Security** check box on the **Options** tab of the logical server is cleared, this requirement doesn't apply. △

Host Access to SAS Tables

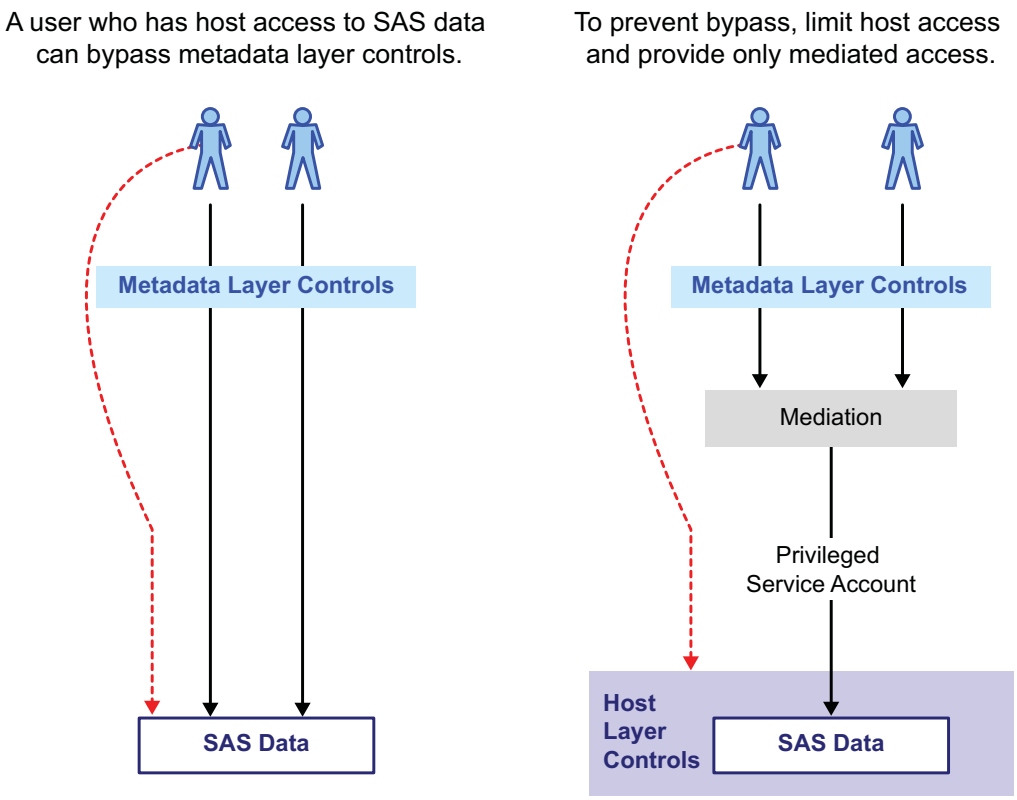
Direct Access Versus Mediated Access

A SAS table (data set) is stored as a file in a host operating system. In general, a SAS table can be directly accessed and read by anyone who has the necessary host operating system permissions to that file. You can use the SAS metadata authorization layer to further constrain access. However, those additional constraints apply only when the user requests that data in a metadata-aware context.

As a simple example, assume that you have registered a library and several tables in the metadata. In SAS Management Console, you deny userA the ReadMetadata permission for a table named Salary. This prevents userA from seeing the Salary table in metadata-aware applications. However, if userA has host access to the physical file that contains the table data, userA can open that file in Base SAS and examine all of the data. When the table is accessed directly, the metadata layer denial of

ReadMetadata permission isn't applied. In the following figure, the first image illustrates the issue and introduces the concept of mediated access.

Figure 14.2 Host Access to SAS Data



With mediated access, a workspace server or stored process server retrieves SAS data from the host operating system using a service account called a launch credential. The server uses one account to retrieve data on behalf of each requesting user. Mediation is an integral part of a standard configuration as follows:

- Host access from the stored process server and the pooled workspace server is always mediated.
- Host access from the standard workspace server is mediated if the server is configured to use SAS token authentication or if the server is providing client-side pooling.

Mediation offers the following advantages:

- Mediation makes pooling of workspace servers possible. For Web applications, in particular SAS Web Report Studio, some form of pooling (client-side or server-side) is strongly recommended for performance reasons. Both forms of pooling always result in mediated host access to SAS data.
- Mediation facilitates partial access to a SAS table through metadata layer controls (for example, row-level or column-level access). Host layer controls are at the file level. Such controls can grant or deny access to only an entire table.
- Mediation helps you avoid creating end-user accounts on back-end servers that host SAS data.

CAUTION:

Along with its advantages, mediation introduces some risk. If your deployment includes sensitive data in SAS tables, it is essential to review existing host layer controls and to understand the effects of mediated access. △

Managing the Risks of Mediated Host Access

The risks of mediated host access are as follows:

- Someone might obtain the service account ID and password and use those credentials to directly access the data.
- Someone might misuse the server in order to access data in a manner that circumvents metadata layer controls.

The following table describes techniques for reducing risk. Keep in mind that there is no absolute security and that you must balance security goals against other considerations.

Table 14.2 Alternatives for Reducing Mediation Risks

Modification	Risk Reduction	Notes
Use a different launch credential for each server. For example, configure the server-side pooled workspace server with its own dedicated launch credential.	Diversification reduces exposure for each launch credential.	Recommended if you have sensitive data in SAS tables. See "Launch Credentials" on page 183.
Limit the availability of SAS Information Map Studio by installing that application only where it is needed.	Reduces opportunity to misuse that application.	Recommended if you have server-side pooling.
Limit the SAS folder locations from which relational information maps can be used.[1]	Reduces opportunity to use a rogue information map, but affects only SAS Web Report Studio and SAS Web Report Viewer.	See the wrs.map.accessibility.check property in the *SAS Intelligence Platform: Web Application Administration Guide*.
Set up client-side pooling using a protected, dedicated service account as the puddle login.	Reduces opportunity to misuse the server by creating a rogue application. A client-pooled workspace server can be used from only designated Web applications.	See "Choices in Workspace Server Pooling" on page 192.
Delete the server-side pooled workspace server (and, for performance reasons, set up client-side pooling for SAS Web Report Studio).	Eliminates opportunity to misuse that server (in SAS Information Map Studio or in a rogue application).	See "Choices in Workspace Server Pooling" on page 192.
Set up a restricted deployment of SAS Web Report Studio with client-side pooling.	Fully isolates the restricted client-side pool (with a dedicated pool administrator as well as a dedicated puddle login).[1]	See Chapter 8, "BI Row-Level Permissions," on page 93.

1 This measure is necessary in order to make row-level permissions that are defined in information maps fully secure.

Example: Multiple Levels of Host Access

The preceding figure depicts a simple case where there is only one level of host access. You can establish multiple levels of host access by setting up multiple servers, each with its own launch credential. This can involve a trade off of granularity (*how many different levels of access will I establish?*) against administrative effort (*how much server metadata will I create and manage?*).

For example, the following instructions show how you can physically isolate sensitive data that is accessed through a logical workspace server that uses SAS token authentication.

1 In the operating system that hosts the SAS data:

   a Establish a separate file system directory for each level of access. This enables you to avoid setting host permissions on individual files. For example, to separate HR tables, you might create an HR directory.

   b Create or identify one service account for each level of access. For example, to provide physical protection for HR data, you might add an hrsrv account.

   Note: For servers on Windows, each launch account must have the **Log on as a batch job** privilege. See "Windows Privileges" on page 36. △

   c Set operating system permissions so that each account has appropriate access to your SAS data. For example, deny the general use account (sassrv) physical access to the HR data and grant the HR account (hrsrv) physical access to all data (including the HR data).

   Note: If you want certain IT staff to be able to access the data directly using their own accounts, preserve that access. △

2 In SAS Management Console's **User Manager**, select the SAS General Servers group, right-click, and select the **Accounts** tab. For each service account that you created in step 1, click **New** and enter the service account credentials as an additional login for the SAS General Servers group. Include a password in each of these logins. For example, you might enter the HR login as:

```
DefaultAuth | WIN\hrsrv | password
```

Note: Each of these logins will be directly associated with a particular server, instead of being looked up by authentication domain. For this reason, it doesn't matter which authentication domain you use. You can ignore any warnings about the SAS General Servers group having multiple logins in the same authentication domain. △

Note: Users should not be members of the SAS General Servers group. This group is for service identities only. Users don't retrieve these credentials, so users shouldn't have access to these logins. In a standard configuration, these logins are available only to the privileged service identity (the SAS Trusted User) that the spawner uses to read all server metadata. △

3 In **Server Manager**, select the logical workspace server  , right-click, and select **Properties**. On the **Options** tab, verify that the **Use server access security** check box and the **SAS token authentication** radio button are selected. Click **Cancel**.

4 Select the logical workspace server again, right-click, and select **Add Server**. Set up one additional server  for each level of access. All of the servers can use the same port. For example, you might add a server named Human Resources.

5 Beneath the logical workspace server, complete these steps for each server  that you added:

   a Select the server, right-click, select **Properties**, and select the **Options** tab. In the **Launch credentials** drop-down list, select one of the service accounts (use a different account for each server). For example, for the Human Resources server you would first select **More logins** and then search for and select the login that references the hrsrv account.

   Note: In order to see this login in the list, you need user administration capabilities. △

   b On the **Authorization** tab, use the ReadMetadata permission to control who can use that server.

   Note: Be sure to preserve necessary service access. See "Hiding Server Definitions" on page 88. △

6 Under **Server Manager**, select the object spawner  , right-click, and select **Properties**. On the object spawner's **Servers** tab, add the new servers to the **Selected Servers** list.

7 To make the changes take effect:

   a Expand the object spawner  , right-click the computer  , and select **Connect**.
   b Right-click the computer again and select **Refresh Spawner**. In the message box, click **Yes**.
   c Right-click the computer a third time and select **Disconnect**.

8 To validate, expand the logical workspace server  and select a server  . On each connection object  in the display area, right-click and select **Test Connection**. Test the connections for every other sibling server  that you added under the logical workspace server.

If you want to also enable power users to access data using their own accounts, set up a separate application server  for those users. In that application server, add a logical workspace server  that uses some form of host authentication. Even though regular users wouldn't be able to authenticate (because they wouldn't have host accounts for the workspace server machine), it is a good practice to limit ReadMetadata access to the additional application server.

You can use a similar approach to set up multiple levels of host access for a stored process server or a server-pooled workspace server. For multiple levels of host access from a client-pooled workspace server, set up multiple puddles. Always ensure that the server's host identity meets all of the requirements that are documented in "Launch Credentials" on page 183.

Choices in Workspace Server Pooling

About Workspace Server Pooling

The primary purpose of pooling is to enhance performance by avoiding the time penalty that is associated with launching all workspace servers on demand. In pooling, a set of workspace servers are made available to process certain types of query requests. In general, pooling is used when a relational information map is queried, processed, opened, or used indirectly (through a report).

The initial configuration in a new deployment conforms to the following general recommendations:

- Use server-side pooling. The initial configuration includes a logical pooled workspace server.
- Use client-side pooling only if you have security requirements that aren't met by server-side pooling. The initial configuration doesn't include client-side pooling.

Benefits and Risks of Server-Side Pooling

Server-side pooling offers the following benefits:

- With server side-pooling, the spawner can request user-specific metadata layer access evaluations, such as whether a user has permission to use a particular server. This isn't possible in client-side pooling because use of the pool administrator prevents the spawner from knowing who the requesting user is.
- With server-side pooling, you can track each request to a specific server instance. This level of auditing isn't possible in client-side pooling because use of the pool administrator prevents the spawner from knowing who the requesting user is.
- With server-side pooling, testing of an information map in SAS Information Map Studio more closely approximates results in SAS Web Report Studio, because both requests can use the same server.
- With server-side pooling, allocation of server processes is managed across clients, using the spawner's load-balancing capabilities. Client-side pooling uses round-robin assignments on a per-application basis.
- A side effect of pooling is that the launched workspace servers run under a designated service account. This side-effect is beneficial in avoiding credential gaps that can occur when a desktop application requests a workspace server from a middle-tier service. This benefit is not provided by client-side pooling, because client-side pooling is supported only for Web applications.

Server-side pooling introduces the following risks:

- Someone might write a rogue application that bypasses metadata layer access controls. This risk originates from the following difference in server control:
  - In client-side pooling, the server is controlled by a designated Web application. Only that application can request a pooled workspace server, because only that application can provide the pool administrator's credentials. No other application can use (or exploit) the server.
  - In server-side pooling, there is no pool administrator or alternate form of application-based gatekeeping. Anyone who can use the server legitimately (from a SAS application) can also exploit that server by writing a rogue

application that uses the server to fetch SAS tables. The security issue is that retrieval from such an application bypasses metadata-layer controls that apply when the same data is accessed in a legitimate manner. For example, a request in SAS Web Report Studio might be filtered by a metadata layer permission condition that enables the user to see only certain rows in a data set.

Note: This exposure is also applicable to other SAS processing servers that provide mediated access (standard workspace servers that use SAS token authentication, stored process servers). This exposure occurs in client-side pooling if a user obtains the pool administrator's credentials. △

Note: To avoid a similar exposure, nobody is allowed to assign a stored process to a server-side pooled workspace server. △

- Someone might exploit the server-side pooled workspace server from within SAS Information Map Studio. In general, only information map creators who have host access to the target data use this application. If someone else has SAS Information Map Studio, that person could bypass metadata layer controls when querying a relational information map from within that application.

