SAS® 9.1.3 Intelligence Platform

Security Administration Guide
Second Edition

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *SAS® 9.1.3 Intelligence Platform: Security Administration Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.1.3 Intelligence Platform: Security Administration Guide, Second Edition

ISBN-13: 978–1–59994–155–4
ISBN-10: 1–59994–155–4

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

1st printing, November 2006

2nd printing, June 2007

Contents

What’s New

Overview

Changes to security aspects of the SAS Intelligence Platform in Service Pack 4 include a new access requirement, a BI row-level permissions feature, and expanded support for Web authentication.

New Access Requirement: The Read Permission for Information Maps

A new access requirement affects users who view reports that are based on information maps (or otherwise interact with information maps). In order to access data through an information map, users must have the Read permission for that information map. In the interest of greater security, the Read permssion is not granted to anyone in the initial configuration. Therefore, before users can perform actions such as generating reports that are based on information maps, you must grant Read access using a strategy that is appropriate for your site. This is a new requirement with SAS Web Report Studio 3.1, SAS Information Map Studio 3.1, and Service Pack 4. For instructions, see “How to Manage Read Access” on page 117.

BI Row-Level Permissions

BI row-level permissions is a new feature that enables you to define fine-grained access controls for relational data and SAS data sets when the data is accessed through information maps. This feature requires Service Pack 4 and SAS Information Map Studio 3.1. For more information, see Chapter 10, “BI Row-Level Permissions,” on page 137.

Expanded Support for Web Authentication

Web authentication support has been expanded to include the SAS Web OLAP Viewer for Java. This enables you to authenticate users of this application using the Web server's authentication provider (rather than the metadata server's authentication provider). This is an enhanced feature that requires SAS Web OLAP Viewer 3.1. For more information, see "Using Web Authentication" on page 82.

Documentation Enhancements

The following changes have been made in the second edition of this document:

- The instructions for directly using an alternate authentication provider have been revised.
- A chapter on OLAP member-level permissions has been added.

PART 1

Before You Begin

CHAPTER 1

Understanding Authentication

Scope of This Document

This document explains the security model for the SAS Intelligence Platform and provides instructions for performing security-related administrative tasks. The emphasis is on suite-wide aspects of the security functionality that SAS provides. Some interactions with other security layers (such as operating system permissions, WebDAV access controls, and third-party database security) are noted. Detailed information about features and requirements that are unique to a particular application is provided in the administrative documentation for that application.

This document assumes that you are familiar with the concepts and terminology that are introduced in *SAS Intelligence Platform: Overview*. For a list of all of the documents that SAS publishes to support administration of the SAS Intelligence Platform, see **`support.sas.com/administration`**.

Accessibility Features in the SAS Intelligence Platform Products

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

Authentication Overview

Introduction to Authentication

Authentication is an identity verification process that attempts to determine whether users (or other entities) are who they say they are. Authentication is a prerequisite for authorization, because a user's identity is the basis for authorization decisions about which actions the user is permitted to perform with which resources.

In the SAS Intelligence Platform, a user's identity is verified first when the user logs on to an application and again as the user requests access to other systems. For example, when a user logs on to SAS Data Integration Studio, the user authenticates to the SAS Metadata Server. When the user makes a request from SAS Data Integration Studio to run a job against an Oracle table, the user must authenticate to the SAS Workspace Server that processes the request and to the Oracle server that manages the table.

How Identities Are Verified

In most cases, SAS servers rely on their host operating systems to verify identities. This process is called host authentication. For example, before allowing a user to run a stored process, a stored process server asks its host computer to authenticate the user. The host computer compares a provided user ID and password to a list of valid accounts in the operating system (or in a back-end authentication database that the operating system is using). If the provided user ID and password correspond to a valid account, the authentication is successful.

Note: As an alternative to relying on the host operating system, the metadata server and the OLAP server can make direct use of Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory to verify identities. However, the preferred way to use an alternative authentication provider is as a back-end user store behind host authentication, because direct use of LDAP and Active Directory Direct can significantly increase the need to store user IDs and passwords in the metadata repository. △

In some cases, SAS servers trust verification that has been performed by other components. The SAS Intelligence Platform supports the following trust relationships:

- □ The metadata server trusts the identity verification that the SAS OLAP Server performs.
- □ By default, the metadata server trusts the identity verification that a connecting SAS process performs.
- □ If SAS Web applications are configured to use Web server authentication, the metadata server trusts the identity verification that a Web server performs.

Related Topics:

"Using LDAP or Active Directory" on page 82

"Using Web Authentication" on page 82

"The Authentication Process" on page 11

Single Sign-On

Single sign-on enables users to access a variety of computing resources without being repeatedly prompted for their user IDs and passwords. The SAS Intelligence Platform provides these single sign-on features:

- □ Most applications can cache the credentials that a user submits to log on.
- □ All applications can retrieve credentials that have been stored in the metadata repository.
- □ Web applications can share user and session contexts.

Related Topics:

"Reuse of Credentials That Are Cached from an Interactive Log On" on page 17

"Retrieval of Credentials from the Metadata Repository" on page 18

"Shared User Context (Among Web Applications)" on page 20

Identity Management

In addition to managing user accounts in external systems, administrators must create and maintain some user information in the metadata repository. You can minimize the amount of identity information that you need to replicate in the metadata by choosing your authentication providers carefully and making appropriate use of shared accounts.

The SAS Intelligence Platform provides the following tools for management of identity information in the metadata:

- □ Administrators can use SAS Management Console to define and manage metadata identity information.
- □ Administrators can use batch processes to extract identity information from sources such as LDAP or UNIX /etc/passwd files and create corresponding identity

information in the metadata repository. Batch processes can also be used to periodically update the identity information. Batch processes cannot be used to manage passwords.

- □ Users can use the SAS Personal Login Manager desktop application to manage their own account information.

Related Topics:

"Choosing Authentication Providers" on page 52

"How to Use Shared Accounts" on page 80

Appendix 2, "Bulk-Load Processes for Identity Management," on page 185

Authentication Terminology

Introduction to Authentication Terminology

The following terms are important to understanding how authentication works in the SAS Intelligence Platform:

authentication provider
: a technology that servers or applications can use to verify that users are who they say they are. Operating systems, LDAP, Active Directory, and third-party database system authentication mechanisms are examples of authentication providers.

metadata identity
: a metadata object that represents an individual user or a group of users on a SAS Metadata Server. Each metadata identity must be unique within a metadata server.

login
: a metadata object that is owned by a metadata identity. Each login contains the user ID (and, sometimes, the password) for an account that has been established with an authentication provider. Each login corresponds to a particular user account with a particular authentication provider. For example, if you have a UNIX account with a user ID of **`tara`** and a password of **`tara1234`**, then you can store that account information in the metadata as a login.

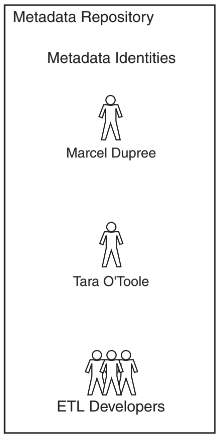
authentication domain
: a metadata object that links logins to the servers for which the logins are valid. Each authentication domain should be associated with one or more servers and with the logins that provide access to those servers. All of the computing resources within an authentication domain use the same authentication provider. You can choose to use the same groupings and names for your authentication domains as you do for your host domains or network domains, but you are not required to do so.

The following topics explain the role of metadata identities, logins, and authentication domains in the authentication model.

How Metadata Identities Are Used

Metadata identities are used as the basis for making authorization decisions and responding to requests for credentials. The following figure depicts several metadata identities within a SAS Metadata Repository.

Figure 1.1 Metadata Identities



The metadata server discovers a user's metadata identity by performing these steps:

1 The metadata server searches the metadata repository for a login that contains a user ID that matches the user ID with which the user was authenticated.

In this process, the metadata server attempts to match the fully qualified user ID. For example, if a user logs on to a server that is using Windows host authentication, and the user's Windows user ID is **`marcel`** in a Windows domain named **`WinNT`**, then the metadata server searches the repository for a login that includes the user ID **`WinNT\marcel`**. For this reason, it is important to carefully specify the user ID in each login that you create.

- □ When you create a login for a network Windows user account, specify the user ID in the form *Windows-domain-name\userID* or in the form *userID@Windows-domain-name*.
- □ When you create a login for a local Windows user account, specify the user ID in the form *machine-name\userID* or in the form *userID@machine-name*.
- □ When you create a login for an LDAP user account, specify the user ID in the form *userID@authentication-provider*.
- □ When you create a login for a Microsoft Active Directory user account, specify the user ID in the form *Windows-domain-name\userID* or in the form *userID@Windows-domain-name*.
- □ When you create a login for a UNIX or z/OS operating system user account, specify the user ID in the form *userID*.

2 The metadata server determines which metadata identity owns the login that contains the matching user ID. For example, if the metadata server finds that a login that contains the user ID **`WinNT\marcel`** is stored with the user definition for Marcel Dupree, then the metadata server knows that Marcel Dupree is the metadata identity of the user who logged on.

If there is no matching user ID in the repository, then the user's access corresponds to the access of the PUBLIC group, with these exceptions:

- □ If the user is using SAS Web Report Studio (with the surrogate user configuration) or the SAS Information Delivery Portal, then the user's access corresponds to the access that has been defined for the surrogate user. By default this is the SAS Guest User.
- □ If the user has special status as an *unrestricted user* or an *administrative user* of the metadata server, then the user can perform certain tasks even if he or she does not have an individual metadata identity.

How Logins Are Used

Logins are primarily used in two ways:

□ The metadata server uses logins to determine a user's metadata identity. When a login is used to determine a user's metadata identity, the login is functioning as an *inbound login* (the login is inbound to the metadata server). As explained in the preceding topic, the metadata server does not examine passwords or consider authentication domains in this process.

□ Applications use logins to acquire credentials as part of a single sign-on approach to authentication. An application can retrieve a login from the metadata server and send those credentials to another system that needs to verify a user's identity. When a login is used to provide access to a server other than the metadata server, the login is functioning as an *outbound login* (the login is outbound from the metadata server to another system). An outbound login must include a user ID and password that are appropriate for the server or host to which the login provides access. An outbound login must be associated with an authentication domain.

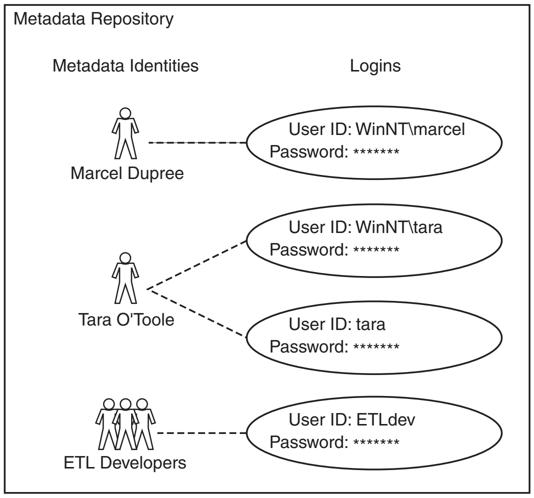
Logins can also be used in these ways:

□ The object spawner uses logins to obtain credentials for launching servers that run under designated accounts. When you configure a stored process server or a pooled workspace server, you specify an account under which the server will run. For example, during installation the stored process server is configured to run under the sassrv account. In order to launch that stored process server, the object spawner needs the credentials for the sassrv account. The object spawner obtains those credentials from a login that is owned by the SAS General Servers group. For more information, see "Accessing a Pooled SAS Workspace Server" on page 27.

□ The Xythos WebFile Server uses logins to discover and set up users for access control in the WebDAV authorization layer. Xythos builds its list of users by retrieving from the metadata server all of the logins that are associated with the authentication domain of the Xythos WebFile Server. In the default configuration, that server is associated with the DefaultAuth authentication domain, so a user must have a login that is associated with the DefaultAuth authentication domain in order to be a valid user of the Xythos WebFile Server. In alternate configurations, a user must have a login for some other authentication domain in order to be a valid user of the Xythos WebFile Server. For more information, see "Accessing a Xythos WebFile Server" on page 31.

Each login is owned by only one metadata identity. Each metadata identity can own multiple logins. The following figure depicts the relationships between logins and metadata identities in a metadata repository.

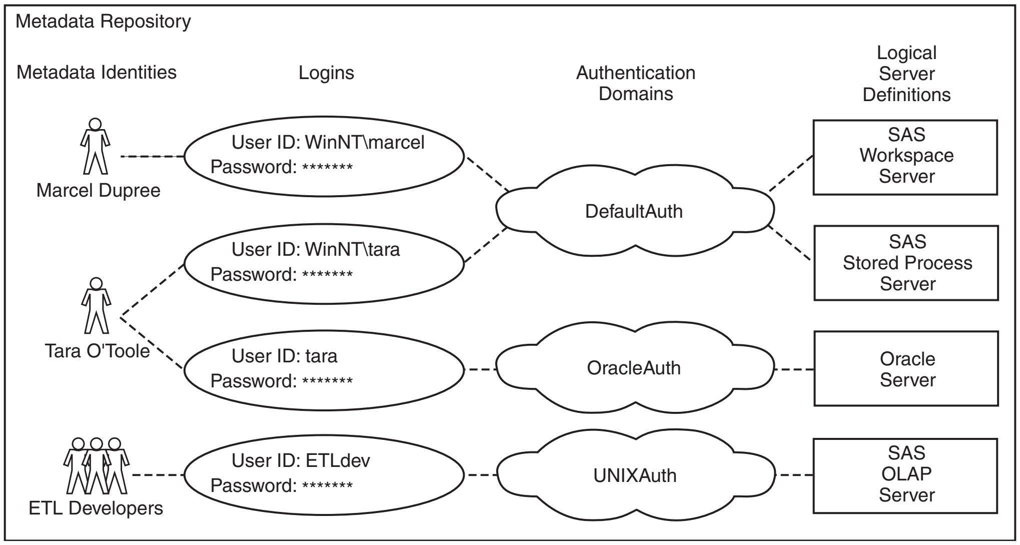
Figure 1.2 Metadata Identities and Logins



How Authentication Domains Are Used

Authentication domains are used to support single sign-on from an application to other systems. Each authentication domain corresponds to a logical grouping of servers and logins within a metadata repository. The following figure depicts the relationships between servers, authentication domains, and logins.

Figure 1.3 Metadata Identities, Logins, and Authentication Domains



When an application searches the metadata for a login that provides access to a particular server, the application uses authentication domains to determine which logins contain credentials that are appropriate for that server. For example, if Tara makes a request that requires access to the Oracle server, then the Oracle server will have to verify Tara’s identity. The application that Tara is using must provide Tara’s Oracle user ID and password to the Oracle server. The application will complete these steps:

1 Determine that the Oracle server definition is associated with the OracleAuth authentication domain.

2 Ask the metadata server for a login that is both associated with the OracleAuth authentication domain and owned by Tara's metadata identity (or by a group to which Tara's identity belongs).

In the preceding figure, Tara's second login meets these criteria. If this login includes Tara's password for the Oracle server, then Tara will be able to access that server. If Marcel makes a similar request, he will be denied access to the Oracle server because Marcel does not have a login for the OracleAuth authentication domain. For additional examples, see "Authentication Scenarios" on page 21.

Uniqueness Requirements for Names and User IDs

Within a SAS Metadata Server, the following uniqueness requirements apply to metadata identity names and stored credentials:

- □ You cannot create a user definition that has the same name as an existing user definition.
- □ You cannot create a group definition that has the same name as an existing group definition.
- □ You cannot assign the same user ID to two different metadata identities. All of the logins that include a particular user ID must be owned by the same metadata identity. This enables the metadata server to resolve each user ID to a single metadata identity.
  - □ This requirement is case-insensitive. For example, you cannot assign a login with a user ID of *smith* to one user and a login with a user ID of *SMITH* to another user.
  - □ This requirement applies to the fully qualified form of the user ID. For example, you can assign a login with a user ID of *winDEV\brown* to one user and a login with a user 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 a user ID.
  - □ This requirement cannot be mitigated by associating the logins with different SAS authentication domains. For example, if one user has a login with a user ID of *smith* that is associated with a SAS authentication domain named *DefaultAuth*, you cannot give any other user a login with the user ID *smith*, even if you plan to associate 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 user ID, the logins must be associated with different authentication domains. Within an authentication domain, each user ID must be unique. For example, if you give the person *Tara O'Toole* two logins that both have a user ID of *tara*, then you cannot 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 user ID. △

The Authentication Process

Overview of the Authentication Process

The authentication process can be thought of as occurring in two phases:

1. In the *initial authentication* phase, a user logs on with a SAS Intelligence Platform client or opens a metadata profile. The user ID and password that the user submits are sent to an authentication provider to verify the user's identity. After the user ID and password are verified, the metadata server determines the user's metadata identity.
2. In the *additional authentication* phase, a user makes a request that requires access to an additional system such as a workspace server, stored process server, or database server. The application that the user is using provides the user's credentials to the additional server. This enables the additional server to verify the user's identity against its authentication provider.

These phases are described in detail in the following sections.

Initial Authentication

Overview of Initial Authentication

Initial authentication is the verification of a user's identity based on information that the user provides when the user logs on with a SAS Intelligence Platform client. Initial authentication requires that the user have an account with the authentication provider that verifies the user ID and password that is submitted. The account can be any of the following:

- a local user account in the operating system of the computer on which the authenticating server is running
- a network user account that provides access to the operating system of the computer on which the authenticating server is running
- an LDAP or Active Directory user account (if the authenticating server is using one of these alternative authentication providers)
- a user account with any authentication provider that the Web application server uses (for applications that are configured to use Web authentication)

The initial authentication process varies depending on the software component that the user is using. The following table describes how each software component verifies identities.

Table 1.1 Initial Authentication

Type of Software Component	Identity Verification Process
Desktop applications Web applications that are using metadata server authentication	The metadata server's authentication provider verifies that the user ID and password that the user submits correspond to an existing account. For a depiction of this process, see "Initial Authentication on a Metadata Server" on page 12.
Web applications that are using Web authentication	The Web application server's authentication provider verifies that the user ID and password that the user submits correspond to an existing account. The Web application then uses *trusted user** functionality to enable the user to access the metadata server. The user does not need an account with the metadata server's authentication provider. For a depiction of this process, see "Initial Authentication on a Web Application Server" on page 13.
Components that connect directly to a SAS OLAP Server (such as the SAS OLAP Data Provider)	The SAS OLAP Server's authentication provider verifies that the user ID and password that the user submits correspond to an existing account. The SAS OLAP Server then uses *trusted user** functionality to enable the user to access the metadata server. The user does not need to have an account with the metadata server's authentication provider. For a depiction of this process, see "Initial Authentication on a SAS OLAP Server" on page 15.

* *Trusted user* functionality supports a multi-tier server environment in which user identities are authenticated by a server other than the metadata server.

After the user ID and password that the user submits are verified by the appropriate authentication provider, the proof-of-identity is complete. None of the user information that is stored in the metadata repository is used to prove the user's identity.

Next, the metadata server must discover the user's metadata identity for these reasons:

□ In order to provide authorization decisions and credential management, the metadata server needs to know who the user is.

□ Some applications have an *additional* requirement beyond proof-of-identity and will not allow users to log on unless they have a metadata identity. For example, a user must have a metadata identity in order to log on to SAS Information Map Studio or to access the SAS Information Delivery Portal beyond the public kiosk. SAS Web Report Studio can also be configured to require each user to have a metadata identity.

In order to discover your metadata identity, the metadata server examines the user IDs that are stored in the metadata repository. Passwords that are stored in the metadata repository are not examined at any point during initial authentication.

Initial Authentication on a Metadata Server

The metadata server handles initial authentication when a user logs on with the following types of applications:

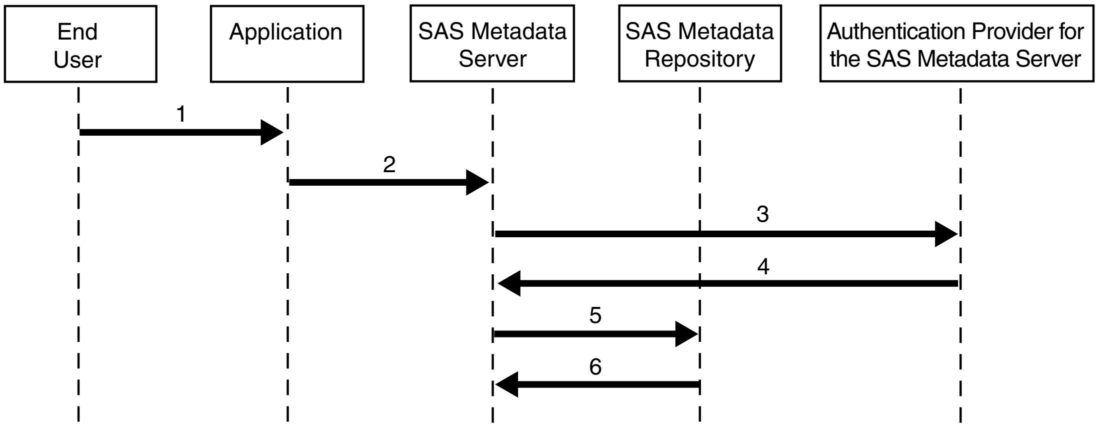
□ a desktop application such as SAS Management Console, SAS Data Integration Studio, SAS OLAP Cube Studio, or SAS Information Map Studio

□ a Web application (such as SAS Web Report Studio or the SAS Information Delivery Portal) that is configured to authenticate users on the metadata server

The following figure depicts these activities:

- □ verification of credentials that a user submits to an application that authenticates users on a metadata server
- □ determination of the user's metadata identity

Figure 1.4 Initial Authentication on a Metadata Server



In this figure, the numbered arrows correspond to the following activities:

1 The user submits a user ID and password to a SAS application (by logging on or by opening a metadata profile).

2 The application sends the user ID and password to the metadata server.

3 The metadata server passes the user ID and password to its authentication provider for verification. For example, if the authentication provider is the host operating system, then the metadata server passes the user ID and password to the operating system of the machine on which the metadata server is running.

4 The authentication provider verifies that the user ID and password combination corresponds to an existing user account. For example, if the authentication provider is the host operating system, then the user ID and password combination must correspond to a local or network user account that has been established in the operating system. After verification, the authentication provider tells the metadata server that the user ID and password are valid and sends the user ID back to the metadata server.

5 The metadata server looks for the user ID in the logins that are stored in the metadata repository.

Note: The metadata server attempts to match the user ID in its fully qualified form, as described in "How Metadata Identities Are Used" on page 6. △

6 The metadata server determines which metadata identity owns a login that contains the matching user ID.

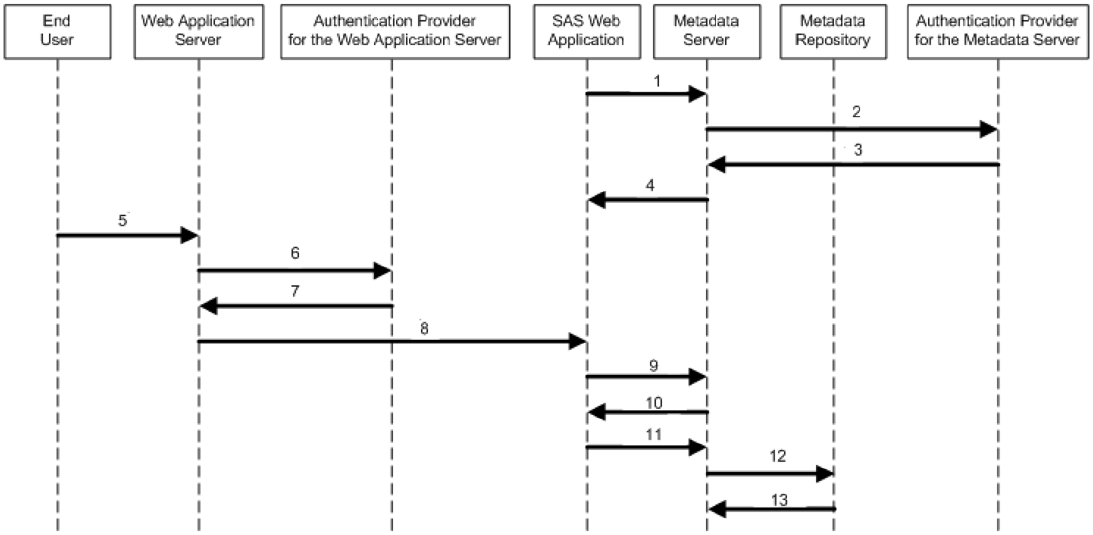
Initial Authentication on a Web Application Server

When a Web application such as SAS Web Report Studio or the SAS Information Delivery Portal is configured to use Web authentication, the authentication provider of the Web application server must verify the credentials that users submit.

The following figure depicts these activities:

- □ establishment of a trusted connection between a Web application and the metadata server
- □ verification of credentials that a user submits to a Web application that is configured to authenticate users on a Web application server
- □ determination of the user's metadata identity

Figure 1.5 Initial Authentication on a Web Application Server


End User
Web Application Server
Authentication Provider for the Web Application Server
SAS Web Application
Metadata Server
Metadata Repository
Authentication Provider for the Metadata Server
1
2
3
4
5
6
7
8
9
10
11
12
13

In this figure, the numbered arrows correspond to the following activities:

1 When the SAS Web application initializes, it sends the user ID and password for the *trusted user* (sastrust) to the metadata server to request a trusted connection.

2 The metadata server passes the sastrust user ID and password to its authentication provider.

3 The metadata server's authentication provider verifies that the sastrust user ID and password correspond to an existing account.

4 The metadata server tells the Web application that the trusted connection is accepted. This connection will be used in step 9.

5 After navigating to a URL for a SAS Web application, a user submits a user ID and password in response to a prompt from the Web application server.

6 The Web application server passes the user's ID and password to its authentication provider.

7 The Web application server's authentication provider verifies that the user's ID and password correspond to an existing account.

8 The Web application server sends the authenticated user ID to the SAS Web application.

9 The SAS Web application uses the (previously established) trusted connection to the metadata server to request a one-time-use password for the user.

10 The metadata server generates a one-time-use password for the user and sends that password to the SAS Web application. The metadata server trusts that the user's credentials have already been verified.

11 The SAS Web application uses the user's ID and the generated password to establish a connection to the metadata server for the user.

12 The metadata server looks for the user ID in the logins that are stored in the metadata repository.

Note: The metadata server attempts to match the user ID in its fully qualified form, as described in “How Metadata Identities Are Used” on page 6. △

13 The metadata server determines which metadata identity owns the login that contains the matching user ID.

Initial Authentication on a SAS OLAP Server

The SAS OLAP Server handles initial authentication when a user accesses a component that connects directly to a SAS OLAP Server. For example, when a user accesses SAS OLAP data from Microsoft Excel, the SAS OLAP Data Provider passes the user’s credentials to the SAS OLAP Server for initial authentication.

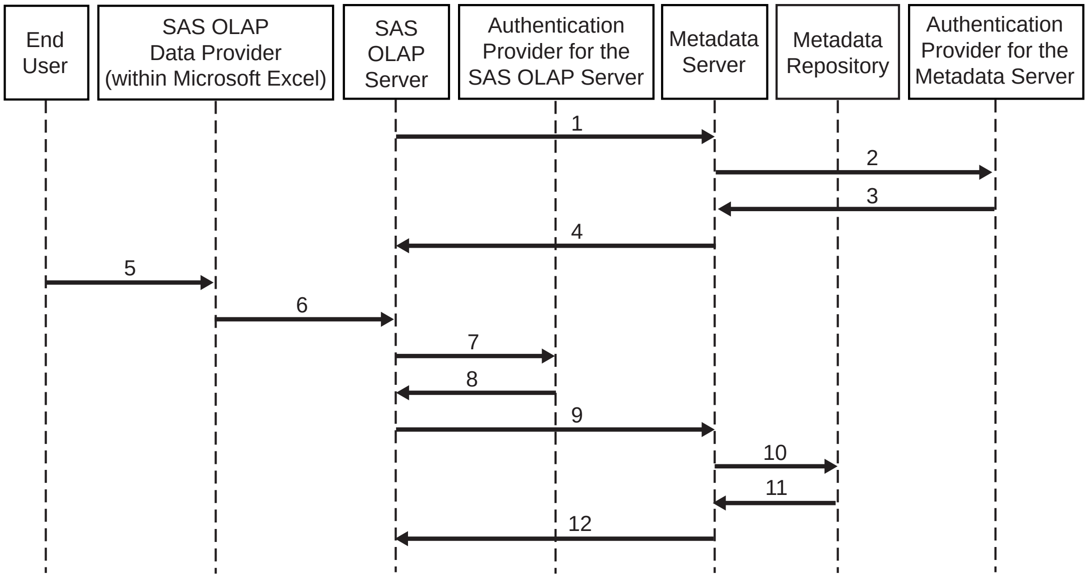
The following figure depicts these activities:

□ establishment of a trusted connection between a SAS OLAP Server and the metadata server

□ verification of credentials that a user submits to a component that connects directly to a SAS OLAP Server

□ determination of the user’s metadata identity

Figure 1.6 Initial Authentication on a SAS OLAP Server


End User
SAS OLAP Data Provider (within Microsoft Excel)
SAS OLAP Server
Authentication Provider for the SAS OLAP Server
Metadata Server
Metadata Repository
Authentication Provider for the Metadata Server
1
2
3
4
5
6
7
8
9
10
11
12

In this figure, the numbered arrows correspond to the following activities:

1 When the SAS OLAP Server initializes, it sends the user ID and password for the *trusted user* (sastrust) to the metadata server to request a trusted connection.

2 The metadata server passes the sastrust user ID and password to its authentication provider.

3 The metadata server’s authentication provider verifies that the sastrust user ID and password correspond to an existing account.

4 The metadata server tells the SAS OLAP Server that the trusted connection is accepted. This connection will be used in step 9.

5 After requesting access to SAS OLAP data from within Microsoft Excel, a user submits a user ID and password in response to a prompt from the SAS OLAP Data Provider.

6 The SAS OLAP Data Provider passes the user's ID and password to the SAS OLAP Server.

7 The SAS OLAP Server passes the user's ID and password to its authentication provider for verification.

8 The SAS OLAP Server's authentication provider verifies that the user's ID and password correspond to an existing account.

9 The SAS OLAP Server uses the (previously established) trusted user connection to request a credential handle for the user. In this process, the user's authenticated ID is passed to the metadata server. The metadata server trusts that the SAS OLAP Server has already verified the user's credentials.

Note: The SAS OLAP Server uses credential handles to specify which user is making a particular request over the trusted connection. △

10 The metadata server looks for the user ID in the logins that are stored in the metadata repository.

Note: The metadata server attempts to match the user ID in its fully qualified form, as described in "How Metadata Identities Are Used" on page 6. △

11 The metadata server determines which metadata identity owns the login that contains the matching user ID.

12 The metadata server sends a credential handle for the user to the SAS OLAP Server.

Trusted Peer Session Connections

During installation, the SAS Configuration Wizard incorporates the **`trustsaspeer`** option in the start command for the metadata server. This causes the metadata server to accept trusted peer connections from SAS processes that request access using a certain proprietary protocol. In a trusted peer connection, the metadata server trusts the authentication that another SAS process (a SAS session, workspace server, or stored process server) has already performed. Trusted peer session connections are used by applications that need to generate code or run batch jobs without explicitly providing credentials to the metadata server.

Additional Authentication

Overview of Additional Authentication

Additional authentication is the use of credentials by other systems after initial authentication. For example, when a user accesses an application such as SAS Web Report Studio to view a report that contains live data, the application might have to provide the user's credentials to a SAS Stored Process Server to enable that server to verify the user's identity.

The SAS Intelligence Platform uses a single sign-on model that enables users to access a variety of computing resources without being repeatedly prompted for their user IDs and passwords. The following sections describe the ways that applications can obtain credentials for the purpose of providing those credentials to the servers that need to verify users' identity.

Reuse of Credentials That Are Cached from an Interactive Log On

Most SAS applications cache the credentials that users provide when they log on. The cached credentials can be reused for authentication to other servers. For example, if a user logs on to SAS Information Delivery Portal and is initially authenticated on a metadata server that is using UNIX host authentication, then the user's cached UNIX credentials can be reused for authentication to a stored process server that is running on UNIX. Of course, the user's cached credentials will be valid for only one authentication provider. For example, the user's cached UNIX credentials cannot be reused for authentication to a stored process server that is running on Windows, or for authentication to a third-party database server that is using a proprietary authentication mechanism. Cached credentials are valid only for servers within only one authentication domain. In most cases, applications use cached credentials for servers that are in the DefaultAuth authentication domain. The following table provides details about how applications determine when to use cached credentials.

Table 1.2 Use of Cached Credentials

Application	**Authentication Domain for Which Cached Credentials are Used**
SAS Add-In for Microsoft Office	The authentication domain that is specified in the AuthenticationDomain Name= field in the **`CSIDL_APPDATA\SAS\Metadata Server\oms_serverinfo.xml`** file.
SAS Data Integration Studio	The authentication domain that the user specified in the metadata profile.[2]
SAS Enterprise Guide	Any authentication domain. When a user requests access to a workspace server, SAS Enterprise Guide 3.1 attempts to reuse the credentials that the user provided when the user logged on.
SAS Enterprise Miner	The authentication domain that is specified in the default_auth_domain= field in the **`SASAPCore\conf\server.config`** file.[1]
SAS Information Delivery Portal	The authentication domain that is specified in the $SERVICES_OMI_DOMAIN$= field in the **`PortalConfigure\install.properties`** file.[1]
SAS Information Map Studio	The authentication domain that the user specified in the metadata profile.
SAS Marketing Automation	The authentication domain that is specified in the **`login.config`** file (or its equivalent) for the Web container that SAS Marketing Automation is using.
SAS OLAP Cube Studio	The authentication domain that the user specified in the metadata profile.[2]

Application	Authentication Domain for Which Cached Credentials are Used
SAS Web Report Studio	The authentication domain that is specified in the $LOGON_DOMAIN$= field in the **`wrs.config`** file.[1]
SAS Web OLAP Viewer	The authentication domain that is specified in the $SERVICES_OMI_DOMAIN$= field in the **`\SASWebOlapViewerforJava\3.1\Configure\install.properties`** file.[1]

[1] The authentication domain that is specified in the application properties file must also match the authentication domain that is specified in the associated **`login.config`** file (or its equivalent). If the two values do not match, the authentication fails.

[2] If you do not specify an authentication domain in your metadata profile, then this application does not use your cached credentials.

Retrieval of Credentials from the Metadata Repository

In most deployments, there are some authentication events for which cached credentials cannot be used. For example, if a user's cached credentials are for a UNIX system, those credentials will not enable the user to access a stored process server that is running on Windows. For these authentication events, the user (or a group to which the user belongs) must have a login in the metadata that contains credentials that are appropriate for the target server.