  Note: This is never a risk with client-side pooling, because client-side pooling isn't available for desktop applications such as SAS Information Map Studio. △

The following table summarizes the trade-offs. For completeness, the table includes a column for a standard workspace server that uses SAS token authentication and a column for a standard workspace server that uses either form of host authentication (credential-based or Integrated Windows authentication).

Table 14.3 Summary: Comparison of Workspace Server Configurations

Advantage	**Pooled**		**Not Pooled**	
	Server-Side	**Client-Side**	**SAS Token**	**Host**
Performance and Compatibility:				
Improves performance of Web applications.	✓	✓		
Compatible with spawner-based load balancing.	✓		✓	✓
Compatible with Web authentication.	✓	✓	✓	
Accommodates batch generation of scheduled reports.[1]	✓		✓	
Fully compatible with Integrated Windows authentication.[2]	✓		✓	
Security:				
The SAS identity of the requesting user (or group) is used for metadata layer evaluations.	✓		✓	✓
Server access security is supported.	✓		✓	✓
The server is controlled by designated client applications.		✓		

Advantage	Pooled		Not Pooled	
	Server-Side	Client-Side	SAS Token	Host
The user (or group) host identity is passed to the host layer.				✓
The user (or group) host identity is passed to SQL Server.[3]				✓

1 Credentials for the workspace server are available to the batch generation process.
2 Accommodates requests through SAS Intelligent Query Services for a workspace server (for example, opening an information map after logging on to SAS Enterprise Guide using Integrated Windows authentication).
3 If the workspace server's host authentication is by IWA and any additional configuration requirements are met.

Which Requests Are Eligible to Use Pooling?

Only requests that are handled by a particular query services software component are eligible to use pooling. That software component is primarily used to query, process, open, or otherwise interact with a relational information map.

Note: In SAS Enterprise Guide and the SAS Add-In for Microsoft Office, not all such requests are eligible. If the libraries that an information map references can be assigned within an existing SAS session, a request to open that information map is not eligible to use pooling. To ensure that such requests are eligible, limit physical (host operating system) access to the directories that are referenced by that information map's libraries. Deny access to users and grant access to the host identity under which a pooled server runs. △

Note: Similar requests that don't involve a relational information map aren't eligible, because those requests aren't handled by the query services component. For example, requests to open a report that directly contains a stored process or open a report that contains OLAP data are not eligible. △

Note: A few specialized low-level tasks (such as dynamically building a list of prompt values) are eligible, because the query services component is used for those tasks. △

Not all requests that can use pooling actually do so. See the following topic.

Which Eligible Requests Actually Use Pooling?

Use of pooling for eligible requests is constrained as follows:

client-side pooling
: can be used for eligible requests in only specially configured Web applications. For example, if SAS Web Report Studio's configuration includes a pool administrator, that application uses client-side pooling to process information maps.

server-side pooling
: can be used for eligible requests in any application. For example, server-side pooling can be used to process information maps from SAS Web Report Studio, SAS Information Map Studio, SAS Enterprise Guide, and the SAS Add-In for Microsoft Office. However, eligible requests don't use server-side pooling in the following circumstances:

  - If a Web application has a configured pool administrator, requests from that application don't use server-side pooling.

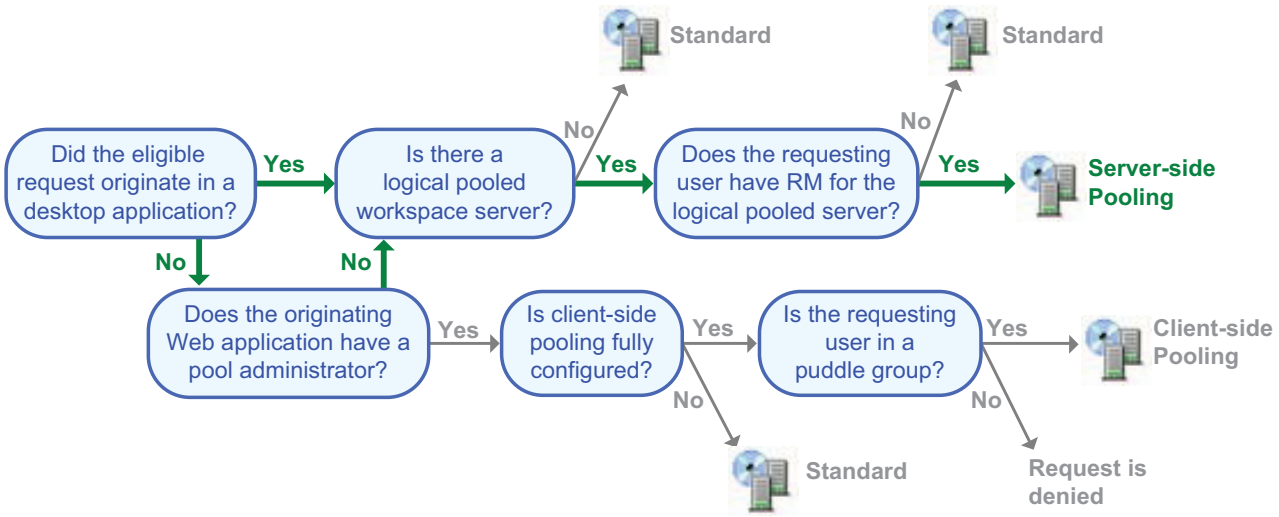
- If there is no logical pooled workspace server under a particular application server , requests for resources that are assigned to that application server can't use server-side pooling.
- If a user doesn't have the ReadMetadata permission for the logical pooled workspace server , requests made by that user can't use server-side pooling.

  Note: If you chose to limit ReadMetadata permission permission for the server, be sure to preserve necessary access for service identities. △

  Note: If the **`Use Server Access Security`** check box on the logical server's **`Options`** tab is cleared, users don't need ReadMetadata permission. This check box should always be selected. △

The following figure summarizes the decision sequence that determines what type of server processes an eligible request. Most of the determinative factors can be controlled by an administrator.

Figure 14.3 Use of Pooling (Applies to Only Eligible Requests)



In the preceding figure, the bold (green) text and arrows mark the decision path in a new deployment with the default configuration. In such a deployment, there are two logical workspace servers within the general-use application server (for example, **`SASapp`**):

- a server-side pooled workspace server (the **`Logical Pooled Workspace Server`**) that is visible to all registered users.
- a standard workspace server (the **`Logical Workspace Server`**) that is not converted to client-side pooling. There is no configured pool administrator, so client-side pooling is not attempted.

Modifying the Initial Pooling Configuration

If you are concerned about the risk that server-side pooling introduces, and that concern outweighs the advantages of server-side pooling, consider these options:

- For a high security implementation of BI row-level permissions, you must use client-side pooling in a separate deployment of SAS Web Report Studio. You can continue to use server-side pooling in the original deployment.
- To prevent only Web applications from using server-side pooling, configure those applications to use client-side pooling.

  Note: In order to get the security benefit, you must set up client-side pooling correctly. For example, don't use the general SAS Spawned Servers credential (sassrv) as the puddle login. △
- To enable information map creators to run queries under their own host identities (instead of under the launch credential of the server-side pooled workspace server), hide that server definition from those users. Someone who can't see the logical pooled workspace server uses the standard workspace server instead.

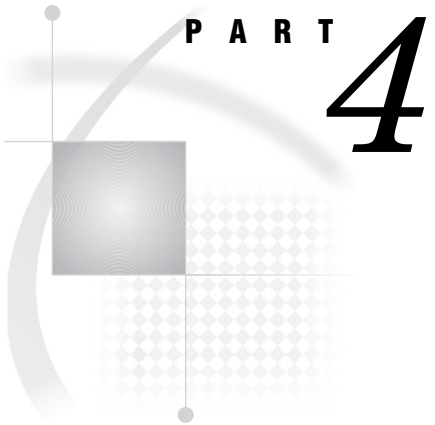
  Note: This is appropriate only if you have an information map creator who shouldn't be able to access all data that is available to the pooled workspace server. This doesn't eliminate the risk of unauthorized use of SAS Information Map Studio by someone who legitimately uses the server-side pooled workspace server from other applications. △
- To completely eliminate use of server-side pooling, delete the logical pooled workspace server. For performance reasons, we recommend that you also configure client-side pooling for SAS Web Report Studio.

See Also

Chapter 8, "BI Row-Level Permissions," on page 93

"Configuring Client-side Pooling" in the *SAS Intelligence Platform: Application Server Administration Guide*

"Hiding Server Definitions" on page 88



PART 4

Encryption

CHAPTER
15

Encryption Model

Encryption Strength and Coverage

Two Classes of Encryption Strength

SAS offers two classes of encryption strength:

- If you don't have SAS/SECURE, you can use only the SASProprietary algorithm. SASProprietary encoding uses 32-bit keys and is appropriate for preventing accidental exposure of information. SASProprietary is licensed with Base SAS software and is available in all deployments.
- If you have SAS/SECURE, you can use an industry-standard encryption algorithm such as AES. SAS/SECURE is an add-on product that is licensed separately and offers maximum protection.

Two Contexts for Encryption Coverage

SAS provides encryption in two contexts:

- On-disk encryption protects data at rest. The emphasis is on protection of passwords, both in configuration files and in the metadata repository.
- Over-the-wire encryption protects data while in transit. The emphasis is on protection of credentials, but you can also choose to protect all traffic in transit.

Default Settings for On-Disk Encryption

On-disk encryption protects data at rest. The following table describes initial support:

Table 15.1 On-Disk Encryption and Encoding

Content and Context	Algorithm	Configuration
Password on disk in the metadata	AES[1]	Controlled by the STOREPASSWORDS option in the metadata server's omaconfig.xml file.
Password on disk in a configuration file	SASProprietary	If you have SAS/SECURE, you can upgrade to AES. See "Password Updates for Service Accounts" on page 14.
Most other metadata on disk	None	Not configurable.
SAS data sets on disk	None	To apply encryption, use the ENCRYPT= data set option.[2]

1 If you don't have SAS/SECURE, SASProprietary is used.
2 The ENCRYPT= data set option uses a proprietary encryption algorithm that is not the same as the SASProprietary algorithm. See the *SAS Language Reference: Dictionary*.

Note: Configuration files and metadata repository data sets should also be host protected. △

Default Settings for Over-the-Wire Encryption

Over-the-wire encryption protects data while in transit. The following table describes initial support:

Table 15.2 Over-the-Wire Encryption and Encoding

Content and Context	Algorithm	Configuration
Password in transit when a user logs on to a SAS desktop application	AES[1]	NETENCRALG option in the metadata server's invocation command.
Password in transit when a user connects directly to the OLAP server (for example, from a SAS data provider)	AES[1]	NETENCRALG option in the server's invocation command.
Password in transit when a user connects to the metadata server from a Base SAS session	AES[1]	NETENCRALG option in the server's invocation command. Can be affected by the METAENCRYPTALG option in the client session.
Password in transit when a client retrieves a stored password from the metadata	SASProprietary	RETURNPASSWORDS option in the metadata server's omaconfig.xml file. If you have SAS/SECURE, you can upgrade to AES. See "How to Increase Encryption Strength for Outbound Passwords in Transit" on page 206.
Password in transit when a client sends a password to the metadata for storage	SASProprietary	Not configurable.

Content and Context	Algorithm	Configuration
Other data in transit to and from SAS servers	None	CEL object server parameter.
Data (including passwords) in transit between a Web browser and a Web application server	None	See the *SAS Intelligence Platform: Web Application Administration Guide*.

1 If you don't have SAS/SECURE, SASProprietary is used.

About SAS/SECURE

What Does SAS/SECURE Provide?

SAS/SECURE makes industry-standard encryption algorithms available for use in the SAS Intelligence Platform as follows:

- SAS/SECURE enables you to provide stronger protection for data in transit than is provided by SASProprietary encoding. This affects communications among SAS servers and between SAS servers and SAS desktop clients. Here are the supported algorithms by host:
  - On UNIX and z/OS, SAS/SECURE supports AES (Advanced Encryption Standard), AES predecessors (DES and TDES), and the RC4 and RC2 algorithms.
  - On Windows, SAS/SECURE supports algorithms that are included in the Microsoft Cryptographic API.
- SAS/SECURE enables you to provide stronger protection for stored passwords than is provided by SASProprietary encoding. This affects both passwords that are stored in the metadata and passwords that are included in configuration files. The only supported industry-standard algorithm for stored passwords is AES (SAS003).

CAUTION:
Passwords that are stored in SAS003 format become unusable and inaccessible if SAS/SECURE is unavailable. If SAS/SECURE is installed, the default format for stored passwords is SAS003. It is important to keep your SAS/SECURE license current. If you choose to discontinue use of SAS/SECURE, you must revert all stored passwords to SAS002 format before uninstalling the software. To revert passwords, set STOREPASSWORDS="SAS002", restart the metadata server, and use SAS Management Console to re-enter passwords in any logins that need to include passwords. △

Note: In the SAS Intelligence Platform, SAS/SECURE provides only encryption features. Other security features (such as metadata authorization, single sign-on, and use of SSL by SAS applications that run in a third-party Web application server) are not related to SAS/SECURE. △

How Are SAS/SECURE Features Surfaced?

SAS/SECURE isn't an interactive software product (like SAS Management Console) or a product that has its own SAS language elements (like SAS/ACCESS). In the SAS Intelligence Platform, SAS/SECURE features are surfaced as follows:

- In server invocation commands, the -netencralg option support values other than SASProprietary only if you have SAS/SECURE.
- In SAS Management Console, server encryption algorithm values other than SASProprietary are supported only if you have SAS/SECURE.