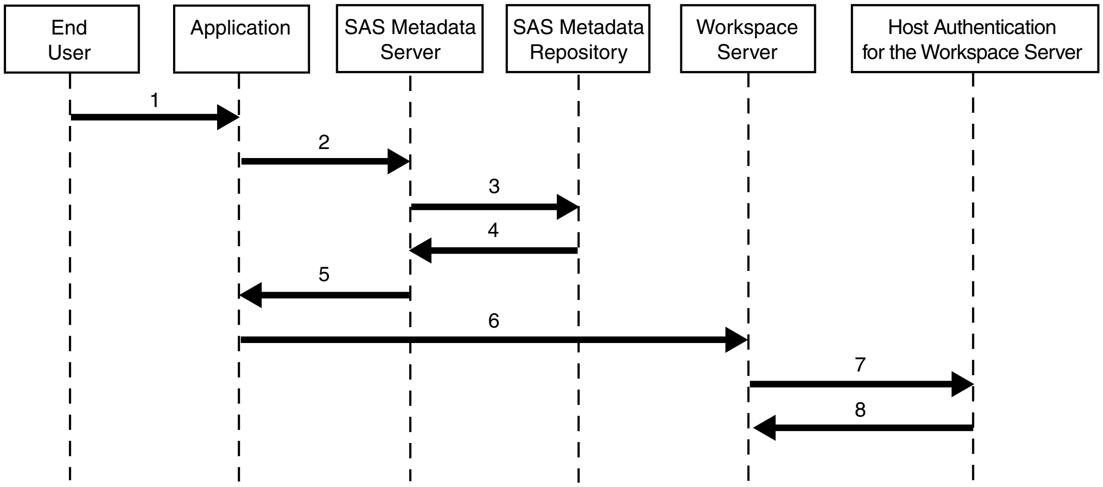
Note: In most deployments, you will need to store some passwords in the metadata. However, you can minimize the need for this by giving careful consideration to your selection of authentication providers and using shared accounts where appropriate. For details, see "Choosing Authentication Providers" on page 52 and "How to Use Shared Accounts" on page 80. △

When an application needs to provide a user's credentials to another server (and cached credentials cannot be used), the application asks the metadata server to search the metadata repository for a login that contains credentials that can be used to access the target server. The login must be owned by the user's metadata identity (or by a user group to which the user's metadata identity belongs). If the application finds an appropriate login, the application passes that user ID and password to the server that the user needs to access. The target server then uses those credentials to verify the user's identity against its authentication provider.

The following figure depicts this process. The example assumes these conditions:

- □ The user is represented in the repository by a metadata identity that owns a login that contains credentials for accessing the SAS Workspace Server.
- □ The user has already completed initial authentication.
- □ The target server is a workspace server that is not configured for pooling.

Figure 1.7 Additional Authentication



Note: In order to provide a generalized depiction that is applicable to a wide variety of target servers, the figure omits the object spawner (which is used to launch workspace servers and stored process servers). For the purposes of completeness, implementation details relating to the object spawner are noted in the following process description. Additional examples and depictions of server-specific aspects of this process are provided in "A Closer Look: Accessing SAS Servers" on page 25 and "A Closer Look: Accessing Third-Party Servers" on page 30. △

In the figure, the numbered arrows correspond to the following activities:

1 The user makes a request that requires access to a SAS Workspace Server.

2 The application recognizes that the request requires access to a workspace server, so the application goes to the metadata server to get credentials that will give the user access to a workspace server.

3 The metadata server looks for the requested credentials in the metadata repository. The credentials must meet both of these criteria:

   - □ The credentials are stored in a login that is owned by the requesting user's metadata identity (or by a group to which that identity belongs).
   - □ The credentials are stored in a login that is associated with the authentication domain in which the workspace server is registered.

4 The metadata server locates the appropriate credentials in the metadata repository and retrieves those credentials from the metadata repository.

5 The metadata server sends the credentials to the requesting application.

6 The application sends the credentials to the target server.

   Note: Because the target server is an unpooled workspace server, the application actually sends the credentials to the object spawner that will launch the workspace server (rather than to the workspace server itself). △

7 The target server passes the credentials to its authentication provider for verification.

   Note: In this example, it is actually the object spawner (rather than the workspace server) that passes the credentials to its authentication provider for verification. The authentication provider for a workspace server is always the host operating system. △

8 The authentication provider tells the target server that the credentials are valid. The target server then accepts the connection.

Note: In this example, the host operating system tells the object spawner (rather than the workspace server) that the credentials are valid. The object spawner then launches a workspace server for the requesting user. △

Shared User Context (Among Web Applications)

The previous topics describe single sign-on from an application to different servers. SAS Web applications can also support single sign-on from one application to another. When Web applications share user context and session information, users can launch one Web application from within another Web application without having to log on to the second application. For example, because the SAS Web Report Viewer and the SAS Information Delivery Portal use the same remotely deployed session service, a user can access the SAS Web Report Viewer from the portal application without logging in again. In this example, the SAS Web Report Viewer shares the session and user context that was initiated when the user logged on to the SAS Information Delivery Portal.

Interactive Prompting for SAS Server Credentials

If credentials cannot be otherwise obtained, some SAS applications prompt users for their credentials for accessing SAS servers. For example, SAS Data Integration Studio prompts users for their user ID and password for a workspace server if those credentials are not stored in the metadata repository. Interactive prompting is available only for accessing SAS servers. For authentication to a third-party database server, credentials must be stored in the metadata repository, as described in “Example: Managing Authentication to a Database Server” on page 81.

Summary: Credential Management Features by Client

The following table shows which credential management features work with each SAS client.

Table 1.3 Accessing Additional Resources after Initial Authentication

Client	Supported Credential Management Features			
	Retrieval of Stored Credentials	**Reuse of Cached Credentials**	**Interactive Prompting**	**Sharing User Context**
SAS Add-In for Microsoft Office	x	x	x	
SAS Data Integration Studio	x	x	x	
SAS Enterprise Guide	x	x[1]	x	
SAS Enterprise Miner	x	x	x	
SAS Information Delivery Portal	x	x		x
SAS Information Map Studio	x	x		

Client	Supported Credential Management Features			
	Retrieval of Stored Credentials	Reuse of Cached Credentials	Interactive Prompting	Sharing User Context
SAS OLAP Cube Studio	x	x	x	
SAS Web Report Studio	x	x		x
SAS Web OLAP Viewer	x	x		x
SAS Stored Process Application	x	x		x

[1] SAS Enterprise Guide 3.1 attempts to reuse cached credentials to provide access to workspace servers (but not to provide access to stored process servers).

Authentication Scenarios

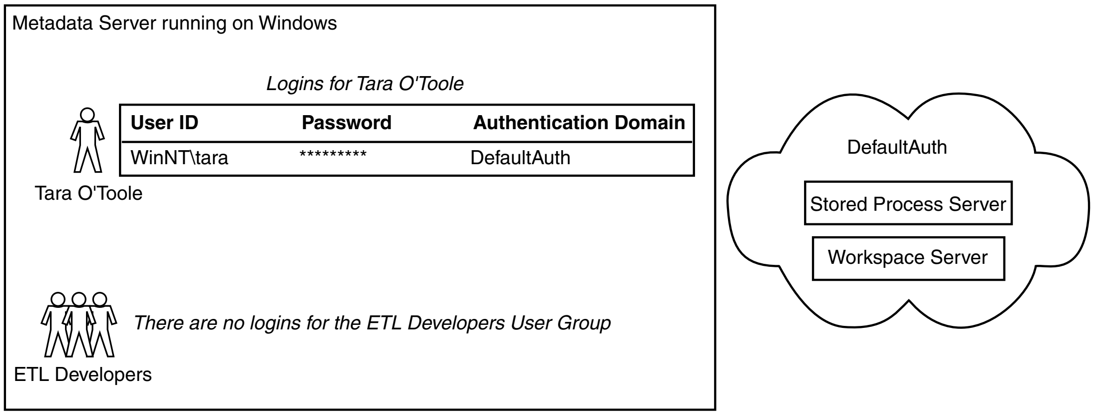
Introduction to Authentication Scenarios

This section explains the relationships between servers, authentication domains, and logins in a variety of deployment scenarios. In each scenario, the logins that are stored in the metadata for an individual user (Tara O'Toole) and a particular user group (ETL Developers) are identified.

Single Platform Environments

In a homogeneous environment, you might need only one authentication domain. The following figure depicts a deployment in which all of the logical servers and all of the logins for all metadata identities are associated with an authentication domain that is named DefaultAuth.

Figure 1.8 Homogeneous Environment, One Authentication Domain


Metadata Server running on Windows
Logins for Tara O'Toole
Tara O'Toole
There are no logins for the ETL Developers User Group
ETL Developers
DefaultAuth
Stored Process Server
Workspace Server

User ID	Password	Authentication Domain
WinNT\tara	*********	DefaultAuth

In this figure, the metadata identity that represents Tara owns only one login, which functions as both an inbound and an outbound login. Because the servers are running under Windows, the user ID in the login is fully qualified with the name of the Windows domain (WinNT). Because Tara's password is stored in the login, Tara will be able to access the workspace server and stored process server without being prompted for her credentials.

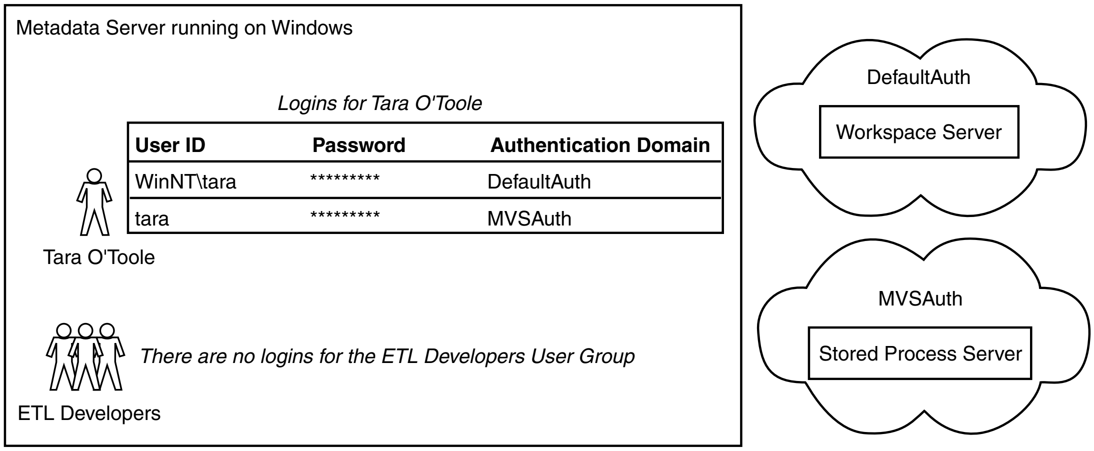
Note: If all of the applications that Tara uses can cache credentials, then Tara's login does not have to include a password. △

In this deployment, no logins have been defined for the ETL Developers user group. This user group exists to simplify administration of access controls.

Mixed Platform Environments

In a multi-host environment, you will usually need more than one authentication domain. For example, if you modify the previous deployment by moving the stored process server to z/OS, then you will need an additional authentication domain, because your users access servers on z/OS using different credentials than they use on Windows. In the metadata, you need to link the stored process server to the logins that contain credentials for accessing that server. You create this link by associating both the server and the logins with a new authentication domain. The following figure depicts this modification to the previous deployment.

Figure 1.9 Mixed Environment, Two Authentication Domains



User ID	Password	Authentication Domain
WinNT\tara	*********	DefaultAuth
tara	*********	MVSAuth

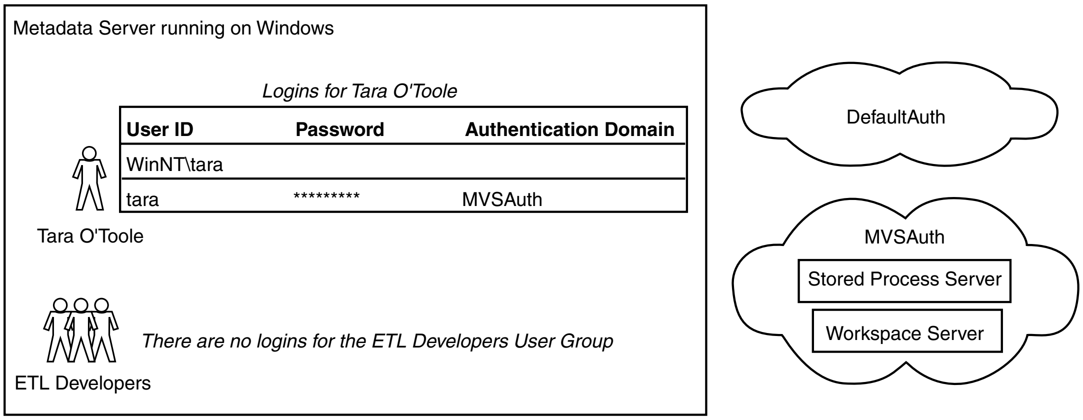
In this figure, a new authentication domain named MVSAuth has been defined, and the stored process server has been registered in that authentication domain. Two logins have been defined for Tara:

- □ The first login is for the DefaultAuth authentication domain. This login is used by the metadata server to determine Tara's identity and by the workspace server during additional authentication.
- □ The second login is for the MVSAuth authentication domain. This login enables Tara to access the stored process server during additional authentication.

Note: If the applications that Tara uses cache her credentials, then Tara can access the workspace server using credentials that are cached from initial authentication. In this scenario, Tara's first login would not have to include a password. △

The next figure depicts the deployment after you move the workspace server to z/OS. Now only the metadata server is running under Windows. All of the other servers are running under z/OS and are registered in the MVSAuth authentication domain.

Figure 1.10 Mixed Environment, One Authentication Domain



User ID	Password	Authentication Domain
WinNT\tara		
tara	*********	MVSAuth

In this figure, the DefaultAuth authentication domain still exists, but it is not associated with any servers or logins. Tara still owns two logins, but it is no longer essential to include a password or an authentication domain in the first login. Tara's first login is now only used to determine her metadata identity; it is not used for any other purposes.

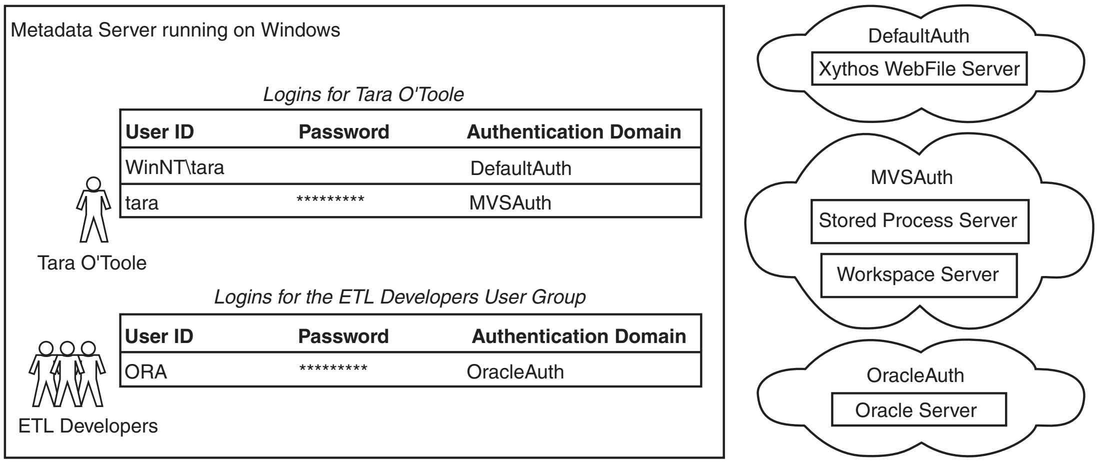
Diverse Environments

In a diverse environment, you might need more authentication domains. In this example, you add two servers to the previous deployment:

- an Oracle server that uses database authentication. When you add this server, you must add another authentication domain, because your users access the Oracle server with different credentials than they use to access the other servers. In the metadata, you must link the Oracle server to the logins that contain credentials for accessing that server. You create this link by associating both the Oracle server and the logins with a new authentication domain.
- a Xythos WebFile Server that delegates authentication to the SAS Metadata Server.* When you add the Xythos server, you do not need to add a new authentication domain, because your users will use their metadata server credentials to access the Xythos server. However, for each user who will access resources on the Xythos server, you must specify the DefaultAuth authentication domain on the login that the metadata server uses to determine that user's identity. This enables the user to authenticate to the Xythos server.

The following figure depicts the revised deployment.

Figure 1.11 Diverse Environment, Multiple Authentication Domains



Logins for Tara O'Toole

User ID	Password	Authentication Domain
WinNT\tara		DefaultAuth
tara	*********	MVSAuth

Logins for the ETL Developers User Group

User ID	Password	Authentication Domain
ORA	*********	OracleAuth

In the figure, a new authentication domain named OracleAuth has been defined, and an Oracle server has been registered in that authentication domain. The Xythos WFS server has been added to the DefaultAuth authentication domain.

The metadata identity that represents Tara O'Toole owns two logins:

- The first login is used by the metadata server to determine Tara's identity. This login is now also used to enable the metadata server to authenticate Tara on behalf of the Xythos server. This use requires the first login to be assigned to the DefaultAuth authentication domain.

* This is the default configuration for authentication to a Xythos WebFile Server. More information and configuration details are provided in "Accessing a Xythos WebFile Server" on page 31.

- □ The second login provides access to the stored process and workspace servers that are registered in the MVSAuth authentication domain. This login functions as an outbound login (it is outbound from the metadata server), so this login includes a password to support a single sign-on approach to additional authentication.

Note: A different set of logins might be required if your metadata server uses an alternative authentication provider or your deployment includes pooled servers. △

Tara does not directly own a login that provides access to the server in the OracleAuth authentication domain, so she can access that server only if she is a member of a user group that owns an appropriate login. In this example, Tara is a member of the ETL Developers user group, so she can use that group's shared login to get to the Oracle server in the OracleAuth authentication domain. If you give the ETL Developers group a login for the OracleAuth authentication domain, you should not also give Tara a login for the OracleAuth authentication domain. If more than one login for a particular authentication domain is available to Tara, then a requesting application might not be able to determine which set of credentials to use.

Note: In order to access the Oracle server from SAS Data Integration Studio, Tara must be able to access both the workspace server *and* the Oracle server. △

A Closer Look: Accessing SAS Servers

Introduction to SAS Server Access Examples

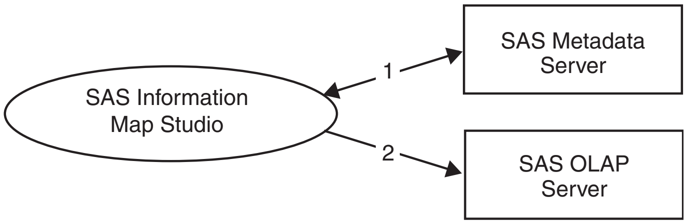
This section contains specific examples of additional authentication from various applications to SAS OLAP Servers, SAS Workspace Servers, and SAS Stored Process Servers. The examples assume these conditions:

- □ The deployment includes the standard, required accounts that are described in the pre-installation checklist.
- □ The user has completed initial authentication.
- □ The user has a metadata identity.
- □ The logins that the user needs for additional authentication are defined in the metadata repository.
- □ The accounts that the user needs have been established with the appropriate authentication providers.
- □ Each SAS OLAP Server, SAS Workspace Server, and SAS Stored Process Server is registered in the metadata and is associated with an appropriate authentication domain.

Accessing a SAS OLAP Server

This example describes the additional authentication process from SAS Information Map Studio to a SAS OLAP Server. The process is initiated when a user makes a request to access cubes from SAS Information Map Studio. The process is depicted in the following figure.

Figure 1.12 Additional Authentication to a SAS OLAP Server



The prerequisites for accessing a SAS OLAP Server are that the metadata server and the OLAP server must be running (the metadata server must be started before the OLAP server).

The numbers in the diagram correspond to these activities:

1 SAS Information Map Studio goes to the metadata server to get the user's credentials for the SAS OLAP Server. As the requesting client, the user must have ReadMetadata permission to the SAS OLAP Server definition. The user (or a group to which the user belongs) must have a login for the authentication domain that is associated with the SAS OLAP Server definition. The user ID and password in that login must correspond to an account that has been established with the SAS OLAP Server's authentication provider.

Note: If the application can use the user's cached credentials to access the SAS OLAP Server, then this step is omitted. △

2 SAS Information Map Studio provides the user's credentials to the SAS OLAP Server. The SAS OLAP Server then authenticates the user against its authentication provider.

Accessing a SAS Workspace Server

This example describes the additional authentication process from SAS Web Report Studio to a SAS Workspace Server. The process is initiated when a user makes a request that requires access to a workspace server from SAS Web Report Studio.

Note: In this example, the SAS Workspace Server is not part of a pool. The next example describes the process for accessing a pooled workspace server. △

These are the prerequisites for accessing a workspace server:

□ The metadata server must be running.

□ The object spawner must be running and must have been started after the metadata server was started.

□ When it initializes, the object spawner must be able to get information about the workspace server from the metadata server. To get this information, the object spawner connects to the metadata server as the SAS Trusted User (which corresponds to the sastrust account on the metadata server).*

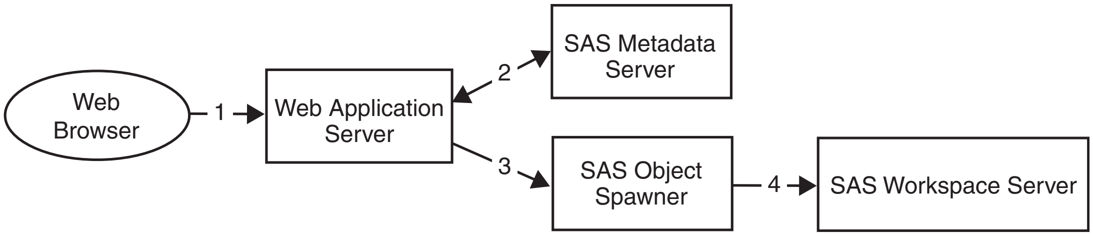
By default, the SAS Trusted User can see the workspace server definition because sastrust is a member of the SAS System Services user group, which has ReadMetadata access to the repository. As you set access controls, you must

* As explained in "Minimizing the Availability of Accounts" on page 76, this connection does not make use of any *trusted user* functionality.

ensure that the SAS System Services group does not lose its ReadMetadata access to the workspace server definition.

The following diagram depicts the process that is initiated by the user's request.

Figure 1.13 Additional Authentication to a SAS Workspace Server



The numbers in the figure correspond to these steps:

1 The user's Web browser sends the request to the SAS Web Report Studio application.

2 The application goes to the metadata server to get the user's credentials for the workspace server. The application must find a login that is associated with the workspace server's authentication domain and is owned by the user (or by a user group to which the user belongs).

Note: If the application can use the user's cached credentials to access the workspace server, then this step is omitted. △

3 The application asks the object spawner to launch a workspace server, using the user's credentials.

4 The object spawner uses the credentials that were obtained from the metadata server to authenticate the user (using host authentication). The object spawner then launches a workspace server for the user.

Accessing a Pooled SAS Workspace Server

This example describes the additional authentication process from SAS Web Report Studio to a pooled SAS Workspace Server.

When you set up pooling, you assign one login to each puddle within the pooled logical workspace server. Each puddle login corresponds to an account that has been established in the host environment of the workspace server. When an application asks the object spawner to launch an additional physical workspace server into the pool, the application must provide the user ID and password for one of the puddle logins. Before the object spawner launches the physical workspace server, the object spawner checks those credentials against the host operating system.

If a user makes a request that requires access to the pooled workspace server, the request does not trigger any further authentication. For this reason, a user does not have to have a host account in order to access a pooled workspace server. A user does, however, have to be a member of at least one user group that is associated with at least one puddle in the pool of workspace servers (as this example explains).

These are the prerequisites for accessing a pooled workspace server:

□ The metadata server must be running.

□ The object spawner must be running and must have been started after the metadata server was started.

- □ When it initializes, the object spawner must be able to get information about the workspace server from the metadata server. To get this information, the object spawner connects to the metadata server as the SAS Trusted User. As explained in the previous example, the SAS Trusted User can see the workspace server definition because sastrust is a member of the SAS System Services user group, which has ReadMetadata access to the repository.
- □ The requesting application must be able to obtain all of the puddle logins from the metadata repository. This enables the requesting application to provide the object spawner with the credentials that the object spawner will use to launch the workspace servers. The account that the requesting application uses to retrieve the puddle logins from the metadata repository is called the pool administrator.
- □ The logical workspace server must be configured for pooling.

The following figures depict the process that is initiated by a user's request.

Figure 1.14 Accessing an Existing Pooled SAS Workspace Server

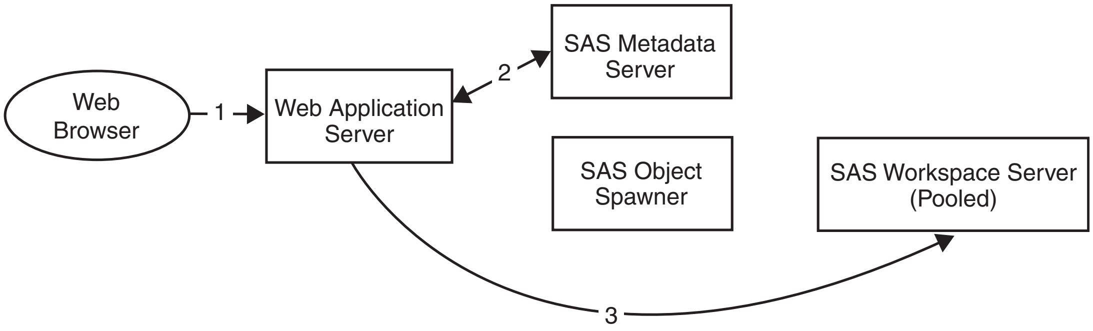
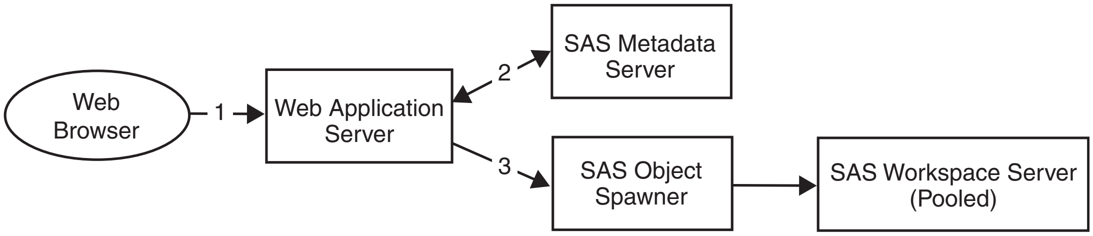


Figure 1.15 Accessing a Newly Launched Pooled SAS Workspace Server



The numbers in the figures correspond to these steps:

1 The user's Web browser sends the request to the SAS Web Report Studio application.

2 SAS Web Report Studio checks the user's group membership information in the metadata repository in order to determine which puddles the user is allowed to use.

Note: In the metadata, each puddle is assigned to one user group. If a user does not belong to any user groups that are assigned to a puddle, then the user will not be able to connect to a workspace server. △

3 SAS Web Report Studio performs one of the following actions:

- □ If there is an available workspace server in a puddle that the user is allowed to use, then SAS Web Report Studio sends the request to that workspace server.

- □ If there are no available workspace servers in any of the puddles that the user is allowed to use, then SAS Web Report Studio asks the object spawner to launch a new workspace server in an appropriate puddle.

Accessing a SAS Stored Process Server

This example describes the additional authentication process from the SAS Add-In for Microsoft Office to a SAS Stored Process Server. The process is initiated when a user makes a request that requires access to a stored process server from the SAS Add-In for Microsoft Office.

These are the prerequisites for accessing a stored process server:

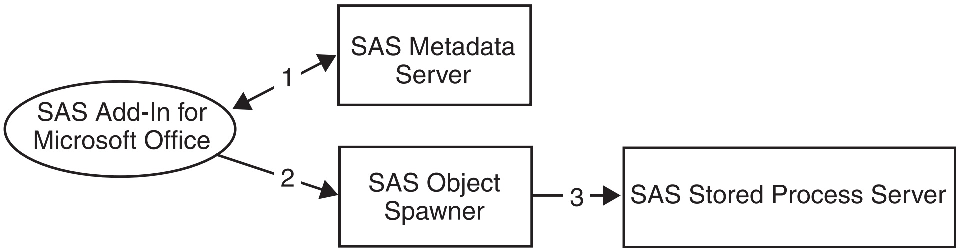
- □ The metadata server must be running.
- □ The object spawner must be running and must have been started after the metadata server was started.
- □ When it initializes, the object spawner must be able to get information about the stored process server from the metadata server. To get this information, the object spawner connects to the metadata server as the SAS Trusted User (sastrust). This user must be able to see the stored process server definition *and* to use the sassrv login (under which the stored process server runs).

  1. By default, the SAS Trusted User can see the stored process server definition because sastrust is a member of the SAS System Services user group, which has ReadMetadata permission for the repository. As you set access controls, you must ensure that the SAS System Services group does not lose its ReadMetadata access to the stored process server definition. To learn how to manage access to server definitions, see "Access Requirements for Server Definitions" on page 129.
  2. The SAS Trusted User can use the sassrv login because sastrust is a member of the SAS General Servers user group, which owns the sassrv login.

     Note: Only members of the SAS General Servers group can use the sassrv login. An *unrestricted user* such as the SAS Administrator (which corresponds to the sasadm account on the metadata server) cannot obtain any passwords, so you should not use the sasadm account in place of the sastrust account. △

The following figure depicts the process that is initiated by a user's request.

Figure 1.16 Additional Authentication to a SAS Stored Process Server



The numbers in the figure correspond to these steps:

1. SAS Add-In for Microsoft Office goes to the metadata server to get the user's credentials for the stored process server. The application must find a login that is associated with the stored process server's authentication domain and is owned by the user (or by a user group to which the user belongs).

Note: If the application can use the user's cached credentials to access the stored process server, then this step is omitted. △

2 The application asks the object spawner for a stored process server.

3 The object spawner either uses an existing stored process server or launches a new one. The stored process server uses host authentication to verify the user's identity and then runs under the sassrv account.

A Closer Look: Accessing Third-Party Servers

Introduction to Third-Party Server Access Examples

This section contains specific examples of additional authentication from various applications to third-party database servers and WebDAV servers. The examples assume these conditions:

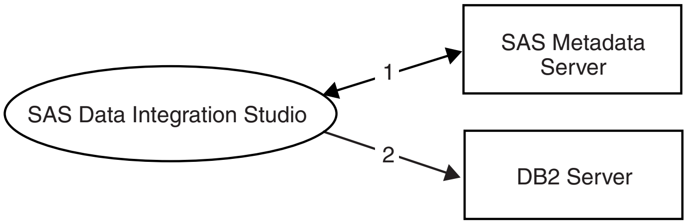
- □ The deployment uses the standard, required accounts that are described in the pre-installation checklist.
- □ The user has completed initial authentication
- □ The user has a metadata identity.
- □ The logins that the user needs for additional authentication are defined in the metadata repository.
- □ The accounts that the user needs have been established with the appropriate authentication providers.
- □ Each third-party server is registered in the metadata and is associated with an appropriate authentication domain.

Accessing a DB2 Database

This example describes the additional authentication process from SAS Data Integration Studio to a DB2 database, using the SAS/ACCESS Interface to DB2.

The process is initiated when a user makes a request to access DB2 data from SAS Data Integration Studio. The following figure depicts the process.

Figure 1.17 Additional Authentication to a DB2 Database



The numbers in the figure correspond to the following activities:

1 SAS Data Integration Studio goes to the metadata server to get the user's credentials for the DB2 system. As the requesting client, the user must have

ReadMetadata access to the DB2 server definition. The user (or a group to which the user belongs) must have a login for the authentication domain that is associated with the DB2 server definition. The user ID and password in that login must correspond to an account that has been established with the DB2 server.

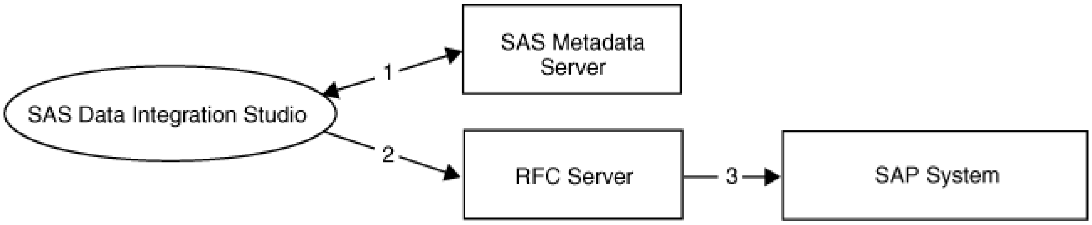
2 SAS Data Integration Studio provides the user's DB2 credentials to the DB2 server. The DB2 server verifies that those credentials correspond to an existing DB2 account.

Accessing an SAP System

This example describes the additional authentication process from SAS Data Integration Studio to an SAP system, using the SAS Data Surveyor for SAP.

The process is initiated when a user makes a request to access SAP data from SAS Data Integration Studio. The following figure depicts the process.

Figure 1.18 Additional Authentication to an SAP System



The numbers in the figure correspond to the following activities:

1 SAS Data Integration Studio goes to the metadata server to get the user's credentials for the SAP system. As the requesting client, the user must have ReadMetadata access to the SAP server definition. The user (or a group to which the user belongs) must have a login for the authentication domain that is associated with the SAP server definition. The user ID and password in that login must correspond to an account that has been established with the SAP system.

2 SAS Data Integration Studio provides the user's SAP credentials to the Remote Function Call (RFC) server.

3 The RFC server passes the SAP credentials to the SAP system, which verifies that those credentials correspond to an existing account on the SAP system.

Accessing a Xythos WebFile Server

The process for accessing a Xythos WebFile Server differs from the process for accessing other servers in some important ways. For this reason, before presenting an example of how this process works, this topic explains how user credentials for Xythos are acquired and verified.

The first step in the authentication process is for the application (the SAS Information Delivery Portal in this example) to acquire the requesting user's credentials.

□ In the default configuration, the SAS Information Delivery Portal acquires the requesting user's credentials for the Xythos server by caching the credentials that the user supplied when logging on.

□ In an alternate configuration, the SAS Information Delivery Portal retrieves the requesting user's credentials for the Xythos server from the metadata repository.

The SAS Information Delivery Portal determines the authentication domain for the Xythos server by checking a configuration file, rather than by examining metadata that describes that server.

The second step in the authentication process is for the target server (the Xythos WebFile Server) to verify the acquired credentials against its authentication provider.