  Note: All algorithms are listed regardless of whether you have SAS/SECURE. Do not select a value other than SASProprietary unless you have licensed SAS/SECURE. Use the same algorithm and level on all servers. △
- In the PWENCODE procedure, the METHOD option supports the SAS003 value (AES) only if you have SAS/SECURE.
- In the RETURNPASSWORDS and STOREPASSWORDS options in the metadata server's omaconfig.xml file, the SAS003 value (AES) is supported only if you have SAS/SECURE.

Licensing and Availability of SAS/SECURE

Licensing and availability for SAS/SECURE is as follows:

- Although SAS/SECURE is automatically included in all deployment plan files that include Base SAS, SAS/SECURE is not licensed as part of Base SAS. SAS/SECURE requires a separate license on each SAS server machine. Client-side licenses are not needed.
- Availability of SAS/SECURE is subject to import and export restrictions. Some countries have import restrictions on products that contain encryption. The U.S. has export restrictions on products that contain encryption.
- SAS/SECURE is not supported on VMS.

See Also:

Chapter 16, "Encryption Tasks," on page 203

Encryption in SAS

CHAPTER
16

Encryption Tasks

How to Change Over-the-Wire Encryption Settings for SAS Servers

Automatic Configuration

When you install the metadata server, you select an encryption level (which traffic content is encrypted) and an encryption algorithm (how that traffic is encrypted). The settings that you select for the metadata server are applied to all SAS servers. SAS clients usually don't specify encryption settings; they simply conform to the requirements of the servers.

CAUTION:
In the SAS Deployment Wizard, all algorithms are listed regardless of whether you have SAS/SECURE. Do not select a value other than SASProprietary unless you have licensed SAS/SECURE on all SAS server machines. △

Instructions for Post-Installation Changes

If you need to change over-the-wire encryption settings after installation is complete, use the following instructions.

1 Update server configuration files as follows:

   a In the operating system that hosts the metadata server, navigate to your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**.

      □ To change the algorithm, add the NETENCRALG setting that you need to the sasv9_usermods.cfg file.

□ To change the encryption level, copy the entire OBJECTSERVERPARMS line from the sasv9.cfg file into the sasv9_usermods.cfg file. Then edit the CEL value in the usermods version of the file.

For example, to encrypt all traffic with AES, add these lines:

```
-netencralg "AES"
-objectserverparms "cel=everything  {other-parameters}"
```

On z/OS, exclude the initial hyphens and add equal signs as follows:

```
netencralg="AES"
objectserverparms="cel=everything  {other-parameters}"
```

Note: Do not specify a NETENCRALG value other than SASProprietary unless you have licensed SAS/SECURE on all SAS server machines. △

b (Optional) If your deployment offers direct connections from clients to the OLAP server, make the same updates to that server's configuration file.

Note: The OLAP server's configuration files are in your equivalent of **SAS/Config/Lev1/SASApp/OLAPServer/**. △

2 Update server metadata definitions as follows:

a In SAS Management Console, under **Server Manager**, select the metadata server's definition  .

Note: To get to the server definition, you must expand the application server node  and the logical server node  . △

b Right-click the first connection object  , and select **Properties**.

c In the Connection dialog box, select the **Options** tab and click **Advanced Options**. Adjust the settings as necessary.

d In the Advanced Options dialog box, select the **Encryption** tab.

Note: All algorithms are listed regardless of whether you have SAS/SECURE. Do not select a value other than SASProprietary unless you have licensed SAS/SECURE on all SAS server machines. △

Repeat the preceding steps for each server that is launched by the object spawner (the stored process server, the workspace server, and the pooled workspace server).

3 Stop, restart, and validate the servers.

Details About NETENCRALG and CEL

On direct connections, encryption is governed by the server's invocation command. Here are details and some examples:

Note: On z/OS, the following syntax examples are slightly different. See step 1a in the preceding topic. △

NETENCRALG (network encryption algorithm)
is a SAS system option. The NETENCRALG setting that is defined for the metadata server during installation is in the metadata server's sasv9.cfg file.

□ If you accept the default encryption settings during installation, the configuration file includes this line:

```
-netencralg "SASProprietary"
```

- If you have licensed SAS/SECURE and selected the AES algorithm during installation, the setting in the metadata server's sasv9.cfg file is as follows:

```
-netencralg "AES"
```

- If a different NETENCRALG setting has been added to the metadata server's sasv9_usermods.cfg file, that setting has priority.
- Other supported values for NETENCRALG are DES, TripleDES, RC4, and RC2. However, if you haven't licensed SAS/SECURE, only SASProprietary is supported.

CEL (client encryption level)
is a parameter in the OBJECTSERVERPARMS SAS system option. The CEL setting that is defined for the metadata server during installation is in the metadata server's sasv9.cfg file.

- If you accept the default encryption settings during installation, the configuration file includes this line:

```
-objectserverparms "cel=credentials  {other-parameters}"
```

- If, during installation, you selected the option to encrypt all traffic, the setting in the metadata server's sasv9.cfg file is as follows:

```
-objectserverparms "cel=everything {other-parameters}"
```

- If a different CEL setting has been added to the metadata server's sasv9_usermods.cfg file, that setting has priority.

It isn't necessary to specify encryption settings in the invocation command for every component for the following reasons:

- Encryption algorithm and level are negotiated between each pair of communicating components. For example, when the OLAP server and object spawner initialize, they contact the metadata server and conform to the metadata server's encryption settings. The same negotiation occurs when a client application contacts a server.
- For spawned servers (the stored process server, the pooled workspace server, and the workspace server), encryption is determined by metadata settings, not by a server invocation command.

How to Increase Encryption Strength for Passwords at Rest

For passwords in configuration files, use the SAS Deployment Manager's password update utility and supply AES-encrypted passwords. See "Password Updates for Service Accounts" on page 14.

For passwords in the metadata, AES encryption is used by default if you have SAS/SECURE (so no configuration activity is necessary in order to maximize protection).

Note: If the metadata server's omaconfig.xml file specifies STOREPASSWORDS="SAS002", passwords in metadata are stored in SASProprietary format (even if you have SAS/SECURE). The metadata server's omaconfig.xml file is located in your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**. △

CAUTION:

Passwords that are stored in SAS003 format become unusable and inaccessible if SAS/SECURE is unavailable. If SAS/SECURE is installed, the default format for stored passwords is SAS003. It is important to keep your SAS/SECURE license current. If you choose to discontinue use of SAS/SECURE, you must revert all stored passwords

to SAS002 format before uninstalling the software. To revert passwords, set STOREPASSWORDS="SAS002", restart the metadata server, and use SAS Management Console to re-enter passwords in any logins that need to include passwords. △

How to Increase Encryption Strength for Outbound Passwords in Transit

About Outbound Passwords and Over-the-Wire Encryption

A password is outbound when a client retrieves the password from the metadata in order to provide seamless access to a server such as Oracle. The password is outbound from the perspective of the metadata server. Connections to third-party servers often use outbound passwords. Most other connections don't use outbound passwords.

In the initial configuration, outbound passwords are transmitted in SAS002 format (SASProprietary encryption). If you have licensed SAS/SECURE, you can choose to increase the encryption strength for outbound passwords to SAS003 (AES encryption).

Upgrade to RETURNPASSWORDS=SAS003

To increase encryption strength for outbound passwords (if you have SAS/SECURE):

1. Edit the metadata server's omaconfig.xml file to change the initial setting, RETURNPASSWORDS="SAS002", to the more secure setting, RETURNPASSWORDS="SAS003". The metadata server's omaconfig.xml file is located in your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**.
2. Restart the metadata server.
3. Verify that server connections continue to function as expected. If you encounter problems, either review the following topics or revert to RETURNPASSWORDS="SAS002".

RETURNPASSWORDS=SAS003 and Compatibility

Almost all connections are compatible with SAS003 passwords, because almost all connections involve a SAS server and SAS servers can decode SAS003 passwords. For example, connections from SAS Information Map Studio to an Oracle server go through a workspace server. The workspace server decodes the outbound Oracle password.

However, a few specialized connections run directly from a Java client or .NET client to a third-party server. These clients can't decode SAS003 passwords. This is a deliberate limitation that reduces security exposures. Of course, a third-party server can't decode SAS003 passwords either. As a result, such connections fail if they attempt to use a password that is in SAS003 format. Here are some specific types of connections that can't use a SAS003 password (the list isn't exhaustive):

- Connections from a Java client or .NET client to an ESRI server.

  Note: The ESRI server uses host authentication. If possible, avoid the use of outbound passwords by locating this server on a machine that recognizes the accounts with which users log on to SAS applications. This facilitates use of cached credentials to access the ESRI server. △

- Connections from a Java client or .NET client to a non-standard WebDAV server. A non-standard WebDAV server is any server other than the SAS Content Server or Xythos.
- Certain connections within the following solutions:
  - SAS Profitability Management
  - SAS Activity-Based Management
  - SAS Merchandise Intelligence
  - SAS Model Manager (in some configurations)

Accommodating Connections That Can't Use SAS003 Passwords

If you have SAS/SECURE but your deployment requires connections that are incompatible with SAS003 passwords, choose either of the following approaches:

- Simply preserve the initial setting of RETURNPASSWORDS="SAS002".

  Note: With the default settings for sites that have SAS/SECURE (STOREPASSWORDS="SAS003" and RETURNPASSWORDS="SAS002"), outbound passwords are stored in SAS003 format and downgraded to SAS002 format before they are transmitted. △

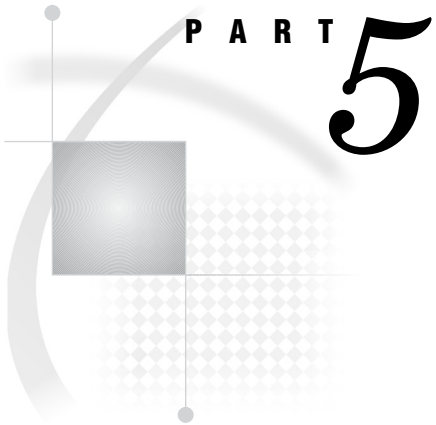
- Set RETURNPASSWORDS="SAS003", but also selectively force the outbound passwords for the problematic connections to be transmitted in SAS002 format. To force a particular password to be transmitted in SAS002 format, store that password in that format. A password that is stored in SAS002 format is transmitted in that format even if RETURNPASSWORDS="SAS003", because the metadata server doesn't upgrade the encryption strength of a stored password when the password is transmitted.

  To use this approach:

  1. In the metadata server's omaconfig.xml file, set STOREPASSWORDS="SAS002". Restart the metadata server.
  2. In SAS Management Console, under User Manager, locate and select the login that contains the outbound password. The login is on the **Accounts** tab of a user or group definition.
  3. Click **Edit**. In the Login Properties dialog box, enter and confirm the password. Click **OK** to close the Login Properties dialog box. Click **OK** again to close the user or group properties dialog box.

     Note: This stores the password in the metadata in SAS002 format. △
  4. Repeat steps 2 and 3 for any other problematic outbound passwords.
  5. In the metadata server's omaconfig.xml file, set STOREPASSWORDS="SAS003" and RETURNPASSWORDS="SAS003". Restart the metadata server.
  6. Verify that server connections continue to function as expected.

  Note: If you update the SAS002 passwords, you must repeat the process so that the new password is stored in SAS002 format. △



PART 5

Appendixes

APPENDIX

1

Checklists

Checklist For a More Secure Deployment

Introduction

You can use this appendix to verify that security issues are being addressed as appropriate for your environment and goals. Not all measures are relevant in all deployments. It is a good practice to review your security configuration on a periodic basis.

General Measures

- Make sure that the configuration directories are protected by appropriate operating system permissions. See the table "Recommended and Default Operating System Protections" in the chapter "What to Do Next: Administration Tasks" in the *SAS Intelligence Platform: System Administration Guide*.
- If your deployment includes sensitive data, review host access to SAS data and evaluate the workspace server pooling configuration. See Chapter 14, "Server Configuration, Data Retrieval, and Risk," on page 181.
- If your deployment includes sensitive data, make sure that the SAS Web Report Studio query cache library (wrstemp) is protected by appropriate operating system permissions. See "Manage Host Access to the Query Cache Directory" in the

chapter "Configuring SAS Web Report Studio" in the *SAS Intelligence Platform: Web Application Administration Guide*.

- If your deployment includes sensitive data, make sure that the SAS Web Report Studio distribution library (wrsdist) is protected by appropriate operating system permissions. See "Verifying Permissions for the Distribution Library" in the chapter "Pre-generated Reports From SAS Web Report Studio" in the *SAS Intelligence Platform: Web Application Administration Guide*.
- On Windows, minimize availability of the privilege **`Log on as batch job`**. See “Windows Privileges” on page 36.
- For industry-standard encryption, license SAS/SECURE. See “About SAS/SECURE” on page 201.
- If your deployment includes the SAS Anonymous Web user, determine whether it is necessary and appropriate to keep that identity in place. See “About PUBLIC Access and Anonymous Access” on page 140.
- Consider tracking changes to server logging levels. See "Audit Server Logging Changes” in the chapter "Administering Logging for SAS Servers” in the *SAS Intelligence Platform: System Administration Guide*.
- Consider limiting the scope of trusted peer connections. See “Trusted Peer Connections” on page 154.