- □ In the default configuration, the SAS User Customization for Xythos WFS uses the SAS Metadata Server as its authentication provider (this enables you to avoid maintaining an additional store of user information in the Xythos WebFile Server). In this process, the metadata server uses its authentication provider to verify the acquired credentials.

  After the authentication provider of the metadata server verifies the requesting user's credentials, the SAS User Customization for Xythos WFS must locate a login that is owned by the requesting user and associated with the authentication domain of the Xythos server. If no such login exists, then the user cannot connect to the Xythos server. In the default configuration, the Xythos server is associated with the DefaultAuth authentication domain.

- □ In an alternate configuration, the SAS User Customization for Xythos WFS first retrieves (from the metadata server) the requesting user's login for the authentication domain with which the Xythos server is associated. The SAS User Customization for Xythos WFS then authenticates the requesting user by determining whether the password in the retrieved login is the same as the password that was provided by the connecting client (the SAS Information Delivery Portal in this example). This process requires that the requesting user's login for the authentication domain of the Xythos server includes a password.

The following example explains the configuration details and illustrates the additional authentication process from the SAS Information Delivery Portal to a Xythos WebFile Server. The example assumes that you are using the default configuration, which includes these settings:

- □ In your SAS Information Delivery Portal configuration file, the authentication domain of the WebDAV server is the same as the cached credentials authentication domain. The **install.properties** file in the **PortalConfigure** directory includes these lines:

  ```
  $DAV_DOMAIN$=DefaultAuth
  $SERVICES_OMI_DOMAIN$=DefaultAuth
  ```

- □ In your SAS User Customization for Xythos WFS configuration file, the authentication domain of both the Xythos server and the metadata server is DefaultAuth. The **saswfs.properties** file in the **wfs-4.0.48** directory includes these lines:

  ```
  com.sas.wfs.domain.dav=DefaultAuth
  com.sas.wfs.domain.metadata=DefaultAuth
  ```

- □ The Xythos WebFile Server is *not* configured to force DIGEST HTTP authentication.

  Note: By default, the SAS User Customization for Xythos WFS configures Xythos to use BASIC authentication. This is the preferred configuration. △

These configuration files are created for you based on values that you supply during installation of the SAS Information Delivery Portal and the SAS User Customization for Xythos WFS. The following tables document how the values that you supply during installation correspond to the variables in the configuration files.

Table 1.4 Installation of the SAS Information Delivery Portal

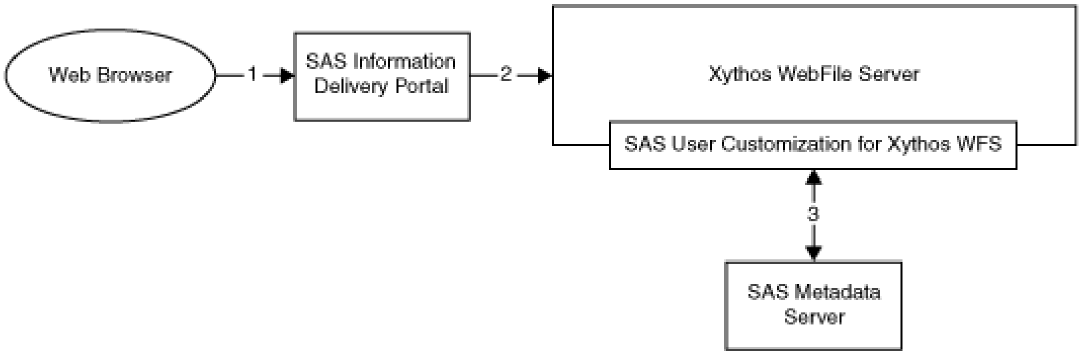
Value That You Supply during Installation	Line That Is Generated in the PortalConfigure\install.properties file
Enter the authentication domain for the SAS Metadata Server > *DefaultAuth*	$SERVICES_OMI_DOMAIN$=DefaultAuth
Enter the authentication domain for the WebDAV Server >*DefaultAuth*	DAV_DOMAIN=DefaultAuth

Table 1.5 Installation of the SAS User Customization for Xythos WFS

Value That You Supply during Installation	Line That Is Generated in the wfs-4.0.48\saswfs.properties file
Enter the authentication domain for the SAS Metadata Server > *DefaultAuth*	com.sas.wfs.domain.metadata=DefaultAuth
Enter the authentication domain for the Xythos WFS WebDAV Server >*DefaultAuth*	com.sas.wfs.domain.dav=DefaultAuth

The following figure depicts the process that is initiated when a user makes a request to access a resource that is stored on the Xythos WebFile Server.

Figure 1.19 Additional Authentication to a Xythos WebFile Server


Web Browser
1
SAS Information Delivery Portal
2
Xythos WebFile Server
SAS User Customization for Xythos WFS
3
SAS Metadata Server

The numbers in the figure correspond to the following activities:

1 From a Web browser, the user makes a request to the SAS Information Delivery Portal for a resource that is stored in a WebDAV area on a Xythos WebFile Server.

2 The SAS Information Delivery Portal sends the user's credentials to Xythos for authentication. In this example, the default configuration is used, so cached credentials are used.

Note: If the `PortalConfigure\install.properties` file does not assign both the Xythos server and the metadata server to the same authentication domain, then cached credentials are not used. Instead, the SAS Information Delivery Portal gets the user's credentials for the Xythos server from the metadata server. In this process, the SAS Information Delivery Portal searches the metadata repository for credentials that are associated with the authentication domain that is specified in the DAV_DOMAIN= setting in the SAS Information Delivery Portal's `install.properties` file. △

3 The SAS User Customization for Xythos WFS sends the user's credentials to the metadata server for authentication. In this example, the default configuration is used, so the metadata server's authentication provider verifies the user's identity. After the credentials are verified, the SAS User Customization for Xythos WFS verifies that the user has a login for the authentication domain of the Xythos server. In this example, the default configuration is used, so the Xythos server is associated with the DefaultAuth authentication domain.

Note: If the **`wfs-4.0.48\saswfs.properties`** file does not assign the Xythos server and the metadata server to the same authentication domain, or if the Xythos server is configured to force DIGEST authentication, then the user's identity is not verified by the metadata server's authentication provider. Instead, the credentials are retrieved from the metadata repository and verified against the credentials that are provided by the SAS Information Delivery Portal. This requires that the user's login for the authentication domain of the Xythos server includes a password. △

CHAPTER 2

Understanding Authorization

Authorization Overview

Introduction to Authorization

Authorization is the process of determining which users have which permissions for which resources. The outcome of the authorization process is an authorization decision that permits or denies a specific action on a specific resource, based on the requesting user's identity and group memberships.

The SAS Intelligence Platform includes an authorization layer that consists of access controls that are stored in a metadata repository and managed by the metadata server's authorization facility. You can use these controls to manage access to metadata and, in some cases, to the computing resources that the metadata represents. The metadata authorization layer has these characteristics:

- □ Management of access controls is centralized. Metadata access controls are defined, stored, and evaluated by the authorization facility of a SAS Metadata Server.
- □ A graphical user interface for setting access controls is provided in SAS Management Console.
- □ Eight standard permissions are available. These permissions are described in the following topic.

Which Actions are Controlled by Each Permission?

The following table explains which actions are controlled by each of the standard permissions.

Table 2.1 Permissions

Permission (Abbreviation)	**Actions Controlled**
ReadMetadata (RM)	Reading a metadata object. For example, a user must have ReadMetadata permission to a SAS library definition in order to see that library in SAS Information Map Studio or SAS Data Integration Studio.
WriteMetadata (WM)	Creating, updating, or deleting a metadata object. For example, a user must have WriteMetadata permission to a repository in order to add a new metadata object (such as an information map or a server definition) to that repository.
CheckInMetadata (CheckInM)	Checking in metadata from a project repository, and checking out metadata to a project repository. This permission is applicable only to SAS Data Integration Studio users who are working in a change-managed environment.
Read (R)	Reading data from the resource that is described by a metadata object. For example, on an OLAP cube, the Read permission controls viewing of the data within the cube. The Read permission is enforced only if the SAS OLAP Server or the SAS metadata LIBNAME engine is used to access the data, or the data is accessed through an information map.
Write (W)	Updating data in the resource that is described by a metadata object. For example, on a table, the Write permission controls updating the rows in the table.*
Create (C)	Adding data to the resource that is described by a metadata object. For example, on a table, the Create permission controls adding rows to the table.*
Delete (D)	Deleting data from the resource that is described by a metadata object. For example, on a library, the Delete permission controls deletion of tables from the library.*
Administer (A)	Accessing the administrative interfaces of SAS servers such as the SAS OLAP Server, the SAS Stored Processes Server, and IOM spawners. For example, a user must have the Administer permission on the application server in order to perform OLAP server administration tasks such as viewing sessions, terminating sessions, and refreshing cubes.

* The Create, Write, and Delete permissions are enforced only if the SAS metadata LIBNAME engine is used to access the data.

The metadata server requests and enforces authorization decisions for the ReadMetadata, WriteMetadata, and CheckInMetadata permissions. The other permissions are not enforced by the metadata server, but can be enforced by applications.

CAUTION:

Because not all applications enforce the Read, Write, Create, and Delete permissions, these permissions are not always sufficient to control access. For example, even if PersonA is denied Read access to DataSet1, PersonA *can* view the data in DataSet1 if PersonA is using SAS Data Integration Studio. SAS Data Integration Studio does not enforce the Read permission when accessing data sets. You can prevent PersonA from seeing DataSet1 by denying PersonA the ReadMetadata permission for the data set. You should also use another authorization layer (such as the data source authorization layer or the operating system authorization layer) to protect the data. △

Related Topics:

Chapter 8, "Using the Metadata Authorization Layer," on page 107

How are Authorization Decisions Made?

When a user requests access to a resource, an authorization decision is made based on an evaluation of all of the relevant access controls. If you want to understand how the authorization facility will evaluate a permission, you must consider both of these questions:

Where is the permission set?
You can set a permission on the resource that you want to protect or on a resource that is a parent to the resource that you want to protect. For example, you can deny ReadMetadata permission directly on a report, or you can set the denial on the folder in which the report is stored. The report inherits effective permissions from the folder.

To whom is the permission assigned?
You can assign a permission to a specific user or to a group to which the user belongs. For example, you can deny ReadMetadata permission directly to the SAS Demo User, or you can assign the denial to a group to which the SAS Demo User belongs. The permissions that you assign to a group are applicable to the members of that group.

The answers to these questions are incorporated into the access control evaluation process, with the *Where?* question having priority over the *To whom?* question. This is a high-level explanation; the rest of this chapter provides detailed information about how permissions can be set and how authorization decisions are made.

Related Topics:

"Where Can Permissions Be Set?" on page 38

"To Whom Can Permissions Be Assigned?" on page 45

"A Closer Look: How Authorization Decisions are Made" on page 46

Authorization Terminology

The following terms are important to understanding how authorization works in the SAS Intelligence Platform:

access control
a grant or denial of a particular permission for a particular resource to a particular user or group. For example, an access control can consist of a denial of the WriteMetadata permission for a particular information map to the PUBLIC user group.

authorization layer
a set of access controls that exists within a particular security framework, such as an operating system or a database management system. Authorization layers that can affect access to resources in a SAS Intelligence Platform environment include the following:

- the metadata authorization layer, which is part of the SAS Intelligence Platform
- the operating system authorization layer, which consists of the file, directory, and system permissions that you specify on a particular machine
- the data source authorization layer, which consists of permissions to relational database objects, passwords for SAS data sets, and other access controls specific to data sources
- the WebDAV authorization layer, which consists of the third-party Web server access controls on report content objects such as files and directories
- the physical authorization layer, which consists of tangible protective measures such as locking a server room or cabinet

A user's ability to perform a particular action is determined not only by metadata layer access controls but also by external authorization mechanisms such as operating system permissions and database controls. In order to perform a particular action, a user must have the necessary permissions in *all* of the applicable authorization layers.

For example, regardless of the access controls that have been defined for a user in the metadata repository, that user cannot access a particular file if the operating system permissions do not permit the action.

effective permissions
a calculation of the access that a user actually has to a particular object. For each authorization layer, the calculation determines the net effect of all applicable access controls. A user's effective permissions to resources are limited to access that is permitted by all authorization layers.

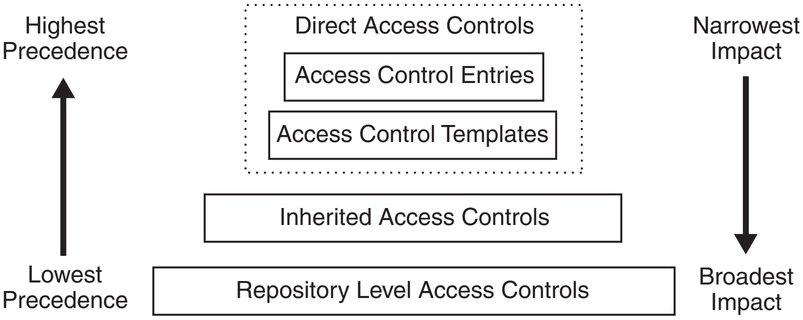
identity hierarchy
a ranking that is based on a user's metadata identity and group memberships.

Where Can Permissions Be Set?

Direct, Inherited, and Repository-Level Access Controls

The following figure introduces the possible answers to the *Where?* question by depicting the relative priority and specificity of direct, inherited, and repository-level access controls.

Figure 2.1 Access Controls in the Metadata Authorization Layer



From top to bottom, the access controls in the figure are ordered as follows:

- □ from highest precedence (hardest to override) to lowest precedence (easiest to override).
- □ from narrowest impact (most specific) to broadest impact (least specific). For example, a repository level access control can affect all of the resources in a repository, while an access control entry can be directly assigned to only one resource.

Note that the *Where?* question is a relative question that depends on the resource whose access controls you are examining. For example, a permission setting that is a direct access control for one resource (such as a report folder) can be the source of an inherited permission for another resource (such as a report within that folder).

Direct Access Controls

The direct access controls for a resource consist of the grants and denials of permissions that are set on the **Authorization** tab of a particular resource. There are two types of direct access controls—access control entries (ACEs) and access control templates (ACTs).

- □ An ACE is an access control that is specifically tied to a particular resource. For example, an ACE can consist of a grant to a user of ReadMetadata permission for a particular table. On the **Authorization** tab, permissions that come from directly assigned ACEs have no added background color.
- □ An ACT is a reusable, named pattern of identities and permissions. For example, you can create an ACT named AdminOnlyAccess that consists of a denial of WriteMetadata permission for the PUBLIC group and a grant of all permissions for an Administrators user group. You can then apply that ACT to all objects that should have those particular protections. On the **Authorization** tab, permissions that come from directly assigned ACTs have a green background color.

  Note: Using ACTs rather than individual ACEs centralizes some of your access control management, because an ACT can be updated independently of the resources to which it has been applied. So if you change your mind about how you want to handle this type of access, you can make changes on the AdminOnlyAccess ACT, without having to revisit every resource to which you applied that ACT. △

Inherited Access Controls

Introduction to Inherited Access Controls

The inherited access controls for a resource consist of the effective permissions on each of the resource's immediate parents. For example, a library is an immediate parent to all of the tables that are assigned to it, and the server to which the library is assigned is an immediate parent to the library.

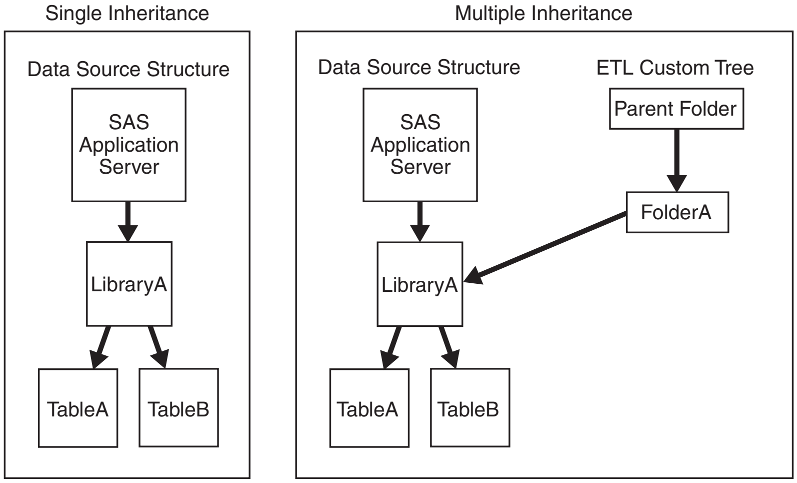
The effective permissions that each parent resource conveys represent the *net effect* of all of the parent's direct, inherited, and repository-level controls. For example, if the net effect of all of the access controls on LibraryA yields a grant of ReadMetadata permission to a user, then every table that is assigned to LibraryA has an inherited grant of ReadMetadata permission to that user.

Because the effective permissions on the parent encompass all access controls up to the level of the repository itself, every parent will convey either a grant or denial of every permission for every user. However, one resource can convey effective permissions to another resource only if both resources are registered in the same metadata repository. Even when there is a dependency relationship between two repositories, resources in one repository cannot convey effective permissions to resources in another repository.

Multiple Inheritance of Access Controls

The SAS metadata environment supports multiple inheritance, which means that one resource can have more than one immediate parent. The following figure depicts examples of single inheritance and multiple inheritance in the SAS metadata environment.

Figure 2.2 Single and Multiple Inheritance



In the single inheritance figure, TableA and TableB share the same immediate parent, LibraryA. Both tables have inherited access controls that are conveyed from the effective permissions on LibraryA. LibraryA has inherited access controls that come from the effective permissions on the SAS Application Server. For example, if the net effect of the applicable access controls on LibraryA yields a grant of ReadMetadata permission to a user, then both TableA and TableB have inherited access controls that grant ReadMetadata permission to that user.

Note: You can override an inherited access control by adding a direct access control. For example, If you set a permission on TableA that denies the user the ReadMetadata permission, then that direct access control on TableA will override the conflicting grant that LibraryA conveys to TableA. In this example, the authorization decision process will not consider the inherited access control because of the presence of the direct access control. △

In the multiple inheritance figure, LibraryA is assigned to both a SAS Application Server and to a folder in the ETL custom tree. The inherited access controls for LibraryA consist of the effective permissions on both of these immediate parents. For example, if the net effect of the access controls on the SAS Application Server is a grant of ReadMetadata permission to a user, and the net effect of the access controls on FolderA is a denial of the same permission to that user, then LibraryA has both an inherited grant and an inherited denial of this permission for that user. In this circumstance, the user can access LibraryA, because a grant that is conveyed by *any* parent object is sufficient.

Access Control Inheritance Rules

In a SAS metadata environment, inheritance rules determine which objects can be parents to which other objects. For example, there is an inheritance rule that specifies that a SAS library definition is a parent to the SAS data set definitions within that library. The inheritance rules establish containment structures through which resources inherit access controls. In each structure, you can set access controls at varying levels of granularity. For example, you can deny someone ReadMetadata access to an entire library of SAS data sets, or to a particular data set, or to a particular variable within a data set. The main resource-containment structures that are established by inheritance rules are described in the following sections.

Inheritance in SAS Data

In a SAS metadata environment, effective permissions for SAS data flow as follows:

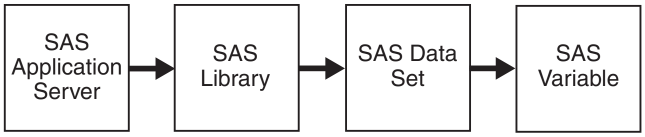
□ from a SAS Application Server to the SAS libraries that are defined on the logical workspace server component of the SAS Application Server

□ from a SAS library to the data sets within that library

□ from a data set to the variables within that data set

The following figure depicts the inheritance flow for SAS data in the metadata environment.

Figure 2.3 Inheritance Flow for SAS Data



Not all permissions are supported at all levels for SAS data.

□ The metadata server enforces the ReadMetadata, WriteMetadata, and CheckInMetadata permissions at all levels—server, library, data set, and variable.

- □ The metadata LIBNAME engine enforces the Read, Write, Create, and Delete permissions at the library and data set levels only. Setting these permissions on a variable has no effect.

Note: SAS data objects can also inherit effective permissions from custom trees. △

Inheritance in Relational Database Data

In a SAS metadata environment, effective permissions for relational database data flow as follows:

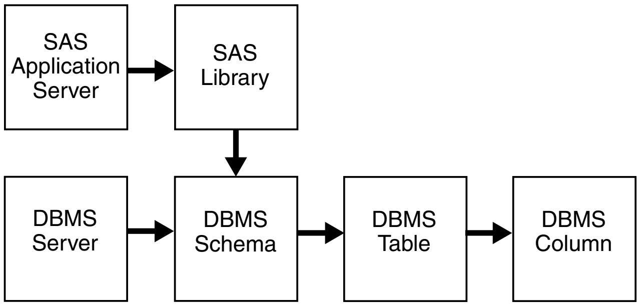
- □ from a database management system (DBMS) server definition to the DBMS schemas that are defined on that DBMS server

  Note: A DBMS schema that is associated with a SAS library will also inherit effective permissions from that library and, in turn, from the SAS Application Server that includes the workspace server component on which the library is defined. △

- □ from a DBMS schema to the tables within that schema
- □ from a DBMS table to the columns within that table

The following figure depicts the inheritance flow for database data in the metadata environment.

Figure 2.4 Inheritance Flow for Relational Database Data



Not all permissions are supported at all levels for relational database data.

- □ The metadata server enforces the ReadMetadata, WriteMetadata, and CheckInMetadata permissions at all levels — server, schema, table, and column.
- □ The metadata LIBNAME engine enforces the Read, Write, Create, and Delete permissions at the library and table levels only. Setting these permissions on a column has no effect.

Note: Relational database objects can also inherit permissions from custom trees. △

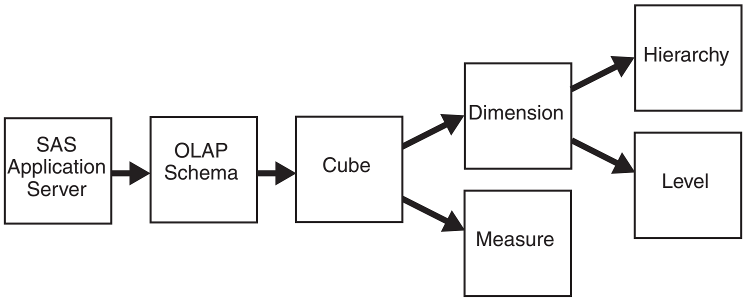
Inheritance in OLAP Data

In a SAS metadata environment, effective permissions for OLAP data flow as follows:

- □ from a SAS Application Server to the OLAP schemas that are defined on the OLAP server component within the SAS Application Server
- □ from a schema to the cubes within that schema
- □ from a cube to the dimensions and measures within the cube
- □ from a dimension to the hierarchies and levels within the dimension

The following figure depicts the inheritance flow for OLAP data in the metadata environment.

Figure 2.5 Inheritance Flow for OLAP Data



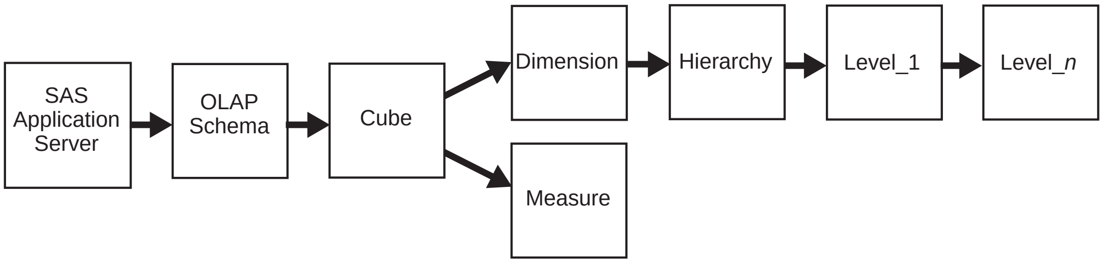
Not all permissions are relevant for all OLAP objects. In order to access a cube, a user must have both ReadMetadata and Read permission for the cube. However, in order to access a dimension, measure, hierarchy, or level, only Read permission is required, because the SAS OLAP Server uses the Read permission to request and enforce decisions for these objects.

A user's ability to access OLAP data is also affected by the requirements for drilling through a cube in order to access the data. If a user does not have Read access to a particular object (such as an OLAP cube), then that user cannot access other objects (such as dimensions and measures) within that object. For example, if a direct access control on an OLAP cube denies the Read permission to a particular user, then that user cannot access any data within the cube. Even if you give the user Read permission to a dimension within the cube, the user will be unable to access that dimension. The problem is not that the user does not have Read access to the dimension. Rather, the problem is that the user does not have the clear path of grants of Read access that is necessary to navigate through the cube to the dimension.

The following list and figure document these navigational access requirements for OLAP data:

- If a user does not have Read permission to a cube, the user cannot navigate to the dimensions and measures within the cube.
- If a user does not have Read permission to a dimension, the user cannot navigate to the hierarchies within the dimension.
- If a user does not have Read permission to a hierarchy, the user cannot 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 cannot navigate to the next level in that structure.

Figure 2.6 Access Requirements for Navigating through OLAP Data

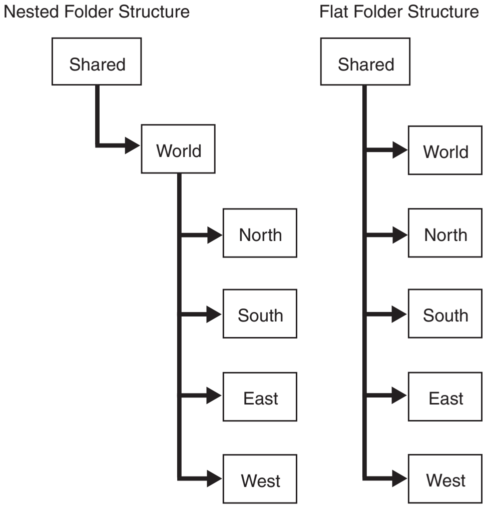


Inheritance in Custom Trees

A SAS Intelligence deployment can include one or more custom trees that you can use to organize and manage access for certain resources. For example, SAS Data Integration Studio enables you to add folders and items to the ETL custom tree. Within a custom tree, each folder inherits the effective permissions of its parent folder. Most items in the ETL custom tree inherit the effective permissions of the folder in which the items are located.

Selecting the optimal folder structure for your custom trees can help you minimize the number of access controls that you have to set and maintain. Within each tree, you can use a flat list structure, a nested tree structure, or a blend of the two structures. For example, the following figures illustrate two of the ways you could structure folders in your ETL custom tree if you are organizing metadata that describes sales data for a worldwide sales team and for four regional sales teams.

Figure 2.7 Custom Folder Structures



The arrows show how the inheritance of effective permissions flows in each custom tree.

- □ In the nested structure, the regions each inherit the effective permissions of the World reports folder, which in turn inherits the effective permissions of the Shared folder.
- □ In the flat structure, the World folder and all of the regional folders inherit the effective permissions of the Shared folder.

Repository-Level Access Controls

The permissions that users have been given to the entire repository are their repository-level access controls. For most resources, you can define access by locating the resource in SAS Management Console and then setting permissions on the resource's **Authorization** tab. However, you do not use this method to define access to an entire repository. Instead, you specify repository-level access controls on the **Users and Permissions** tab of an ACT that has been designated as the repository ACT.

The repository ACT is represented in SAS Management Console by a blue cylinder icon and is named *Default ACT* by default. You can locate the repository ACT under

Environment Management ▶ Authorization Manager ▶ Access Control Templates.

Note: A repository ACT is created for you in each repository. The initial settings on a foundation repository ACT grant ReadMetadata and WriteMetadata permissions to the PUBLIC group. For information about narrowing this access, see "Protecting the Foundation Repository" on page 67. △

Repository-level controls define access to resources for which no specific controls have been set. Repository-level controls also define users' ability to perform the actions that are listed in the following table.

Table 2.2 Actions That Are Controlled by Repository-Level Access Controls

User Action	Required Access to the Repository
Access or view objects in the metadata repository	In many cases, the requesting user must have ReadMetadata access to the repository.
Create a new object (such as an information map or report) anywhere in the metadata repository	The requesting user must have WriteMetadata access to the repository.
Access the SAS Information Delivery Portal (Public Kiosk)	The SAS Guest and the SAS Web Administrator must have ReadMetadata and WriteMetadata access to the repository.
Log on to the SAS Information Delivery Portal	The requesting user must have ReadMetadata and WriteMetadata access to the repository.

As the preceding table indicates, having both ReadMetadata and WriteMetadata access to the repository is a prerequisite to performing many tasks. For example, in order to add a new report to a particular folder, a user must have both of these things:

□ WriteMetadata access to that particular folder

□ WriteMetadata access to the entire repository

To Whom Can Permissions Be Assigned?

The User/Group Identity Hierarchy

The following list introduces the possible answers to the *To Whom?* question by describing the precedence ranking for user and group identities:

1. the user's individual metadata identity.
2. a user-defined group that has the user's metadata identity 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, if the user belongs to a group named ETL_Advanced, and that group is a member of another group called ETL_Basic, then the ETL_Basic group is a second-level group for that user.

   Note: If you have additional levels of nested groups, then each successive group has less precedence than the group or groups that are its members. △

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

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

Examples of Identity Hierarchies

The examples in the following table illustrate how the identity hierarchy works.

Table 2.3 Examples of Identity Hierarchies

User Information	User's Identity Hierarchy
User has no metadata identity.	Primary identity: PUBLIC
User has a metadata 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).	Primary identity: self First-level memberships: GroupA, GroupB Second-level memberships: SASUSERS Third-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 Portal Users group).	Primary identity: self First-level memberships: GroupA, GroupB Second-level memberships: Portal Users Third-level memberships: SASUSERS Fourth-level memberships: PUBLIC

A Closer Look: How Authorization Decisions are Made

The following list describes the authorization decision process:

1 Permissions that are set on the target resource are examined.

- □ Any conflicts that arise from group membership are resolved by the identity hierarchy. For example, a permission that is assigned to a user overrides a conflicting permission that is assigned to a group to which the user belongs.
- □ If there is a conflict between an ACE and an ACT at the same level in the identity hierarchy, then the ACE takes precedence.
- □ If there is a conflict between two ACE's (or two ACT's) at the same level in the identity precedence hierarchy, then the outcome is a denial.
- □ If one or more permission conditions have been defined, then the condition that is assigned at the highest level of identity precedence is applied. Other conditions that also apply to a user because of group memberships do not provide additional, cumulative access (unless there are multiple groups at the highest level of identity precedence).
- □ If there are no pertinent permissions set on the target resource, then the evaluation process continues.

2 The inheritance rules are applied to identify all of the target resource's parent objects. The entire evaluation process is applied to each of the parent objects.

- □ If *any* of the parent objects conveys a grant, then access is granted.
- □ If *all* of the parent objects convey denials, then access is denied.
- □ If the target resource does not have any parent objects, then the evaluation process continues.

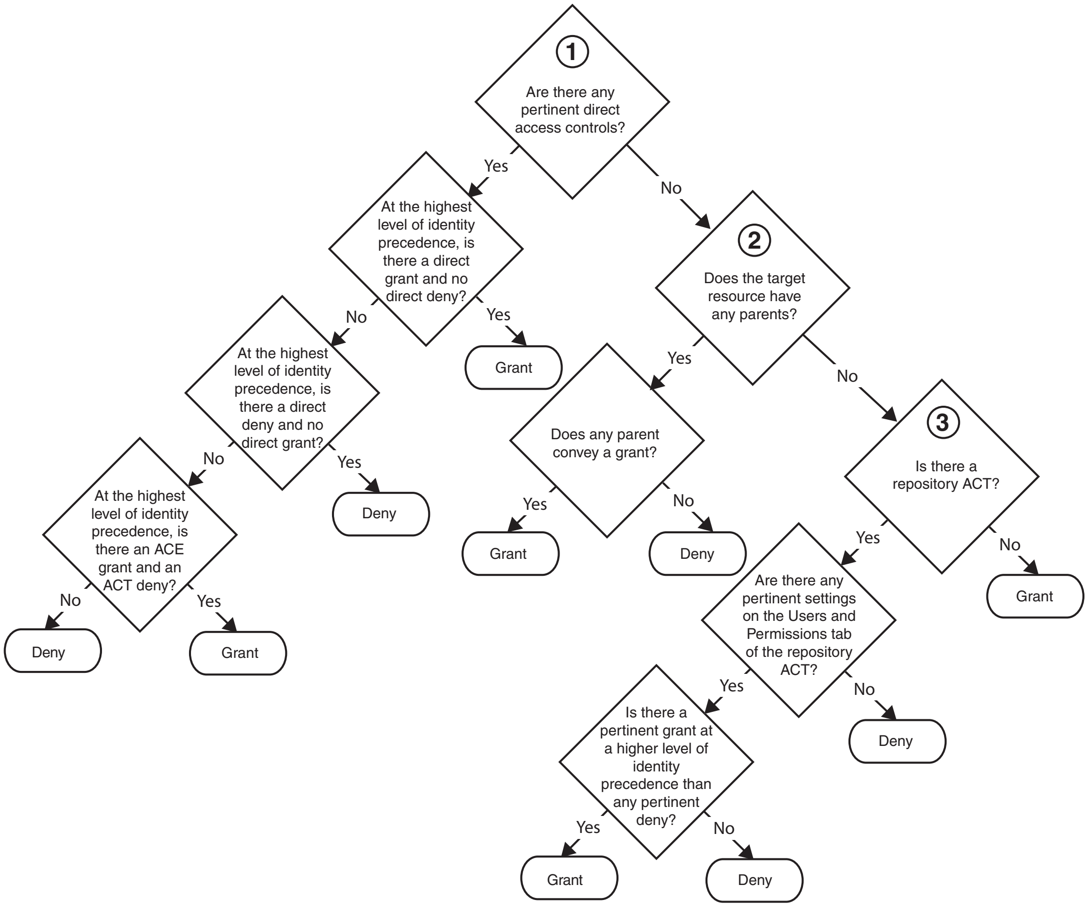
3 The **`Users and Permissions`** tab of the repository ACT is examined. Any conflicts within the repository ACT are resolved by the identity hierarchy.

- □ If the repository ACT grants or denies the requested permission, then that grant or denial is determinative.
- □ If the repository ACT neither grants nor denies the permission, then the permission is denied.

Note: If there is no repository ACT, then the permission is granted. You should always have a designated repository ACT. △

The following flowchart depicts this process.