Metadata Layer Measures

- Limit the WriteMetadata permission on servers. See Chapter 7, “Permissions on Servers,” on page 85.
- Limit the WriteMetadata permission on ACTs. In general, only the SAS Administrators group has a grant of the WriteMetadata permission on the **`Authorization`** tab of an ACT.
- Limit the WriteMetadata permission on custom folders. To reduce the chance of inadvertent or deliberate changes to a custom folder, grant WriteMemberMetadata (instead of WriteMetadata) to users who should contribute only content. See “Using WriteMetadata and WriteMemberMetadata Permissions” on page 45.
- Review the WriteMetadata permission on OLAP schemas and libraries. To prevent someone from adding cubes to an OLAP schema or tables to a library, set denials of the WriteMetadata permission on the schema or library. Remember to preserve access for administrators as appropriate.
- Review the permission pattern of each of your ACTs. See “Permission Patterns of Selected ACTs” on page 214.
- Review who has privileged user status from metadata memberships. See “Distribution of Selected Privileges” on page 213.
- Review who has privileged user status from the adminUsers.txt file. See “Distribution of Selected Privileges” on page 213.
- Consider reducing WriteMetadata access to the user definitions for any unrestricted users. This prevents restricted user administrators from updating an unrestricted user’s definition and then logging on as that unrestricted user. To add this protection, access the **`Authorization`** tab of each unrestricted user and add an explicit ☑ denial of the WriteMetadata permission for PUBLIC.
- Consider tracking changes to metadata layer permissions and ACTs. See Chapter 10, “Security Report Macros,” on page 115.
- Consider limiting the locations from which SAS Web Report Studio uses information maps. See "Limiting the Availability of Relational Information Maps"

in the chapter "Managing SAS Web Report Studio Content and Users" in the *SAS Intelligence Platform: Web Application Administration Guide*.

Enhanced Protections for Passwords

- If you have SAS/SECURE, upgrade encryption strength for passwords in configuration files (from SASProprietary to AES). See "Password Updates for Service Accounts" on page 14.
- If you have SAS/SECURE, upgrade encryption strength for outbound passwords in transit (from SASProprietary to AES). See "How to Increase Encryption Strength for Outbound Passwords in Transit" on page 206.
- Consider preventing users from saving their passwords in their desktop connection profiles by setting SASSEC_LOCAL_PW_SAVE="N" (or ="0" or ="F") in the metadata server's omaconfig.xml file.
- Consider strengthening password policies for internal accounts. See "How to Change Internal Account Policies" on page 163.
- Consider removing the SAS Trusted User's password from some configuration files. See "How to Reduce Exposure of the SASTRUST Password" on page 176.

Distribution of Selected Privileges

Table A1.1 Members of Selected Privileged Groups and Roles

Group or Role	Usual Members (As Listed On the Members Tab)
SAS General Servers	SAS Trusted User (no others)
SAS System Services	SAS Trusted User (no others)
SAS Administrators	SAS Administrator and other administrators
Metadata Server: Unrestricted	SAS Administrator and possibly other administrators
Metadata Server: User Administration	SAS Administrators

Table A1.2 User IDs in Configuration Files That Convey Privileged Status

File[1]	Usual Listed User IDs
adminUsers.txt	Only *sasadm@saspw (or one **alternateID*)
trustedUsers.txt	Only sastrust@saspw (or one *alternateID*)

1 These files are located in your equivalent of **`SAS/Config/Lev1/SASMeta/MetadataServer/`**.

Permission Patterns of Selected ACTs

The following tables document the permission patterns of selected ACTs. Each row documents the entire pattern for a particular ACT. Each ACT's pattern consists of only the explicit ☑ settings on that ACT's **`Permission Pattern`** tab. Don't confuse an ACT's **`Permission Pattern`** tab with its **`Authorization`** tab.

The predefined ACTs in the following table are used in the initial configuration and should not be modified.

Table A1.3 Permission Patterns for Selected Predefined ACTs

ACT Name	Denials	Grants		
	PUBLIC	**SAS Administrators**	**SASUSERS**	**SSS**[1]
Default ACT	All permissions	RM, WM, CM, A	RM, WM. CM	RM, WM
Private User Folder ACT	RM, WM, CM	RM, WM, CM	-	RM
SAS Administrators Settings	-	RM, WM, CM, A	-	RM

1 The SAS System Services group, which usually has the SAS Trusted User as its only member.

The example baseline ACTs in the following table are used in Chapter 7, "Permissions on Servers," on page 85 and Chapter 6, "Permissions on Folders," on page 71.

Table A1.4 Permission Patterns for Example Baseline ACTs

ACT Name	Denials	Grants		
	PUBLIC	**SAS Administrators**	**SSS**[1]	**SGS**[2]
HideServer	RM	RM	RM	RM
Hide	RM	RM	RM	-
Protect	WM, WMM, CM, W, A	WM, WMM, CM, W, A, RM	-	-
LimitData	R	-	-	-

1 The SAS System Services group, which usually has the SAS Trusted User as its only member.
2 The SAS General Servers group, which usually has the SAS Trusted User as its only member.

The preceding tables use these abbreviations:

A	the Administer permission
CM	the CheckInMetadata permission
R	the Read permission
RM	the ReadMetadata permission
W	the Write permission
WM	the WriteMetadata permission
WMM	the WriteMemberMetadata permission

Passwords That Are Managed By the SAS Deployment Manager

About This Topic

This topic is provided for reference and to assist with troubleshooting. This topic does not provide comprehensive, step-by-step instructions for updating managed passwords. The recommended method for updating the passwords that are documented in this topic is to use the password update tool. See "Password Updates for Service Accounts" on page 14.

CAUTION:
You should directly edit passwords in configuration files only as a last resort. Before you directly edit any configuration file, review the configuration file precedence information in the appendix "Configuration Files" in the *SAS Intelligence Platform: System Administration Guide*. △

This topic lists only the locations in which the SAS Deployment Manager directly updates passwords. Some passwords are propagated when the utility rebuilds and redeploys the Web applications as part of the automated password update process. This topic doesn't include any additional managed passwords that are introduced by SAS solutions (the administrative documentation for each solution documents any managed passwords that are specific to that solution).

Except where otherwise specified, passwords in configuration files should be in a SAS encoded or encrypted format (the password string should begin with {SAS002} or {SAS003}). However, you should not supply encoded or encrypted password values in SAS Management Console.

Managed Passwords in the SAS Metadata Repository

The SAS Deployment Manager updates passwords in the following metadata locations:

- The passwords for the SAS Administrator and the SAS Trusted User are in the user service definition. To access these settings in SAS Management Console:
  1. On the **Plug-ins** tab, select **Foundation Services Manager ▶ Remote Services ▶ Core**. Right-click **User Service** and select **Properties**.
  2. On the **Service Configuration** tab, click the **Configuration** button.
  3. On the **Users** tab, select a user ID and click **Edit**. The Edit User dialog box contains the **Password** and **Confirm Password** fields.
- The password for the SAS Trusted User is in the content mapping. To access this setting in SAS Management Console:
  1. On the **Folders** tab, right-click the **SAS Folders** node and select **Properties**.
  2. On the **Content Mapping** tab, if the **WebDAV location** radio button is selected, the **Password** and **Confirm Password** fields are enabled. In the standard configuration, the SAS Trusted User's credentials are used here.
- The password for the SAS Server Invoker is in the SAS Server Invoker (sassrv) login. To access this setting in SAS Management Console:
  1. On the **Plug-ins** tab, select **User Manager**.

2. In the display area, right-click the **SAS General Servers** group and select **Properties**.
3. On the **Accounts** tab, select the login and click **Edit**. The Login Properties dialog box contains the **Password** and **Confirm Password** fields.

   Note: The SAS General Servers group often has multiple logins. The SAS Deployment Manager can update only those logins that were designated as server launch credentials through the installation processes. Because these passwords are stored only in the metadata, you can choose to maintain them interactively in SAS Management Console (instead of with the SAS Deployment Manager). △

- The password for the SAS Anonymous Web User is in the SAS Anonymous Web User's login (if this identity exists and has an external account). To access the login in SAS Management Console:
  1. On the **Plug-ins** tab, select **User Manager**.
  2. In the display area, right-click the **SAS Anonymous Web User** and select **Properties**. Not all deployments include this identity.
  3. On the **Accounts** tab, if there is an entry in the logins table, click **Edit**. The Login Properties dialog box contains the **Password** and **Confirm Password** fields. Usually this identity has an internal account and no logins.
- If you use a SAS internal account for any of these identities, that account's password is stored as part of that identity's user definition. To access an internal account password in SAS Management Console:
  1. On the **Plug-ins** tab, select **User Manager**.
  2. In the display area, right-click the user who owns the internal account and select **Properties**.
  3. At the bottom of the **Accounts** tab, click **Update**. The Internal Account Properties dialog box contains the **New Password** and **Confirm Password** fields.
- If you are using a standard configuration of scheduling with Platform Suite for SAS (with SAS Web Report Studio), the password for the LSF user is in a login on the LSF Services group. To access this setting in SAS Management Console:
  1. On the **Plug-ins** tab, select **User Manager**.
  2. In the display area, right-click the **LSFServices** group and select **Properties**.
  3. On the **Accounts** tab, select the login that is in the authentication domain name **LSFServicesAuthWRS**, and click **Edit**. The Login Properties dialog box contains the **Password** and **Confirm Password** fields.

  Note: The SAS Deployment Manager can update this login only if it was created through the installation processes. △

Managed Passwords in the SAS Configuration Directories

The SAS Deployment Manager updates passwords in the following SAS configuration files. Not all deployments include all components. In a multi-machine deployment, not all directories exist on all machines. If you complete the instructions in "How to Reduce Exposure of the SASTRUST Password" on page 176, the first two rows in the table are no longer applicable.

Note: The table uses default names, generic casing, and generic path formats. It omits the initial part of the path (your equivalent of **SAS/Config/Lev1/**). △

Table A1.5 Passwords in Configuration Files

Configuration File That Contains a Password	Account: Location of Password
sasv9_meta.cfg	SAS Trusted User: **metapass**
ObjectSpawner/metadataConfig.xml	SAS Trusted User: **Password** in the **Login** entry
ConnectSpawner/metadataConfig.xml	SAS Trusted User: **Password** in the **Login** entry
Web/Applications/RemoteServices/jps.properties	SAS Trusted User: **omr_password**
Web/Common/login.config	SAS Trusted User: **trustedpw**
Web/Common/SASDomain/weblogicLogin.config	SAS Trusted User: **trustedpw**
SASMeta/MetadataServer/metaparms.sas[1]	SAS Administrator: **OMAAPW**

1 This supports functions (such as pause, resume, backup, restore, and status) that are implemented as an execution of a SAS session with the METAOPERATE or METADATA procedure, or the OMABACKUP macro.

Additional Managed Passwords in the Web Environment

The SAS Deployment Manager updates passwords in the following additional locations:

- The SAS Trusted User's password is in the JAAS configuration as follows:
  - For JBoss, the password is in the login-config.xml file (for example, at **\jboss-4.2.0.GA\server\SASServer1\conf\login-config.xml**).
  - For WebLogic, the password is in the weblogicLogin.config file in the SAS configuration directory (for example, at **SAS/BIServer/Lev1/Web/Common/SASDomain/weblogicLogin.config**).
  - For WebSphere, the SAS Deployment Manager rebuilds the following JAAS Login Modules with the updated password:
    - SAS Content Server (SCS): **com.sas.services.security.login.OMILoginModule**
    - Platform Foundation Services (PFS): **com.sas.services.security.login.OMILoginModule**
    - Platform Foundation Services (PFS): **com.sas.services.security.login.TrustedLoginModule**

    To access these settings in the WebSphere administration console, navigate to the modules under **Security ▸ Secure administration, applications and infrastructure ▸ Java Authentication and Authorization Service ▸ Application logins ▸ <PFS | SCS> ▸ JAAS login modules ▸ <module> ▸ custom properties ▸ trustedpw**.
- The SAS Trusted User's password might be in a JDBC data source definition as follows:
  - For JBoss, the password would be in the SharedServices-ds.xml file (for example, at **\jboss-4.2.0.GA\server\SASServer1\deploy\SharedServices-ds.xml**).
  - For WebLogic, you can access the setting in WebLogic administrative console.
  - For WebSphere, you can access the setting in the WebSphere administrative console under **Resources ▸ JDBC ▸ Data sources ▸ SharedServices ▸ JAAS ▸ J2C authentication data ▸ SASAppSrv01Node/SASJAAS**. In some releases of WebSphere, SAS encoded or encrypted password values are not supported in this context.

Note: Not all deployments include a JDBC data source definition. If MySQL is used, the MySQL password is stored instead of the SAS Trusted User's password. △

- The SAS Trusted User's password is in the environment properties **`secure.password`**, **`omr_password`**, and **`server.admin.password`**. If you use the SAS Anonymous Web User, that user's password is in the environment property **`web.anonymous.password`**. Like all of the passwords in this topic, these passwords should be managed through the SAS Deployment Manager. As a last resort, these passwords can also be accessed through a specialized XML metadata interface (under **Foundation Services 9.2 ► PropertySets ► Environment.Properties (PropertySet) ► SetProperties**).