Figure 2.8 Authorization Decision Process


1
Are there any pertinent direct access controls?
Yes
No
At the highest level of identity precedence, is there a direct grant and no direct deny?
2
Does the target resource have any parents?
No
Yes
Grant
Yes
No
At the highest level of identity precedence, is there a direct deny and no direct grant?
Does any parent convey a grant?
3
Is there a repository ACT?
No
Yes
Deny
Yes
No
Grant
Deny
Yes
No
At the highest level of identity precedence, is there an ACE grant and an ACT deny?
Are there any pertinent settings on the Users and Permissions tab of the repository ACT?
Grant
No
Yes
Deny
Grant
Yes
No
Deny
Is there a pertinent grant at a higher level of identity precedence than any pertinent deny?
Yes
No
Grant
Deny

Related Topics:

"Where Can Permissions Be Set?" on page 38

"To Whom Can Permissions Be Assigned?" on page 45

"Tip: Interpreting the Authorization Tab" on page 126

Summary: Principles of Access Control Precedence

The following table summarizes the precedence principles for metadata layer access controls and presents an example of each principle.

Table 2.4 Principles of Metadata Layer Access Control Precedence

Principle	**Example**	
	Scenario	**Outcome and Explanation**
A direct access control has precedence over access controls that come from parent objects or from the repository ACT.	A direct access control on LibraryA denies ReadMetadata permission to PUBLIC. The repository ACT grants ReadMetadata permission to the user.	The user cannot see LibraryA. The denial to PUBLIC has precedence because it is assigned *directly* on the target resource (LibraryA). Direct access controls always have precedence over inherited controls *regardless of who the permissions are assigned to*.
If there are conflicting direct controls, then the identity hierarchy determines the outcome.	A direct access control on LibraryA denies ReadMetadata permission to PUBLIC. Another direct access control on LibraryA grants ReadMetadata permission to the user.	The user can see LibraryA. This is a conflict between two direct controls, so the identity hierarchy becomes relevant. The user's primary identity has priority over the implicit group PUBLIC, so the direct grant to the user overrides the direct denial to PUBLIC.
If there are conflicting direct controls at the same level in the identity hierarchy, then the type of access control (ACE or ACT) determines the outcome.	A direct ACT on LibraryA denies ReadMetadata permission to GroupA. A direct ACE on LibraryA grants ReadMetadata permission to GroupB. The user is a member of both GroupA and GroupB.	The user can see LibraryA. The conflict is between direct controls that both come from the user's first-level group memberships, so the type of access control becomes relevant. ACEs are given priority over ACTs, so the ACE grant overrides the ACT denial.
If there are conflicting direct controls at the same identity level *and* they are both ACEs (or they are both ACTs), then the outcome is a denial.	A direct ACE on LibraryA denies ReadMetadata permission to GroupA. Another direct ACE on LibraryA grants ReadMetadata permission to GroupB. The user is a member of both GroupA and GroupB.	The user cannot see LibraryA. The direct controls are both assigned at the same level in the identity hierarchy (to the user's first-level groups) and they are both of the same type (ACEs), so the outcome is a denial.

Principle	**Example**	
	Scenario	**Outcome and Explanation**
If there are no relevant direct controls and there is at least one parent object that conveys a grant, then the outcome is a grant.	There are no direct controls on LibraryA. LibraryA is assigned to ServerA, which makes ServerA a parent object to LibraryA. ServerA conveys a grant of ReadMetadata permission to the user.[1]	The user can see LibraryA. In the absence of relevant direct controls on LibraryA, a grant from *any* of LibraryA's parent objects is sufficient to get access. (Even if LibraryA had another parent object that conveyed a denial, the outcome would be a grant. For an illustration of this scenario, see "Multiple Inheritance of Access Controls" on page 40.)
If there are no relevant direct controls and there are no parent objects, then the repository ACT determines the outcome.	There are no direct controls on LibraryA. LibraryA does not have any parent objects. The repository ACT denies ReadMetadata permission to the user.	The user cannot see LibraryA. Because there are no relevant direct access controls and no parent objects, the settings on the **`Users and Permissions`** tab of the repository ACT are determinative.[2]
Conflicts within an ACT's **`Users and Permissions`** tab are resolved by the identity hierarchy ranking.	There are no direct controls on LibraryA. LibraryA does not have any parent objects. The repository ACT denies ReadMetadata permission to PUBLIC. The repository ACT grants ReadMetadata permission to SASUSERS. The user has a metadata identity.	The user can see LibraryA. The user is a member of both PUBLIC and SASUSERS. In the identity hierarchy, SASUSERS has precedence over PUBLIC, so the net effect of the repository ACT settings is to grant ReadMetadata permission to all members of SASUSERS.

[1] The conveyed permission comes from an *effective permission* on ServerA. For this reason, it makes no difference to LibraryA whether the source of the conveyed effective permission was a direct control on ServerA or a control that ServerA inherited from one of its parent objects. It also makes no difference to LibraryA whether the source of the conveyed permission was an access control that was assigned to the user or an access control that was assigned to a group to which the user belongs.

[2] If there is no repository ACT, then the outcome would be a grant. You should always have a designated repository ACT.

CHAPTER

3

Security Planning

Overview of Security Planning

This chapter outlines a planning process for the security aspects of a deployment of the SAS Intelligence Platform. The process consists of these tasks:

1. defining the security goals
2. making preliminary decisions about how applications and servers will authenticate users
3. identifying the user accounts you will need and the metadata identity information that you will create and maintain
4. reviewing physical access considerations for data

Defining the Security Goals

It is important to customize your security design for your site, so you should begin your security planning by analyzing your environment to determine your security needs. Consider these guidelines:

- □ Different types of data require different levels of protection. It is important that your security policies reflect an understanding of your data and of the needs of the users who interact with that data.
- □ You should usually be more conservative in granting the ability to write to metadata objects and computing resources than you are in granting the ability to read those objects.
- □ Even for data that is not highly sensitive, it might be desirable to constrain access so that your users do not see information that is not relevant to their needs.
- □ You should evaluate how likely it is that someone will accidentally or intentionally compromise the security of your deployment, and how severe the consequences of a compromise would be.
- □ You should consider the nature of your user community and their expectations regarding security and privacy.
- □ Your security policy should meet any applicable organization or legal requirements for security and auditability.

In this process, remember that there is no absolute security. You should choose a security design that strikes the right balance between deployability, usability, maintainability, and security for your environment.

Making Preliminary Decisions about Authentication

Choosing Authentication Providers

Your selection of authentication providers can have a significant impact on the amount of identity information that has to be created and maintained for the deployment.

A primary goal in authentication configuration is to minimize the number of places where you have to create and maintain user IDs and passwords. One way to do this is to make the credentials that a user submits as reusable as possible (for authenticating to various servers). Credentials are reusable only to the extent that servers use the same authentication provider. For example, if all of your servers authenticate users against the same host operating system, then you need only one set of user accounts, and you can avoid storing a lot of passwords in the metadata repository. For these reasons, your choice of authentication providers requires careful consideration.

You can use the following table to document your choices of authentication providers. For each component, select an authentication provider and note the platform, Windows domain name, machine name, or other details as appropriate. Not all deployments use all components.

Table 3.1 Authentication Provider Checklist

Component	How Identity of Requesting User Is Validated
SAS Metadata Server	__ Server relies on the host operating system. Includes host-mediated authentication for which the host relies on a back-end user store such as LDAP or Active Directory. __ Server relies directly on LDAP or Active Directory. Requires configuration changes. Can increase credential management work. See "Using LDAP or Active Directory" on page 82.
SAS OLAP Server	__ Server relies on the host operating system. Includes host-mediated authentication for which the host relies on a back-end user store such as LDAP or Active Directory. __ Server relies directly on LDAP or Active Directory. Requires configuration changes. Can increase credential management work. See "Using LDAP or Active Directory" on page 82.
SAS Stored Process Server	__ Server relies on the host operating system. Includes host-mediated authentication for which the host relies on a back-end user store such as LDAP or Active Directory.
SAS Workspace Server	__ Server relies on the host operating system. Includes host-mediated authentication for which the host relies on a back-end user store such as LDAP or Active Directory.
SAS Workspace Server (pooled)	__ Access depends on membership in a puddle group. Requires configuration changes. Available only for Web applications. Can reduce credential management work. See "Accessing a Pooled SAS Workspace Server" on page 27.
Third Party Database Server	__ Server relies on database authentication. Databases often have their own authentication providers. You can still reduce credential management by using shared accounts. See "Example: Managing Authentication to a Database Server" on page 81.
SAS Web applications	__ Applications rely on the metadata server's authentication provider. For an explanation of how this works, see "Initial Authentication on a Metadata Server" on page 12. __ Applications rely on the Web server's authentication provider. Requires configuration changes. Can increase credential management work. See "Using Web Authentication" on page 82.
SAS desktop applications	__ Applications rely on the metadata server's authentication provider. For an explanation of how this works, see "Initial Authentication on a Metadata Server" on page 12.
SAS OLAP Data Provider	__ Component relies on the SAS OLAP Server's authentication provider. For an explanation of how this works, see "Initial Authentication on a SAS OLAP Server" on page 15.

Mapping Authentication Providers to Authentication Domains

In order to support single sign-on, you must create authentication domain objects in the metadata that correspond to your authentication providers. You must have a separate authentication domain for each authentication provider. To be more specific, the requirement is for a separate authentication domain for each distinct set of user accounts. For example, if you are using Windows host authentication against local Windows accounts on different machines, then you will need a separate authentication domain for each machine (because each machine maintains its own list of valid local accounts).

You can use the following table to document the relationships between your authentication providers, authentication domains, and servers.

Table 3.2 Authentication Domain Checklist

Authentication Provider[1]	**Authentication Domain**[2]	**Associated Servers**

[1] Use a separate row for each authentication provider.

[2] For each authentication domain, choose a name that will be meaningful to the administrators who work with user credentials for that authentication provider.

For example, the checklist for a single-machine deployment might look like this:

Table 3.3 Authentication Domain Checklist (example)

Authentication Provider	**Authentication Domain**	**Associated Servers**
Windows host operating system (local accounts on mymachine)	DefaultAuth	Metadata server, stored process server, workspace server, OLAP server

If an Oracle database server is added to the deployment, then the checklist might look like this:

Table 3.4 Authentication Domain Checklist (example)

Authentication Provider	**Authentication Domain**	**Associated Servers**
Windows host operating system (local accounts on mymachine)	DefaultAuth	Metadata server, stored process server, workspace server, OLAP server
Oracle authentication	OracleAuth	Oracle database server

If the stored process server is moved to a UNIX system, then the checklist might look like this:

Table 3.5 Authentication Domain Checklist (example)

Authentication Provider	**Authentication Domain**	**Associated Servers**
Windows host operating system (local accounts on mymachine)	DefaultAuth	Metadata server, workspace server, OLAP server
Oracle authentication	OracleAuth	Oracle database server
UNIX host operating system	UnixAuth	Stored process server

Related Topics:
"How Authentication Domains Are Used" on page 9
"Authentication Scenarios" on page 21
"How to Create an Authentication Domain" on page 79

Determining the Scope of the Identity Management Tasks

Introduction to Planning for Identity Management

During installation, an initial set of operating system accounts and metadata identity information is created. This section helps you anticipate the additional user and group accounts and metadata identity information that you will need when you populate your deployment.

What User Accounts Are Needed?

Overview of Planning for User Accounts

You need user accounts that enable all users to access the servers that they will use. Depending on the choices you made in the preliminary decisions phase, these might be individual or shared accounts. The accounts can be any of the following:

- □ local accounts in the operating system of the computer on which the authenticating server is running
- □ network accounts that provide access to the operating system of the computer on which the authenticating server is running
- □ LDAP or Active Directory accounts (for certain servers and configurations)
- □ a user account with any authentication provider that your Web application server uses (for applications that are configured to use Web authentication)
- □ user accounts for database authentication

Note: Accounts that are used for Windows host authentication must have the **`Log on as Batch`** right in the operating system. △

Accounts That Enable Users to Log On to SAS Applications

Each application's authentication provider must include an account for every user who logs on to that application. For example, in a deployment that includes the SAS Information Delivery Portal, SAS Web Report Studio, SAS Management Console, and SAS Information Map Studio, the credentials that a user submits to log on might be verified as follows:

- □ For the SAS Information Delivery Portal and SAS Web Report Studio, you choose to have the Web application server handle authentication using LDAP as the authentication provider.
- □ For SAS Management Console and SAS Information Map Studio, the metadata server handles authentication and you choose to use the operating system as the authentication provider.

In this example, you need to add the following user accounts to support initial authentication:

- □ an LDAP account for every user who uses the SAS Information Delivery Portal or SAS Web Report Studio
- □ an operating system account on the computer on which the metadata server is running for each user who uses SAS Information Map Studio or SAS Management Console

Accounts That Enable Users to Access Additional Systems

You can use the following analysis to identify the accounts you need to create to enable users to access additional systems (such as workspace servers, stored process servers, and other data servers):

1. Identify groups of servers that use the same authentication provider.
2. For each authentication provider, make a list of users who will access resources on servers that use that authentication provider. If multiple users will use a shared account for a particular authentication provider, include the shared account in the list rather than including the individual users who will use the shared account.
3. For each authentication provider, make a list of the additional accounts that you will need to create.

For example, if the deployment that is described in the previous section includes stored process and workspace servers running on z/OS and a database server running on UNIX, you would have the following additional authentication processes:

- □ z/OS host authentication for the stored process and workspace servers
- □ database authentication for the database server.

In this example, to enable all of your users to authenticate to all servers, you would need these accounts:

- □ a z/OS operating system account for every user (or for each group of users who will share an account)
- □ an account on the database server for every user (or for each group of users who will share an account).

What User Definitions Will Be Created in the Metadata?

In addition to their user accounts, many users must also have a user definition in the metadata repository. Several of the security features of the SAS Intelligence Platform are available only for those users who are registered in the metadata.

You need a user definition for every user who meets any of the following criteria:

- □ requires additional access beyond the access that you will give to the PUBLIC group (the PUBLIC group includes everyone who can access the metadata server).
- □ needs to retrieve credentials from the metadata for single sign-on
- □ uses SAS Information Map Studio or SAS Enterprise Miner. These applications require that every user have a unique metadata identity.
- □ uses SAS Web Report Studio in an optional configuration that requires that every user have a metadata identity.
- □ uses the SAS Information Delivery Portal (beyond the public kiosk).

You should create all of your user definitions in a single foundation repository.

What Group Definitions Will Be Created in the Metadata?

You need group definitions in the metadata for these purposes:

- □ *To simplify the process of establishing and managing access controls for authorization.* Granting access to resources on an individual basis can be cumbersome. After you define user groups, you can assign permissions to groups rather than to individual users.
- □ *To manage credentials for shared accounts.* The user ID and password of the shared account is stored with a group definition so that any member of the group can use those credentials.
- □ *To support pooled workspace servers for SAS Web applications.* You make each puddle within a pooled workspace server available to the members of one group.

You should create all of your group definitions in a single foundation repository.

Related Topics:

"Organizing Users Into Groups" on page 89

"How to Use Shared Accounts" on page 80

"Accessing a Pooled SAS Workspace Server" on page 27

Which User IDs and Passwords Will be Stored in the Metadata?

You need to store the following external user account information in the metadata:

- □ *You must store one user ID as part of each user definition (for the purpose of establishing a metadata identity for the user).* In each user definition, you must store the user ID of the external account with which the user logs on to a SAS application. This stored user ID is not needed for authentication; it is used only to establish a unique metadata identity for the user who logged on. Security features such as authorization decisions are based on each user's metadata identity. Unlike user definitions, group definitions do not have to include user IDs for the purpose of establishing metadata identities.
- □ *You must store additional credentials to the extent necessary to support single sign-on in your environment.* It is never necessary to store passwords in the metadata to enable a user to log on to a SAS application. However, in some circumstances, you do need to store user IDs and passwords in the metadata to enable users to access other systems after they log on. You need to store credentials only for those authentication events for which no other credential management features will work.

  Note: In most deployments, it is necessary to store some passwords in the metadata. You can minimize the need for this by giving careful consideration to your selection of authentication providers and using shared accounts where appropriate. △

Related Topics:

"How Metadata Identities Are Used" on page 6

"How to Store User IDs and Passwords in the Metadata" on page 97

"Additional Authentication" on page 16

"How to Use Shared Accounts" on page 80

Summary: Identity Management Requirements

The scope of identity management work depends on the number of different authentication providers that you use, the applications that you have, and your usage of shared accounts. Most sites will make limited use of shared accounts. To highlight the identity management consequences, the following table summarizes the requirements for two extreme environments:

- □ In the low-security environment, shared accounts are widely used. In this example, the user IDs and passwords for the shared accounts are stored in the group definition for a group that everyone belongs to.
- □ In the high-security environment, every user has an individual account for every server.

Table 3.6 User Planning Summary

Requirement	Low-Security Environment	High-Security Environment
Accounts for logging on to SAS applications	Each user has an account with an authentication provider.	Each user has an account with an authentication provider.
Accounts for accessing additional systems (such as unpooled workspace servers, stored process servers, and other data servers)	For each authentication provider, all users share a single account. The credentials for these shared accounts are stored with a group definition for a group that everyone belongs to.	Each user has an account for each authentication provider. The credentials for these individual accounts are stored with the user definitions.
User definitions in the metadata	Every user has a user definition. This enables you to assign the users to the group definition where the shared credentials are stored.	Every user has a user definition.
Stored user IDs to establish metadata identities	Each user definition includes one user ID.	Each user definition includes one user ID.
Stored user IDs and passwords to support single sign-on	The group definition that everyone belongs to includes one set of credentials for each authentication provider.	For circumstances where neither cached credentials nor interactive prompting can be used, each user definition includes a set of credentials.