Who's Who in the SAS Metadata

Table A1.6 Main SAS Identities

SAS User, Group, or Role	Description[1]
SAS Administrator	A predefined super user that has unrestricted access in the metadata layer.
SAS Demo User	A predefined first user that can be useful for demonstrations.
SAS Trusted User	A service identity that can act on behalf of other users.
SAS Anonymous Web User	A service identity that can provide anonymous access for a few Web components.[2]
SAS General Servers	A service group that enables its member (the SAS Trusted User) to see server launch credentials.[3]
SAS System Services	A service group that enables its member (the SAS Trusted User) to see servers, cubes, and other objects.
PUBLIC	Everyone who can access the metadata server. A PUBLIC-only user can't use the deployment.
SASUSERS	Everyone who has a well-formed user definition. The subset of PUBLIC that can use the deployment.
SAS Administrators	General metadata administrators. Membership in this group provides broad access to metadata, but doesn't provide unrestricted access.
Metadata Server: Unrestricted	A highly privileged role that provides unrestricted access in the metadata layer.
Metadata Server: User Administration	A role that enables members (SAS Administrators) to manage most users, groups, and roles, but doesn't provide visibility for any plug-in.
Metadata Server: Operation	A role that enables members (SAS Administrators) to perform most server administration activities, but doesn't provide visibility for any plug-in.
Management Console: Advanced	A role that enables members (SAS Administrators) to see all SAS Management Console plug-ins.
Management Console: Content Management	A role that enables members (SASUSERS) to see a subset of SAS Management Console plug-ins.

1 Descriptions reflect the initial configuration in a new deployment. You can choose to redistribute privileges in a more or less granular approach.

2 For only SAS BI Web Services and the SAS Stored Processes application. Not for other Web applications such as SAS Web Report Studio. Anonymous access is an optional feature that is not compatible with Web authentication.

3 Server launch credentials (for example, the SAS Spawned Servers account, **`sassrv`**) are stored in logins on the **`Accounts`** tab of the SAS General Servers group. This facilitates launching of stored process servers and pooled workspace servers.

APPENDIX

2

User Import Macros

Overview of User Import and Synchronization

In order to make access distinctions and track user activity, the metadata server has to know who is making each request. To enable the metadata server to make this determination, each user's metadata definition includes that user's account ID from your authentication provider. The metadata server maintains its own copy of each ID. The metadata server doesn't maintain copies of external passwords for identification purposes.

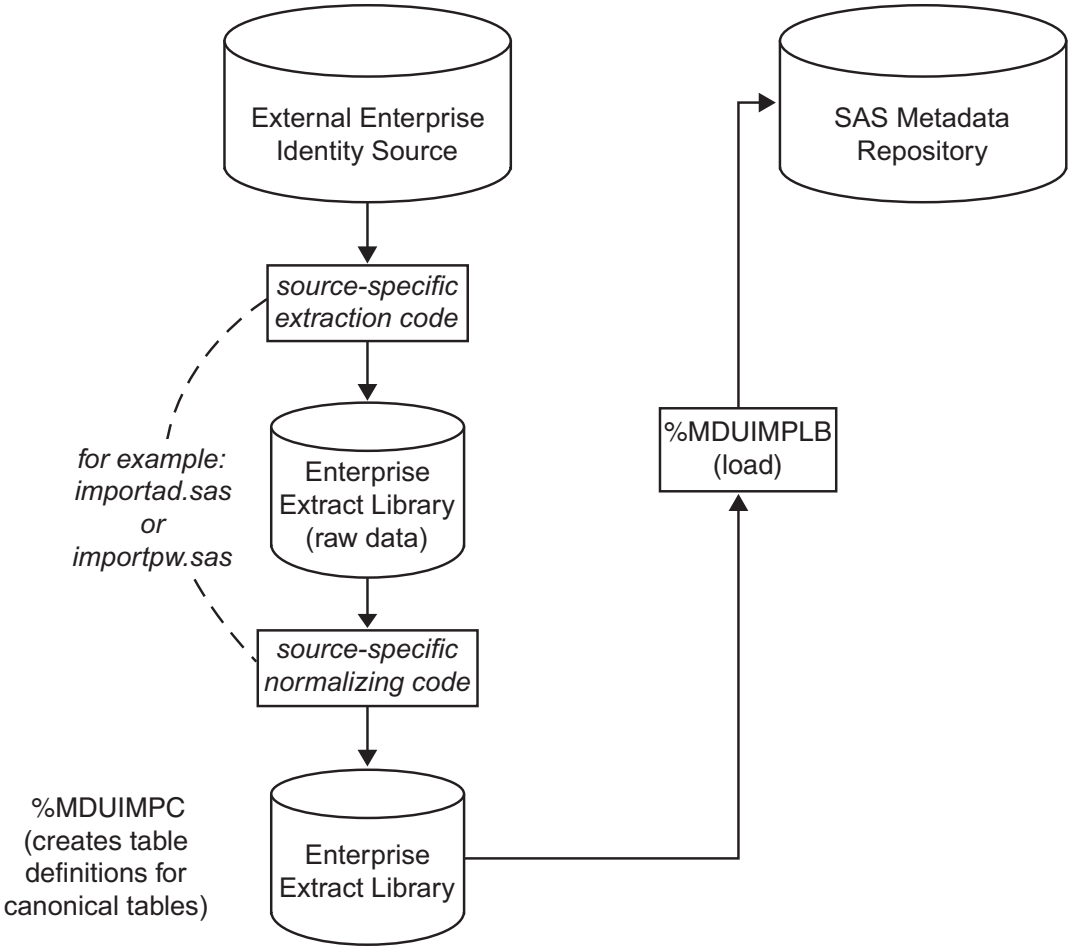
Note: For a few administrators, a SAS internal account can be used instead. See "SAS Internal Authentication" on page 150. △

This chapter helps you use autocall macros and sample code that SAS provides to create your own programs that import and manage user information. The chapter emphasizes coordination with an Active Directory server or UNIX **/etc/passwd** files but also provides information to help you extrapolate to other providers.

The following figures introduce the batch processes for identity information. In the figures, the **MDU****** items are macros and the libraries contain SAS data sets.

The initial import extracts identity information from your authentication provider and loads that information into the metadata.

Figure A2.1 Initial Import

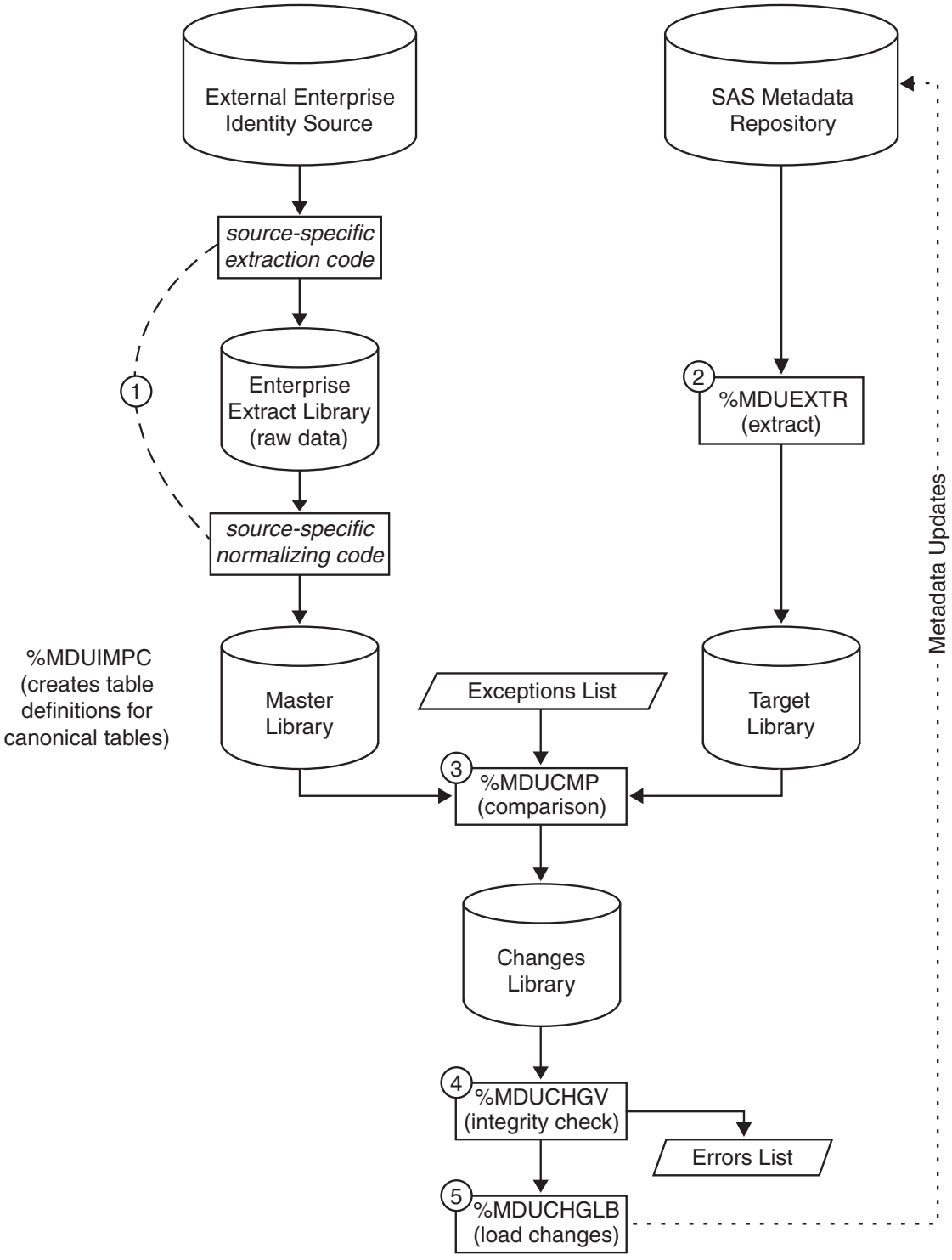


The synchronization performs two extractions (one from your authentication provider and another from the SAS metadata) and then loads validated updates into the metadata. The numbers in the following figure correspond to these activities:

1. Extract information from your authentication provider.
2. Extract information from the SAS metadata.
3. Compare the two sets of tables and identify updates that need to be made to the metadata (excluding any exceptions metadata that you want to preserve).
4. Validate the changes to make sure that they won't violate the metadata server's integrity constraints.
5. Load the updates into the metadata.

Note: Notice that the first part of the import process (the extraction from your authentication provider) is the same as the first part of the synchronization process. You will reuse your import extraction code in your synchronization program. △

Figure A2.2 Periodic Synchronization

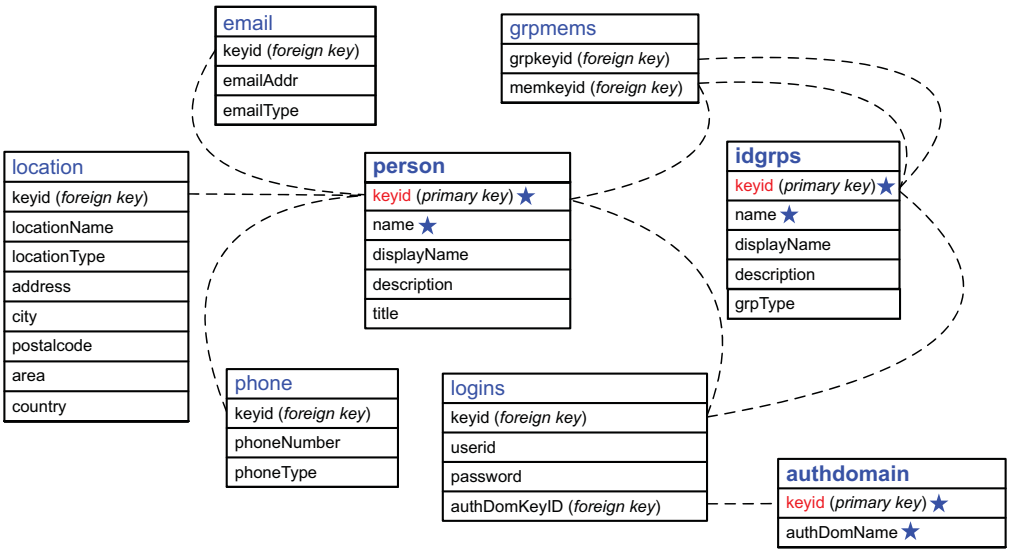


The following two topics document the format of the data sets and explain how corresponding identity entries are mapped between your authentication provider and the SAS metadata.

Canonical Tables

The following figure illustrates the names and structures for the tables in which identity information is stored during batch processing:

Figure A2.3 Table Structures and Relationships



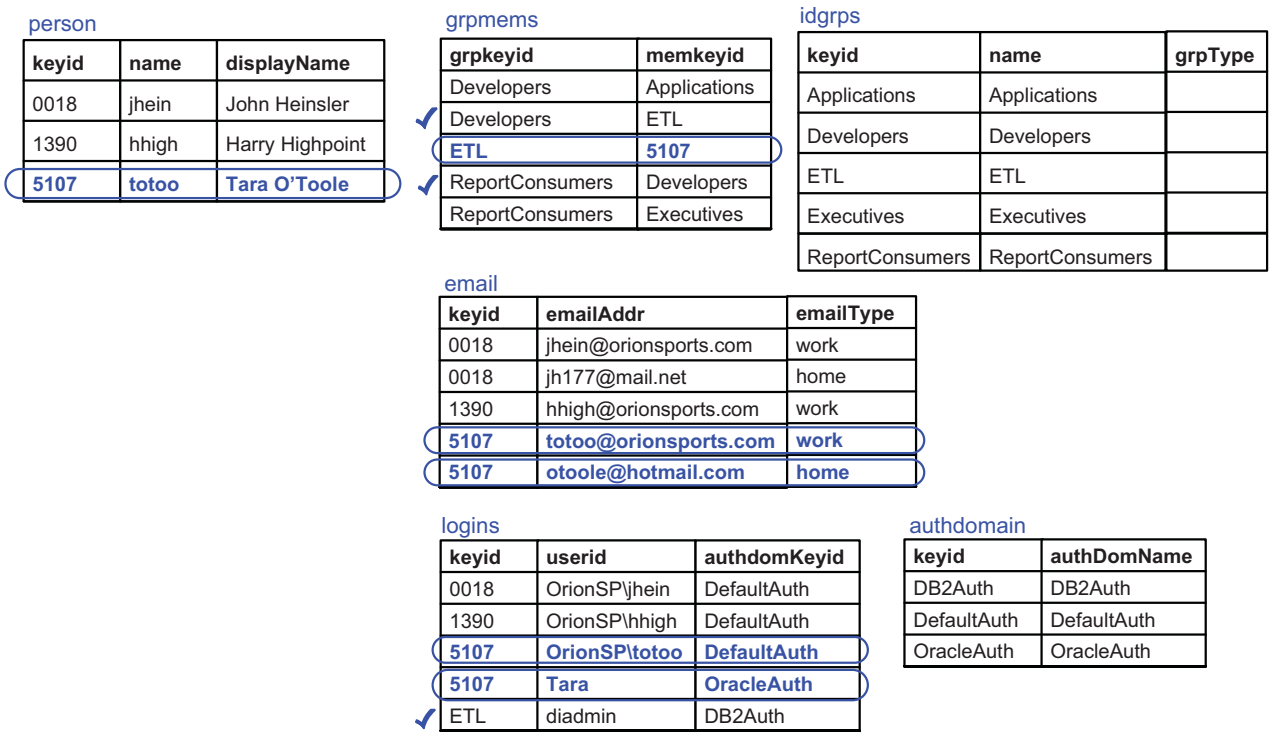
Here are some key points about the tables:

- You don't have to use all of the tables. For example, if you are not going to store e-mail addresses, you don't need an email table.
- For each table that you do use, all columns must be present. However, you don't have to include data in every column. In the figure, the required columns for the primary objects are indicated with a star.

  Note: Each user should have a login that includes a qualified user ID. See "User ID Formats" on page 32. △
- The keyids in the person table (users), idgrps table (groups and roles), and authdomain table (authentication domains) tie each of those primary objects to its related information. In the metadata, the keyid value is stored as an external identity. See "External Identities" on page 226. For each keyid column, use a fixed, enterprise-wide identifier such as these:
  - In the person table, consider using employee identification numbers.
  - In the idgrps table, consider using group names (or LDAP Distinguished Names).
  - In the authdomain table, consider using authentication domain names.
- All of the relationships between a primary object and its related data are zero-or-more relationships. For example, you can store no phone numbers, one phone number, or multiple phone numbers for each user. Constraints are documented in "What Information Is Imported?" on page 227.
- We recommend that you avoid using spaces or special characters in the name of a user, group, or role that you create. Not all components support spaces and special characters in identity names.

The following figure depicts data for a user named Tara O'Toole. The ovals indicate personal data for Tara. The check marks indicate data that is indirectly related to Tara (through her group memberships).

Figure A2.4 Example: Partial Tables Showing Selected User Data



In the figure, notice these things:

- In the person table, the employee number is used as the keyid. The name column uses user IDs (that column also requires unique values). The displayName column contains a common name for each user.
- In the grpmems table (group memberships), Tara is a direct member of the ETL group. Tara is an indirect member of the Developers group, because that group has the ETL group as one of its members. Tara also has a third-level membership in the ReportConsumers group.
- In the idgrps table, the group name serves as the keyid. This choice is appropriate because the metadata server enforces uniqueness across all group and role names.
- In the idgrps table, the grpType column is empty. This indicates that the entries in this table are all groups. To create a role in the metadata, provide a grpType value of **`ROLE`**.
- In the email table, Tara has two e-mail addresses and each one has a different type.
- In the logins table, Tara has personal logins in two different authentication domains. Tara can also use the ETL group's login (for DB2Auth), because Tara is a member of the ETL group.
- In the authdomain table, the authentication domain name serves as the keyid. This is appropriate because the metadata server enforces uniqueness across all authentication domain names.

External Identities

An external identity is a synchronization key that facilitates coordination between identity entries in the metadata and identity entries in your authentication provider. If you use batch processes to coordinate metadata identity information with your authentication provider, external identities are set up and used as follows:

1. In your authentication provider, you select a field to use for the mapping. This should be a field that contains a unique and unchanging value for each user, group, and role that you want to manage with batch processes. Typically, this is an identifier such as employee number.
2. When you perform an initial import from your authentication provider into the metadata, the keyid values in the canonical tables become external identity values in the metadata. Each imported identity has at least one external identity value.
3. During the synchronization process, external identity values that are extracted from the metadata are used as the keyid in the target tables. Because these values also exist in the extraction from your authentication provider, external identity values can be used to match corresponding entries in the two sets of tables.

Note: In SAS Management Console, you can add, view, and manage these external identities from the **`General`** tab of each user, group, or role definition (roles aren't usually imported but are mentioned for completeness). This capability is useful for incorporating manually created identities into a batch synchronization process. See "Scope of the Synchronization Process" on page 228. △

User Import

Scope of the Import Process

Who Participates?

In order to participate in the initial import process, an identity must meet both of these criteria:

- The identity must be included in the import tables. If your identity information is distributed across several authentication providers or user registries, extract information from each source and then combine the resulting sets of tables into one set of canonical tables.

  To limit the import tables, you can perform these tasks:

  - Define a starting point. For example, when you extract identity information from Active Directory, you specify a Distinguished Name as the starting point. Only identities that exist below that Distinguished Name in the Active Directory hierarchy are extracted.
  - Define filters. For example, when you extract identity information from Active Directory, you can use a filter to extract entries only for people who are members of a particular group.
  - Make manual changes to the import tables.

- The identity must not already exist in the SAS environment. You can't import an identity that has the same name as an identity that already exists in the metadata.

CAUTION:
Do not delete existing SAS identities in order to include them in an initial import. When you delete a SAS identity, you lose that identity's associations (such as access controls). Creating a new identity with the same name does not restore those associations. You can incorporate a manually created identity into the *synchronization* process. To do this, add an external identity on the **`General`** tab of that identity's metadata definition. See "External Identities" on page 226. △

What Information Is Imported?

The import process can add this information to the metadata:

- user, group, and role definitions with names, display names, descriptions, and membership information
- job titles, contact information, and personal logins for users

  Note: In most cases, passwords are not added to the metadata because they typically can't be extracted from an authentication provider. If passwords are present in the extracted data, they are loaded into the metadata. It usually isn't necessary to include passwords in logins. △

  Note: Synchronization can process logins for groups. The initial import process does not support these tasks. △

- authentication domains

These constraints apply to the initial import:

- When combined with information that already exists in the metadata, the input data must meet uniqueness requirements. For example, you can't import an identity that has the same name as an identity that already exists in the metadata. See "Unique Names and IDs" on page 33.
- In order to import a user, group, or role, only a name and one external identity value (keyid) is required. However, each user should also have at least one login in order to establish an individual SAS identity. See "Authentication to the Metadata Server" on page 126. Windows user IDs must be qualified. See "User ID Formats" on page 32.

How to Import Identities

Note: It is a good practice to run a backup before you perform an import. △

To import identity information:

1 Locate the sample code that best fits your external identity source.
   - For import from Active Directory, see "About the Sample Code for Active Directory Import" on page 234.
   - For import from UNIX **`/etc/passwd`** files, see "About the Sample Code for UNIX /etc/passwd Import" on page 233.
   - For other formats, the first step is to figure out how to extract the data from your authentication provider. If you have LDAP, you might be able to modify the Active Directory sample for your purposes. Otherwise, use the %MDUIMPC macro to create empty canonical tables, and then use DATA

steps to extract the information and insert it into those tables. See "Sample Code for Generic File Import" on page 230.

2 Decide which attributes you want to add to the metadata. For each attribute, identify a corresponding field in your authentication provider.
3 In the SAS Program Editor, adapt the sample code. The comments in the sample code provide essential details.
4 Submit the code and review the log.
5 In the User Manager plug-in in SAS Management Console, verify that new identities exist. On the **General** tab of an imported user, group, or role, select **External Identities**. You should see an external identity value that matches the identity's keyid in the import tables.
6 Save a copy of your import program for inclusion in your synchronization program.

User Synchronization

Scope of the Synchronization Process

Who Participates?

By default, the synchronization process includes all identities that were originally imported into the metadata (because by default only those identities have an external identity in the metadata). You can modify participation in these ways:

- To include an identity that was manually created, add an external identity on the **General** tab of that user, group, or role definition. See "External Identities" on page 226.
- To exclude an identity that has an external identity in the metadata, define an exception. See "Defining Exceptions" on page 240.

What Information Is Updated?

The synchronization process affects the metadata for participating identities as follows:

- Any identities that have been added to your authentication provider are added.
- Any new logins that have been added to your authentication provider are added. If passwords are extracted from your authentication provider, those passwords are included in the new logins.

  Note: It is typically not possible to extract passwords from an authentication provider. △

  Note: Most logins do not need to include a password. See "How Logins Are Used" on page 139. △

- Any participating identities that are not found in your authentication provider are deleted.
- By default, only logins that meet all of these criteria are deleted:

- The login belongs to a participating identity.
- The login does not exist in your authentication provider.
- The login is in an authentication domain that exists in your authentication provider.

Note: Logins in authentication domains that don't exist in your authentication provider are preserved by default. △

- If the synchronization includes groups (or roles), memberships for participating SAS identities are updated to match the memberships in your authentication provider. A change in the input grpType value (which determines whether an object is a group or a role) does not cause any update to the metadata.
- Locations, telephone numbers, and e-mail addresses for participating identities are updated. Use exceptions to prevent updates to contact information that is added interactively. See "Defining Exceptions" on page 240
- New authentication domains are added. By default, no authentication domains are removed.

These constraints apply to updates:

- When combined with information that already exists in the metadata, the change data must meet all uniqueness requirements. See "Unique Names and IDs" on page 33.
- In order to add a user, group, or role, only a name and one external identity value (keyid) is required. However, each user should also have at least one login in order to establish an individual SAS identity. User IDs must be qualified. See "User ID Formats" on page 32.

How to Synchronize Identities

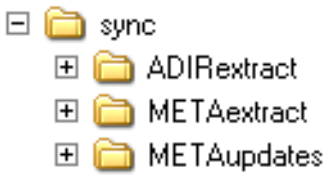
Note: It is a good practice to run a backup before you perform a synchronization (at least until your program is proven). △

To synchronize identity information:

1. If you want to include identities that weren't originally imported, use SAS Management Console to add a correct external identity value on the **`General`** tab of each such identity. See "External Identities" on page 226.
2. If you want to exclude identities or attributes from the update, create an exceptions data set.

   CAUTION:
   If you used SAS Management Console to make updates to imported identities, those updates are not automatically preserved during batch synchronization. To preserve such information, see "Defining Exceptions" on page 240. △

3. In the operating system, set up three directories: an enterprise extract directory, a metadata extract directory, and a change tables directory. For example:

4. In the SAS Program Editor, adapt the sample synchronization code to create your own program. See "Sample Code for User Synchronization" on page 232.

5 Submit the code and review the log. To address any errors, make changes in the source tables, the exceptions tables, or the metadata. For details about the errors, examine the errorsds table. See "%MDUCHGV" on page 241. After making corrections, run the synchronization program again.

Note: An alternative method for dealing with errors is to re-execute the %MDUCMP macro with EXCEPTIONS=ERRORSDS. This recreates the change tables without the offending entries. If an exceptions data set is already being used, you can append the content of the ERRORSDS data set to that data set. △

6 In the User Manager plug-in in SAS Management Console, verify that the metadata reflects the changes that you expect to see.

Note: To ensure that current information is displayed, right-click User Manager and select **View ▶ Refresh**. △

Sample Code for Generic File Import

This code demonstrates one way to import generic identity information. This example is intended to illustrate the structure and relationships of the identity data, rather than to suggest that you enter large quantities of data using this approach. The program uses macro variables to define the tables and uses DATALINES statements to supply input data. The %MDUIMPLB macro creates XML from the tables and submits that XML (in blocks) to the metadata server.

```
/* Specify connection options. Use an unrestricted user ID. */
options metaserver=machine-name metauser="userID" metapass="password";

/* Initialize the macro variables that create canonical tables. */
%mduimpc();

/* Create the person table. */
data &persontbla ;
     %definepersoncols;
     infile datalines delimiter=',' missover;
         input keyid name description title;
    datalines;
P001,Michelle Harrell,Mgr of Operations,Sr. Mgr
P002,Fred Granite,NE Region Acct. Rep.,Account Rep II
P003,Brian Davis,Accounting System Developer,Senior Programmer
;

/* Create the phone table. */
data &phonetbla ;
     %definephonecols;
     infile datalines delimiter=',' missover;
         input keyid phoneNumber phoneType;
    datalines;
P001,x1532,Office
P001,(919) 555-1212,Home
P003,x2312,Office
;

/* Create the location table. */
data &locationtbla ;
```

```
     %definelocationcols;
     infile datalines delimiter=',' missover;
         input keyid locationName locationType address city postalcode area country;
    datalines;
P001,My Company,Office,123 Oak Ave,Clayton,20711,CA,USA
P001,Michelle Harrell,Home,105 Seth Ct.,Apex,20765,CA,USA
P002,Fred Granite,Home,2138 Pond St.,Greenlevel,20832,CA,USA
P002,My Company,Office,123 Oak Ave,Clayton,20711,CA,USA
P003,My Company,Office,123 Oak Ave,Clayton,20711,CA,USA
;

/* Create the email table. */
data &emailtbla ;
     %defineemailcols;
     infile datalines delimiter=',' missover;
         input keyid emailAddr emailtype;
    datalines;
P001,michelle@mycompany.com,business
P001,bosslady1@hotmail.com,home
P002,fred@mycompany.com,business
P003,brian@mycompany.com,business
;

/* Create the idgrp table. */
data &idgrptbla ;
     %defineidgrpcols;
     infile datalines delimiter=',' missover;
         input keyid name description grpType;
    datalines;
G001,Operations Staff,Members of the operations department,
G002,All Groups,Group containing all groups,
G003,Backup Operators, ,
;

/* Create the grpmems table. */
data &idgrpmemstbla;
     %defineidgrpmemscols;
     infile datalines delimiter=',' missover;
         input grpkeyid memkeyid;
    datalines;
G001,P001
G002,G001
G002,G003
G003,G001
G003,P003
;

/* Create the authdomain table. */
data &authdomtbla;
     %defineauthdomcols;
     infile datalines delimiter=',' missover;
         input keyid authDomName;
    datalines;
A001,DefaultAuth
```

```
A002,UnixAuth
;

/* Create the logins table. */
data &logintbla;
     %definelogincols;
     infile datalines delimiter=',' missover;
         input keyid userid password authdomkeyid;
    datalines;
P001,WinNet\Michelle, ,A001
P001,Michelle, ,A002
P002,WinNet\Fred, ,
P003,WinNet\Brian, ,
P003,Brian, ,A002
;

/* Load the information from the work library into the metadata server. */
%mduimplb();
```

Sample Code for User Synchronization

This example synchronization program is intended to help you complete step 4 in "How to Synchronize Identities" on page 229.

```
/*Specify connection options. Use an unrestricted user ID.*/
options metaserver=machine-name metauser="userID" metapass="password";

/*Specify the directory  for the extracted AD data (master tables).❶*/
libname adir "drive:\path\ADIRextract";

/* Specify the directory for the extracted  metadata (target tables).*/
libname meta "drive:\path\METAextract";

/* Specify the directory  for the comparison output (change tables).*/
libname updates "drive:\path\METAupdates";

/* Extract identity information from AD (master).❷*/
%let _EXTRACTONLY = ;
%include "drive:\path\myimportad.sas";

/* Extract identity information from the metadata (target).*/
%mduextr(libref=meta);

/* Compare AD (master) to metadata (target).❸*/
%mducmp(master=adir, target=meta, change=updates);

/* Validate the change tables.*/
%mduchgv(change=updates, target=meta, temp=work, errorsds=work.mduchgverrors);

/* Load  the  changes into the metadata.❹*/
%mduchglb(change=updates);
```

The numbers in the preceding code correspond to these points:

1 The AD extract location should match the importlibref location in your import program. For example, if your original importad.sas put the normalized AD data in the work library, you might revise importad.sas by replacing the line:

```
%let importlibref=work;
```

with these two lines:

```
libname adir "drive:\path\ADIRextract";
 %let importlibref=adir;
```

2 This causes the importad.sas. program to only extract the data (and not perform the load).

3 If you need to exclude items from the comparison, add the EXCEPTIONS=*dataset* parameter here. See "Defining Exceptions" on page 240.

4 If you prefer to not load any changes if %MDUCHGV finds an integrity violation, replace the displayed line with this code:

```
%macro exec_mduchglb;
   %if (&MDUCHGV_ERRORS ^= 0) %then %do;
      %put ERROR: Validation errors detected by %nrstr(%mduchgv). Load not attempted.;
      %return;
      %end;
   %mduchglb(change=updates);
%mend;

%exec_mduchglb;
```

About the Sample Code for UNIX /etc/passwd Import

Note: This code is in *SAS-installation-directory***\SASFoundation\9.2\core\sample\importpw.sas** (Windows) or *SAS-installation-directory***/SASFoundation/9.2/samples/base/importpw.sas** (UNIX). This topic highlights key points about the code. △

This program expects user information in **/etc/passwd** files in this form:

```
user name : password : uid (numeric) : primary group id : <continued>
gcos-field: home-directory : login-shell
```

The gcos-field item consists of additional comma-delimited fields in this form:

```
Person Name, Office, phone extension, misc, employeeid
```

For example:

```
user name : password : uid (numeric) : primary group id : <continued>
Person Name, Office, phone extension, misc, employeeid:<continued>
home-directory : login-shell
```

Here are details about this code:

- The colons delimit standard fields; the commas delimit items within the gcos-field (you can use any delimiter other than a colon).
- If your **/etc/passwd** file has a different format, modify the sample program to either exclude information from the extraction or to extract it from the appropriate fields.
- If the employeeid field for an **/etc/passwd** entry is empty, that entry is dropped from the import.

- The values in the Person Name field should be unique. The program includes a DUPLICATEPERSONS macro variable that, when set to RECODE, changes a duplicate value in the Person Name field to the user ID value of the PW file entry. However, if the DUPLICATEPERSONS macro variable is set to any other value, users with duplicate names in the Person Name field are deleted.

The program expects group information in an **/etc/group** file. The standard format is as follows:

```
groupname : password : gid (numeric) : comma-delimited list of users
```

Here are details about this file:

- Users are identified by user name (rather than by a numeric user ID).
- A group cannot be a member of another group.
- To exclude a user or group, enter an asterisk (*) in the password field. The import program drops entries that contain an asterisk in the password field.

The following table highlights selected variables:

Table A2.1 Selected Variables in a UNIX File Import

Variable Name	Purpose	Notes
UID	Provides an external identity value for each metadata user and group that this program creates.	Use a field that contains a unique and unchanging value for each entry in an **/etc/passwd** and an **/etc/group** files. In the sample code, the uid field of the gcos field is used. If this field is empty for any records, those records are excluded from the import.
MetadataAuthDomain	Enables all metadata logins that this program creates to be associated with an authentication domain.	Specify a SAS authentication domain name. The supplied value is typically **DefaultAuth**.
UNIXEMAILDOMAIN	Enables construction of an e-mail address for each user.	Specify the UNIX domain for the extracted identities. The supplied value is appended to extracted user IDs to yield e-mail addresses in the form *userid@supplied-value*. If your e-mail addresses follow a different convention, modify this part of the code.
PWExtIDTAG	Provides a label for all metadata items that this program creates. The label indicates which objects were created by this program.	Specify a descriptive label that will be applied to all imported objects to indicate where they came from. The default value is **Passwd File Import**. Do not quote this value. If you select the **External Identities** button on an imported identity's **General** tab (in SAS Management Console), you will see this label in the **Context** column of the **External Identities** dialog box.

About the Sample Code for Active Directory Import

Note: This code is in *SAS-installation-directory***\SASFoundation\9.2\core\sample\importad.sas** (Windows) or *SAS-installation-directory***/SASFoundation/9.2/samples/base/importad.sas** (UNIX). This topic highlights key points about the code. △

Here are some tips for using the program:

- The code uses the SAS interface to LDAP (the LDAP CALL Routine interface) to extract information from Active Directory.
- The code references standard Active Directory schemas to identify user and group attributes. If your site has extended the standard schema, you might need to make changes in section 3 to reference additional or alternate attributes.
- The code uses filters to segment retrieval. It might be necessary to alter the filters in sections 3 (user extraction) and 4 (group extraction) to better fit the contents of your Active Directory server. If the number of records in a request exceeds Active Directory's maximum query limit, only a subset of the requested records is returned. The Microsoft utility program LDIFDE can be useful in defining appropriate filters.
- The code won't import membership information for a group that has more than 1500 members. (This limitation is version-specific. Check the documentation for your Active Directory server for details). To incorporate an oversize group, use an approach like this:
  1. Rewrite the section 4 filters to exclude large groups.
  2. Create an additional extraction that uses LDAP range retrieval specifiers to extract the large group membership information in segments. See the LDAPS_SEARCH CALL routine in the *SAS Integration Technologies: Directory Services Reference*.
  3. Add that membership information to the main extracted grpmems table.

The following table highlights selected macro variables:

Table A2.2 Selected Macro Variables in an Active Directory Import

Variable Name	Purpose	Notes
keyidvar	Provides an external identity value for each metadata user that this program creates.	Specify an LDAP attribute that contains a unique and unchanging value for each user. The sample code uses the employeeID attribute [1]
MetadataAuthDomain	Enables all metadata logins that this program creates to be associated with an authentication domain.	Specify a SAS authentication domain name. This value isn't related to a Windows domain name. In the standard configuration, the correct value is **`DefaultAuth`** (or, if you have configured Web authentication and are extracting information for users who use only Web clients, **`web`**).
WindowsDomain	Enables construction of a qualified user ID in each login that this program creates.	Provide the Windows domain name for the extracted identities. The supplied value is prepended to each extracted user ID to yield qualified IDs in the form *supplied-value**userID*.
ADExtIDTag	Provides a label for all metadata items that this program creates. The label indicates which objects were created by this program.	Specify a descriptive label that will be applied to all imported objects to indicate where they came from. The default value is **`Active Directory Import`**. Do not quote this value. If you select the **`External Identities`** button on an imported identity's **`General`** tab (in SAS Management Console), you will see this label in the **`Context`** column of the **`External Identities`** dialog box.

1 If this attribute is empty, consider using sAMAccountName or distinguished name.

Reference: User Import and Synchronization Macros

The user import and synchronization macros are in *SAS-installation-directory*`\SASFoundation\9.2\core\sasmacro` (Windows) or *SAS-installation-directory*`/SASFoundation/9.2/sasautos` (UNIX). The following topics provide basic reference information that helps you use these macros to perform standard user import and synchronization tasks.

%MDUIMPC

Defines a set of macro variables that create canonical tables and, optionally, creates empty tables for views for CSV files.

Used in: User import, user synchronization

Depicted in: Figure A2.1 on page 222, Figure A2.2 on page 223

Syntax

%MDUIMPC (<LIBREF=*libref*>, <MAKETABLE=0|1|2>, <INFILEREF=*fileref*>, <FILEHEADER=1|*other_value*>);

All parameters for this macro are optional.

LIBREF
specifies the output location for any canonical tables or views that are created (the default is `Work`).

Note: If you choose to not create tables or views, this parameter is not used. △

MAKETABLE
controls whether any tables or views are created.

`0`	specifies that no tables or views are created (this is the default).
`1`	specifies that empty canonical tables are created.
`2`	specifies that DATA step views of the CSV files in the *fileref* directory are created.

INFILEREF
specifies the location of the input CSV source files (used only when MAKETABLE=2). The CSV input files must have names and contents that match the table names and structures of the canonical tables. On UNIX, the file names must be lowercase.

FILEHEADERS
indicates whether the CVS input files have a header line (used only when MAKETABLE=2).

`1`	indicates that the input files have header information in the first row.
other_value	indicates that the data begins in the first row.

Tables and Macro Variables Created by %MDUIMPC

Table A2.3 Tables and Macro Variables Created by %MDUIMPC

Table	Purpose	Name/Keep List Global Variable	Column Attributes Macro
person	Create users	&persontbla;	%definepersoncols
location	Store addresses	&locationtbla;	%definelocationcols
phone	Store phone numbers	&phonetbla;	%definephonecols
email	Store email addresses	&emailtbla;	%defineemailcols
idgrps	Create groups and roles	&idgrptbla;	%defineidgrpcols
grpmems	Store membership information	&idgrpmemstbla;	%defineidgrpmemscols
authdomain	Create SAS authentication domains	&authdomtbla;	%defineauthdomcols
logins	Create logins	&logintbla;	%definelogincols

See "Canonical Tables" on page 224.

%MDUIMPLB

Generates and submits blocks of XML to load identity information.

Used in: User import
Depicted in: Figure A2.1 on page 222
Requirement: Connection to the metadata server

Syntax

%MDUIMPLB (<LIBREF=*libref*>, <TEMP=*libref*>, <FAILEDOBJS=*table_name*>, <EXTIDTAG=*context_value*>, <BLKSIZE=#*_of_records*>);

This macro uses PROC METADATA to submit load requests. All parameters for this macro are optional.

LIBREF
specifies the location of the input data. The default is **`Work`**.

TEMP
specifies the location for temporary data sets during processing. The default is **`Work`**.

FAILEDOBJS
is a table that contains the entire contents of all blocks that contain any problematic data. The default is **`mduimplb_failedobjs`**. See "Columns in the FAILEDOBJS Table" on page 238.

This table is populated if integrity and completeness requirements aren't met or if a metadata server request failure occurs during processing. See "What Information Is Imported?" on page 227.

EXTIDTAG
indicates the origin of identity information. The default for this string is **`IdentityImport`**.

BLKSIZE
specifies the number of identities in a block. Each block of XML is generated and submitted to the metadata server independently. The default is 100.

Note: The corresponding macro from previous releases, %MDUIMPL, is still supported. See the 9.1 version of this document. △

Columns in the FAILEDOBJS Table

Table A2.4 FAILEDOBJS

Column	Contents
tablename	Name of an intermediate input table
chgtype	A (add), U (update), or D (delete)
objtype	Internal metadata object type
Name	Name of a user, group or role
Objid	Internal object ID
KeyId	Key ID for an item
UserId	User ID value in a login
AuthDomKeyId	Key ID of an authentication domain
MemObjType*	Internal metadata object type for a member
MemName*	Person indicates a user, IdentityGroup indicates a group
MemObjID*	Internal metadata object ID
MemKeyId*	Key ID of a member

* This column is populated only if there is an error in membership information.

%MDUEXTR

Extracts identity information from the metadata.

Used in: User synchronization

Depicted in: Figure A2.2 on page 223

Requirement: Connection to the metadata server

Syntax

%MDUEXTR (LIBREF=*SAS-library*);

This macro uses PROC METADATA to submit extraction requests.

LIBREF
specifies the location to which the metadata is extracted. This is a required parameter.

The tables that contain the extracted information are in the canonical format with these additional columns:

objectid	specifies a unique metadata identifier. These values map the extracted metadata information back to the source objects in the repository.
extid	specifies an external identity value. These values map the extracted metadata information data to entries in the information that was extracted from your authentication provider.

%MDUCMP

Generates data sets that contain the changes that must be made to the metadata.

Used in: User synchronization

Depicted in: Figure A2.2 on page 223

Syntax

%MDUCMP (MASTER=*libref*, TARGET=*libref*, CHANGE=*libref*,
EXCEPTIONS=<*libref*.>*dataset* , <EXTERNONLY=0 | 1>,
<AUTHDOMCOMPARE=*name* | *keyid*>);

MASTER
specifies the location of the master tables (use the libref that you specify in %MDUIMPC).

TARGET
specifies the location of the target tables that contain information extracted from the metadata (use the libref that you specify in %MDUEXTR).

CHANGE
specifies the location for the change tables. These tables are created (*xxx* is the base name of each canonical table):

*xxx*_add	contains users, groups, and roles to be added to the target tables to make them look like the master tables.
*xxx*_update	contains users, groups, and roles to be modified in the target tables to make them look like the master tables.
*xxx*_delete	contains users, groups, and roles to be deleted from the target tables to make them look like the master tables.
person_summary	summarizes changes to users (Person objects).
idgrps_summary	summarizes changes to groups and roles (IdentityGroup objects).
authdomain_ summary	summarizes changes to SAS authentication domains (AuthenticationDomain objects).

EXCEPTIONS
specifies a data set that contains exception values. See “Defining Exceptions” on page 240.

EXTERNONLY
defines the scope of the comparison.

Note: Unless the master data set has an ObjectId column, this option has no effect. A typical master data set does not include an ObjectId column. A master data set that is extracted from the SAS Metadata Repository (rather than from your authentication provider) does include an ObjectId column. Extraction of a master data set from the metadata repository happens in the identity synchronization processes for some solutions. △

1 specifies that only identities that have an external identity value are included in the comparison. This is the default value.

0 specifies that all identities are included in the comparison.

If EXTERNONLY=1 but AUTHDOMCOMPARE=name, all authentication domains are compared. In other words, for authentication domains AUTHDOMCOMPARE=name overrides EXTERNONLY=1.

AUTHDOMCOMPARE
defines how authentication domains are compared.

name compares all authentication domains by name. Prevents deletion and renaming of all authentication domains. Prevents deletion of logins in authentication domains that do not exist in the master data set. This is the default.

keyid compares by keyid. Can cause deletion of authentication domains that were originally imported but are not present in the master data set. Can cause renaming of authentication domains that were originally imported but have a different name in the master data set. Does not prevent deletion of logins in authentication domains that do not exist in the master data set.

CAUTION:
If you specify AUTHDOMCOMPARE=keyid, authentication domains and logins that are interactively created might be deleted. For a standard synchronization, don't use this setting. △

Defining Exceptions

The exceptions data has these columns:

tablename specifies the name of the canonical table to which the exception applies. Valid values are **person**, **logins**, **email**, **phone**, **location**, **idgrps**, **grpmems**, and **authdomain**.

filter specifies a SAS WHERE clause expression (without the WHERE) to apply against the corresponding table. The WHERE clause consists of a canonical table column name and an exception value.

For example, consider this exceptions data set:

```
phone      PhoneType="manual Phone"
email      EmailType="manual Email"
logins     authDomKeyId="A002"
logins     userid="testid%"
```

Each line protects a set of objects in a particular target table, ensuring that those metadata objects are preserved.

- The first entry excludes objects in the target phone table that have a PhoneType of **manual Phone**. In SAS Management Console, the PhoneType is displayed in the **Type** field in the Phone Properties dialog box.

- The second entry excludes objects in the target email table that have the EmailType value **`manual Email`**. In SAS Management Console, the EmailType is displayed in the **`Type`** field in the Email Properties dialog box.
- The last entry excludes any objects in the target logins table which have a userid value that begins with **`testid`**.

Note: Logins that are in authentication domains that do not exist in the master tables are preserved by default. It is not necessary to define exceptions for such logins. △

%MDUCHGV

Checks the change tables against the target tables to ensure that the updates do not introduce any integrity problems.

Used in: User synchronization

Depicted in: Figure A2.2 on page 223

Syntax

%MDUCHGV (TARGET=*libref*, CHANGE=*libref*, <TEMP=*libref*>, ERRORSDS=*name*);

TARGET
specifies the location of the target canonical tables. This is typically the same libref that you specify in the %MDUEXTR macro.

CHANGE
specifies the location of the change tables (that were populated by the %MDUCMP macro).

TEMP
specifies the location for temporary tables (the default is **`Work`**).

ERRORSDS
identifies a data set that contains information about any integrity problems. This data set has these columns:

errcode	specifies a numeric code for a particular error.
errmsg	describes a particular error.
tablename	specifies the name of the canonical table from which a particular error item should be excepted if the ERRORSDS data set is fed into the %MDUCMP macro as the exception data set.
filter	specifies a SAS WHERE clause that is used to remove all objects related to a particular error from the change tables.
Keyid	specifies the keyid value of the offending object.
Name	specifies the Name value of the object, if the offending object is a Person or IdentityGroup.
userid	specifies the userid value of the object, if the offending object is a Login.

authDomKeyId specifies the keyid value of an authentication domain when duplicate userid values are found in an authentication domain.

If any errors are detected, this macro sets the DUCHGV_ERRORS column to a non-zero value and creates the ERRORSDS data set, which lists the errors that were found.

%MDUCHGLB

Generates XML for loading identity updates.

Used in: User synchronization

Depicted in: Figure A2.2 on page 223

Requirement: Connection to the metadata server

Syntax

%MDUCHGLB (CHANGE=*libref*, TEMP=*libref*, <FAILEDOBJS=*table_name*>,
<EXTIDTAG=*context_value*>, <BLKSIZE=*#_of_records*>);

CHANGE
specifies the location of the change tables.

TEMP
specifies the location for temporary tables. The default is **`Work`**.

FAILEDOBJS
is a table that contains the entire contents of any blocks that weren't successfully loaded into the metadata. An error within a block prevents loading of all of the data in that block. The default name for this table is **`mduchglb_failedobjs`**. See "Columns in the FAILEDOBJS Table" on page 238.

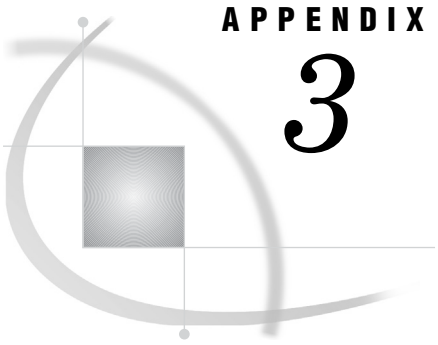
This table is populated if integrity or completeness requirements aren't met or if a metadata server request failure occurs during processing. See "What Information Is Updated?" on page 228. To address errors, see "How to Synchronize Identities" on page 229.

EXTIDTAG
Indicates the origin of identity information. The default for this string is **`IdentityImport`**.

BLKSIZE
specifies the number of added or updated objects in a block. Each block of XML is generated and submitted to the metadata server independently. The default is 100.

Note: For delete operations, BLKSIZE has no effect. △

Note: The corresponding macro from previous releases, %MDUCHGL, is still supported. See the 9.1 version of this document. △



APPENDIX

3

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Intelligence Platform: Overview*
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Intelligence Platform: Web Application Administration Guide*
- *SAS Management Console: Guide to Users and Permissions*

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 at:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: 1-800-727-3228
Fax: 1-919-531-9439
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/bookstore**

Customers outside the United States and Canada, please contact your local SAS office for assistance.

Glossary

access control template
a reusable named authorization pattern that you can apply to multiple resources. An access control template consists of a list of users and groups and indicates, for each user or group, whether permissions are granted or denied. Short form: ACT.

authentication
the process of verifying the identity of a person or process within the guidelines of a specific authorization policy.

authentication domain
a SAS internal category that pairs logins with the servers for which they are valid. For example, an Oracle server and the SAS copies of Oracle credentials might all be classified as belonging to an OracleAuth authentication domain.

authentication provider
a software component that is used for identifying and authenticating users. For example, an LDAP server or the host operating system can provide authentication.

authorization
the process of determining which users have which permissions for which resources. The outcome of the authorization process is an authorization decision that either permits or denies a specific action on a specific resource, based on the requesting user's identity and group memberships.

capability
an application feature that is under role-based management. Typically, a capability corresponds to a menu item or button. For example, a Report Creation capability might correspond to a New Report menu item in a reporting application. Capabilities are assigned to roles.

client-side pooling
a configuration in which the client application maintains a collection of reusable workspace server processes. See also puddle.

connection profile
a client-side definition of where a metadata server is located. The definition includes a computer name and a port number. In addition, the connection profile can also contain user connection information.

credential management
the reuse of cached credentials or the retrieval of stored credentials from the metadata.

credentials
the user ID and password for an account that exists in some authentication provider.

direct LDAP authentication
a configuration in which the metadata server sends credentials to an LDAP provider (such as Active Directory) for validation, bypassing the host authentication process.

encryption
the act or process of converting data to a form that only the intended recipient can read or use.

external identity
a synchronization key for a user, group, or role. For example, employee IDs are often used as external identities for users. This is an optional attribute that is needed only for identities that you batch update using the user import macros.

host authentication
a process in which a SAS server sends credentials to its host operating system for verification.

Integrated Windows authentication
a Microsoft technology that facilitates use of authentication protocols such as Kerberos. In the SAS implementation, all participating components must be in the same Windows domain or in domains that trust each other.

internal account
a SAS account that you can create as part of a user definition. Internal accounts are intended for metadata administrators and some service identities; these accounts are not intended for regular users.

internal authentication
a process in which the metadata server verifies a SAS internal account. Internal authentication is intended for only metadata administrators and some service identities.

IWA
See Integrated Windows authentication.

login
a SAS copy of information about an external account. Each login includes a user ID and belongs to one SAS user or group. Most logins do not include a password.

PAM
See pluggable authentication modules.

permission condition
a control that defines access to data at a low level, specifying who can access particular rows within a table or particular members within an OLAP cube. Such controls are typically used to subset data by a user characteristic such as employee ID or organizational unit. For example, an OLAP cube that contains employee information might have member-level controls that enable each manager to see the salary history of only that manager's employees. Similarly, a table that contains patient medical information might have row-level controls that enable each doctor to see only those rows that contain data about that doctor's patients.

pluggable authentication modules
an industry-standard technology that extends UNIX host authentication to recognize additional authentication providers.

puddle
a group of servers that are started and run using the same login credentials. Each puddle can also allow a group of clients to access the servers.

repository access control template
the access control template (ACT) that controls access to a particular repository and to resources for which access controls are not specified. You can designate one repository ACT for each metadata repository. The repository ACT is also called the default ACT.

restricted identity
a user or group that is subject to capability requirements and permission denials in the metadata environment. Anyone who isn't in the META: Unrestricted Users Role and isn't listed in the adminUsers.txt file with a preceding asterisk is a restricted identity.

role
a set of capabilities. In some applications, certain actions are available only to users or groups that have a particular role.

SAS authentication
a form of authentication in which the target SAS server is responsible for requesting or performing the authentication check. SAS servers usually meet this responsibility by asking another component (such as the server's host operating system, an LDAP provider, or the SAS Metadata Server) to perform the check. In a few cases (such as SAS internal authentication to the metadata server), the SAS server performs the check for itself. A configuration in which a SAS server trusts that another component has pre-authenticated users (for example, Web authentication) is not part of SAS authentication.

SAS token authentication
a process in which the metadata server generates and verifies SAS identity tokens to provide single sign-on to other SAS servers. Each token is a single-use, proprietary software representation of an identity.

server-side pooling
a configuration in which a SAS object spawner maintains a collection of reusable workspace server processes that are available for clients. The usage of servers in this pool is governed by the authorization rules that are set on the servers in the SAS metadata.

service identity
an identity or account that exists only for the purpose of supporting certain system activities and does not correspond to a real person. For example, the SAS Trusted User is a service identity.

single sign-on
an authentication model that enables users to access a variety of computing resources without being repeatedly prompted for their user IDs and passwords. For example, single sign-on can enable a user to access SAS servers that run on different platforms without interactively providing the user's ID and password for each platform. Single sign-on can also enable someone who is using one application to launch other applications based on the authentication that was performed when the user initially logged on.

SSO
See single sign-on.

trusted user
a privileged service account that can act on behalf of other users on a connection to the metadata server.

unrestricted identity
a user or group that has all capabilities and permissions in the metadata environment due to membership in the META: Unrestricted Users Role (or listing in the adminUsers.txt file with a preceding asterisk).

user context
a per-session, in-memory list of credentials that are available to a particular user.

Web authentication
a configuration in which users of Web applications and Web services are verified at the Web perimeter and the metadata server trusts that verification.

well-formed user definition
a user definition that includes a login with an appropriate user ID. For a Windows account, the user ID in the login must be qualified (for example, WIN\marcel or marcel@company.com). The login does not have to include a password. For metadata administrators and some service identities, it is appropriate to use an internal account instead of a login.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.