Reviewing Physical Access Considerations for Data

Importance of the Physical Protections for Data

Metadata-based access controls provide an additional layer on top of physical security. Some clients, such as SAS Web Report Studio, run in a controlled environment with no ability to directly execute SAS code on a back-end server. Other clients, such as SAS Data Integration Studio and SAS Enterprise Guide, enable power users to create and run SAS programs, potentially bypassing the metadata-based controls. For this reason, it is essential to carefully manage physical access in addition to metadata layer access.

Managing physical access to SAS data sets and relational database management system (RDBMS) data requires an understanding of the behavior of the servers that retrieve the data and an understanding of the methods for assigning libraries. The following topics describe the default behaviors and explains how you can get different results by changing your server configuration or pre-assigning libraries.

Physical Access Considerations for SAS Data

When a standard (unpooled) workspace server retrieves SAS data sets for a requesting user, the requesting user's account is used to fetch the data. In the physical layer, you can give each user appropriate permissions for the data sets.

When a pooled workspace server retrieves SAS data sets for a requesting user, a service account called the puddle account fetches the data. If there is only one puddle, then all users of the pooled workspace server have identical operating system access to the data sets. You can create some access distinctions among users by setting up additional puddles and associating a different user group with each puddle. In the physical layer, you can give each puddle account different permissions for the data sets.

When a stored process server retrieves SAS data sets for a requesting user, a service account fetches the data. If there is only one stored process server, then all users have identical operating system access to the data sets. You can create some access distinctions among users by setting up additional stored process servers. Each stored process server should run under a different account and be available to a particular group of users (you can use the ReadMetadata permission on the server definition to control availability). In the physical layer, you can give each stored process server account different permissions for the data sets.

Physical Access Considerations for RDBMS Data

When a standard (unpooled) workspace server retrieves RDBMS data for a requesting user, the database credentials that are available to the requesting user are used to fetch the data. In the database layer, you can give each user appropriate permissions for the data (or each group if you are using shared accounts on the database server).

When a pooled workspace server retrieves RDBMS data for a requesting user, the database credentials that are available to the requesting user are used to fetch the data. In the database layer, you can give each user different access to the data (or each group if you use shared accounts). If instead you want to grant database layer access to only a privileged account, then preassign the database library using the METAAUTOINIT or autoexec method, and put the database credentials in a login that is available to the puddle account. You can create some access distinctions by setting up multiple puddles, as explained in the previous topic.

When a stored process server retrieves RDBMS data for a requesting user, the database credentials that are available to a service account are used to fetch the data. If there is only one stored process server, all users have identical database layer access to the data. You can create some access distinctions among users by setting up additional stored process servers, as explained in the previous topic. You can also use alternate methods of library assignment to achieve different results. For example:

- To provide access using a shared database account, you can preassign the database library using the METAAUTOINIT or autoexec method and store the database credentials in a login that is available to the service account that fetches the data.
- To provide access using a shared database account, you can include a LIBNAME statement with credentials in the stored process. A stored process that includes credentials should be protected with physical and metadata layer access controls.
- To provide physical access based on the requesting user's identity, you can create a parameterized stored process that prompts for database credentials and uses the resulting macro variables for a LIBNAME assignment in the code.

Note: Another way to launch a stored process under the identity of the requesting user is to register the stored process with a standard workspace server rather than a stored process server. In the current release, stored processes that run on workspace servers cannot produce streaming output. △

PART 2

First Steps in Security Administration

CHAPTER 4

Securing a Deployment

Overview of Securing a Deployment

Immediately after installation, you should increase the level of security in your deployment by performing these tasks:

- □ On Windows systems, set operating system permissions to protect the configuration directories.
- □ Enable encryption beyond the default protection for transmission of user credentials.
- □ Set metadata permissions to protect the foundation metadata repository.
- □ For accountability, create an individual metadata identity for each administrator. For convenience, you can make these users members of an Administrators group.
- □ Set metadata permissions to protect security-related resources within the foundation repository.
- □ Minimize the availability of accounts.

Protecting the Configuration Directories (Windows)

Initial Operating System Settings

For a secure deployment, the configuration directory must be protected by operating system controls that prevent copying of repository data sets and potential capture of passwords. In the current release, the installation process does not establish these protections on Windows machines, so you must set them manually. Appropriate default protections are set for you on UNIX and z/OS (see "Default Directory Permissions" in the chapter "Understanding the State of Your System" in the *SAS Intelligence Platform: System Administration Guide*).

Recommended Settings (Windows)

The recommended settings are documented in the following table. These recommendations assume that your SAS servers and spawners run as services under the Local System account.

Table 4.1 Recommended Operating System Protections on Windows

Folders	Recommended Permissions
`MetadataServer`, **`OLAPServer`**, **`ObjectSpawner`**, **`ShareServer`**, **`ConnectServer`**	Grant Full Control to SYSTEM and Administrators. Remove all other users and groups.
`BatchServer`, **`SASEnvironment`**, **`Users`**, **`Utilities`**, **`WorkspaceServer`**	Grant Full Control to SYSTEM and Administrators. Grant Read to all SAS server users.
`WorkspaceServer\logs`	If you enable logging for the workspace server and use this default location for the logs, then all users of the workspace server should be granted Modify for this subdirectory.
`SASEnvironment\SASCode\Jobs`	Grant Modify to all SAS server users.
`StoredProcessServer`, **`StoredProcessServer\logs`**	Grant Full Control to SYSTEM, Administrators, and the SAS General Server User (sassrv). Remove all other users and groups.

Note: By default, these folders are located under **`Lev1\SASMain\`**. △

Enabling Encryption

Initial Encryption Settings

By default, only the user credentials in an initial credential exchange are protected during transmission. All other traffic, including credentials that are retrieved from the metadata repository for authentication to other systems, are transmitted without encryption. The user credentials in an initial exchange are encrypted using the SAS proprietary 32-bit algorithm that is provided with Base SAS software. This algorithm is appropriate for preventing accidental exposure of information.

Increase the Level of Protection

If network security is essential for your site, you can use an industry-standard algorithm to encrypt all traffic, thus making it extremely difficult for anyone to discover the content of messages that are sent over the network. Implementing this protection involves installing and configuring SAS/SECURE software on every host and desktop client machine. You must specify the same algorithm and level of encryption throughout the deployment. On server machines, SAS/SECURE requires a separate software license.

To enable encryption, complete these steps:

1. On each SAS server machine, install the server-side version of SAS/SECURE software, accepting the default installation location.
2. On the middle-tier machine, complete these steps:
   - a Install the client-side version of SAS/SECURE software.
   - b Rebuild the WAR files for the Web applications, including the **`sas.rutil.jar`** file, and then reconfigure and redeploy the Web applications.
3. On each machine that has a Windows desktop client, install the Windows version of SAS/SECURE.
4. On each machine that has a Java desktop client, install the Java version of SAS/SECURE. Then copy the **`sas.rutil.jar`** file to *all* of the following locations (as applicable for each machine):
   - □ For SAS Management Console, copy the file to *drive*`:\Program Files\SAS\SASManagementConsole\9.1` (Windows) or *install-location*`/SASManagementConsole/9.1` (UNIX).
   - □ For SAS Data Integration Studio, copy the file to *drive*`:\Program Files\SAS\SASETLStudio\9.1`.
   - □ For SAS OLAP Cube Studio, copy the file to *drive*`:\Program Files\SAS\SASOLAPCubeStudio\9.1`.
   - □ For SAS Information Map Studio, copy the file to *drive*`:\Program Files\SAS\SASInformationMapStudio\9.1`.
5. Edit the start-up commands for your metadata and OLAP servers as follows:
   - a In the -netencralg option, replace the **`sasproprietary`** algorithm with one of the industry standard algorithms that is offered by SAS/SECURE.
   - b For maximum protection, you can modify the **`cel=credentials`** argument to the **`objectserverparms`** string to specify an encryption level of **`everything`**.

     `CEL=CREDENTIALS` The only information that is encrypted is the transfer of credentials when a connection is made to a SAS Metadata Server, SAS Workspace Server, SAS Stored Process Server, or SAS OLAP Server.

     `CEL=EVERYTHING` All communications are encrypted.

   Note: For servers that run as services on a Windows system, make the preceding changes in the configuration file *path-to-config-dir*`\`*level*`\`*SAS-application-server*`\`*server-type*`\sasv9_`*server-type*`.cfg`. For example, to change the configuration file for a metadata server, you might edit the file `C:\SAS\ETLServerMin\Lev1\SASMain\MetadataServer\sasv9_MetadataServer.cfg`. △

Note: For servers that run on a UNIX system or are started via scripts on a Windows system, edit the start-up scripts. △

6 Use SAS Management Console to edit the server definitions by completing these steps for each physical server under Server Manager:

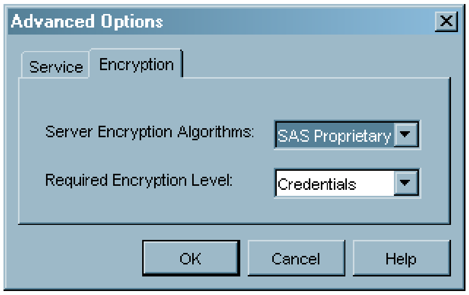
a Select the server. This causes the server's connection(s) to appear on the right side of the interface.

Note: If the server has more than one connection associated with it, you must perform the following steps for each connection. △

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

c In the Properties dialog box, select **Options ▶ Advanced Options ▶ Encryption**.

d On the **Encryption** tab, select the same encryption algorithm and encryption level that you specified in step 5.



Server Encryption Algorithms: SAS Proprietary

Required Encryption Level: Credentials

Note: For more information about encryption and SAS/SECURE, see *Data Security Technologies in SAS* in SAS OnlineDoc. △

Encrypting Passwords that are Included in Configuration Files

During installation, passwords for required accounts (such as sastrust, saswbadm, and sasguest) are written to configuration files. In the current release, the installation process encodes these passwords using base64 encoding. For greater protection, it is recommended that you encrypt these passwords using SAS proprietary 32-bit encryption that is provided with Base SAS software. For example, to encrypt a password of "SAStrust1" you would submit this code in the SAS Program Editor:

```
proc pwencode in='SAStrust1' method=sasenc;
run;
```

The encrypted password is written to your SAS log. When you use **method=sasenc**, the first part of the password is **{sasenc}**. When you copy the encrypted password into a configuration file, always overwrite the *entire* password, including the encoding-type indicator **{sas001}** and any trailing equal signs (**=**).

It is important to provide host protection for configuration files that include passwords. For a list of files that contain passwords, see “Configuration Files That Include Passwords” on page 100. Because Web application deployment areas often include passwords, these areas and any application WAR files should also be host protected against both read and write access.

Protecting the Foundation Repository

Initial Settings for the Foundation Repository

During installation, broad access to the metadata repository is granted to the PUBLIC group. While these initial settings are convenient for an empty repository (because they enable any user to add resources), they do not provide a secure environment for a populated deployment.

Temporarily Restrict Access to the Repository

This section describes how you can protect your deployment while you are adding users, resources, and specific access controls. The approach is to temporarily give a few administrators exclusive access to the repository.*

To deny access for all users other than the *unrestricted user*, complete these steps:

1. Log on to SAS Management Console by opening a metadata profile with the *unrestricted user* account (SAS Administrator).
2. In the navigation panel, select **Environment Management ▶ Authorization Manager ▶ Access Control Templates ▶ Default ACT**.

   Note: In SAS Management Console, the repository ACT is represented by a blue cylinder icon and is named "Default ACT" by default. △

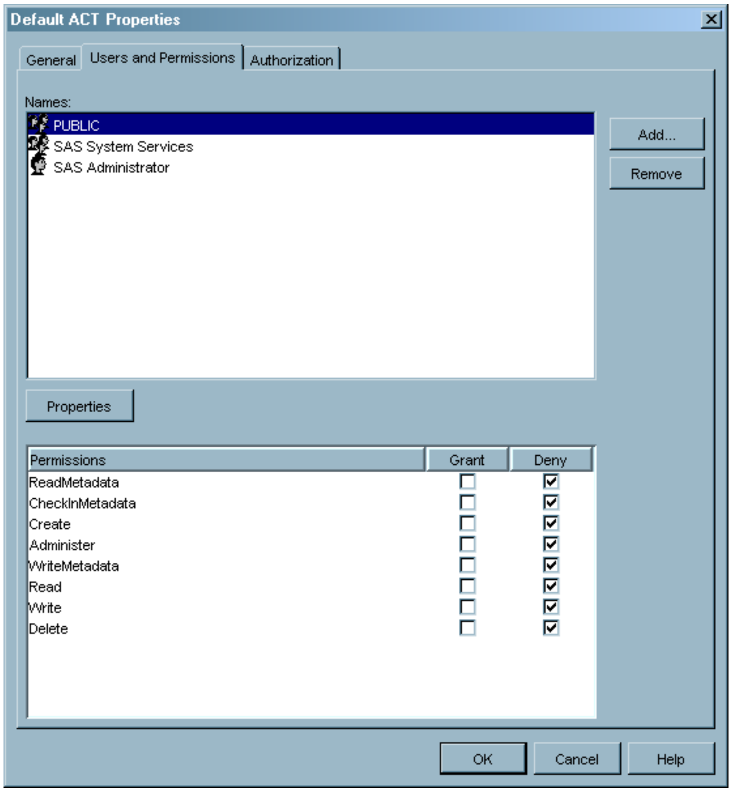
3. From the menu bar, select **File ▶ Properties** to open the Default ACT Properties dialog box. Then select the **`Users and Permissions`** tab.

   CAUTION:
   The Users and Permissions tab looks very similar to the Authorization tab. When you want to set default controls for a repository, be sure that you are on the **`Users and Permissions`** tab. △

4. On the **`Users and Permissions`** tab, select **`PUBLIC`** in the **`Names`** list. In the permissions list for the PUBLIC group, select the **`Deny`** check box for every permission.

* When you are ready to allow regular users to access the deployment, you will have to expand these repository ACT settings as documented in “Repository ACT Settings for ReadMetadata and WriteMetadata Permissions” on page 107.

Display 4.1 Repository ACT Settings for the PUBLIC Group



Names:

- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	□	☑
Write	□	☑
Delete	□	☑

Note: Other identities that are listed (such as the SAS Administrator and the SAS System Services group) were added during the installation and configuration process. Do not modify or remove the permission settings for any such identities. △

5 Click **OK** to save the settings and close the Default ACT.

CAUTION:

At this point, your deployment is not fully functional. You must restore access for the required users (such as sastrust, saswbadm, and sasguest) as explained in the following topic. △

Selectively Restore Access for the Required Users

Because they are members of the PUBLIC group, the SAS Guest User and the SAS Web Administrator are affected by the denials that you just gave to the PUBLIC group. The following table explains how to restore access for these users.

Table 4.2 Restoring Access for Required Users

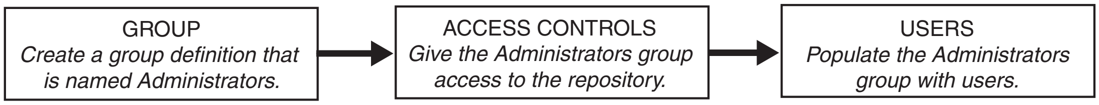
Task	Recommended Approach
For the SAS Web Administrator, restore WriteMetadata access to the repository.	On the **`Users and Permissions`** tab of the repository ACT, add the Portal Administrators group and grant WriteMetadata permission for this group.
For the SAS Guest User, restore ReadMetadata and WriteMetadata access to the repository. The SAS Guest User also needs Read permission for certain resources. For more information, see “How to Manage Read Access” on page 117.	On the **`Users and Permissions`** tab of the repository ACT, add the SAS Guest User and grant ReadMetadata and WriteMetadata permissions for this user. From the Public Kiosk of the SAS Information Delivery Portal, users can view all resources for which the SAS Guest User has ReadMetadata access. For this reason, you might choose to delay restoring this permission until you have set direct access controls to protect any sensitive content.

Note: If your deployment includes a SAS/SHARE server that specifies the sassrv account as metauser in its **`sasv9_ShareServer.cfg`** file, you must also restore ReadMetadata access for the SAS General Server User. On the **`Users and Permissions`** tab of the repository ACT, add the SAS General Servers group and grant ReadMetadata permission for this group. △

Creating a Group of Metadata Administrators (Optional)

The purpose of creating one or more *administrative users* is to enable each administrator to use his or her own account, rather than continuing to share the highly privileged *unrestricted user* account (sasadm). Administrative users can perform almost all metadata administrative tasks. Assigning the administrators to a user group simplifies the process of managing access controls for these users. The following instructions describe how to create a user group for administrators, give the group broad access to the repository, and populate the group by adding *administrative users* to the deployment. The sequence is depicted in the following figure.

Figure 4.1 Sequence for Setting Up Security for Administrators


GROUP
Create a group definition that is named Administrators.
ACCESS CONTROLS
Give the Administrators group access to the repository.
USERS
Populate the Administrators group with users.

To create the administrators group and define their default access, complete these steps:

1 Log on to SAS Management Console by opening a metadata profile with the *unrestricted user* account (SAS Administrator).

2 In the navigation panel, select **`User Manager`**.

3 Open the New Group Properties dialog box by selecting this path from the menu bar: **Actions ▶ New ▶ Group**.

4 Create a group definition for your administrators.

- **a** On the **`General`** tab of the New Group Properties dialog box, enter **`Administrators`** as the group name.
- **b** On the **`Logins`** tab, do not add any account information (unless the group members share an account on another system).
- **c** Click **`OK`** to save and close the group definition.

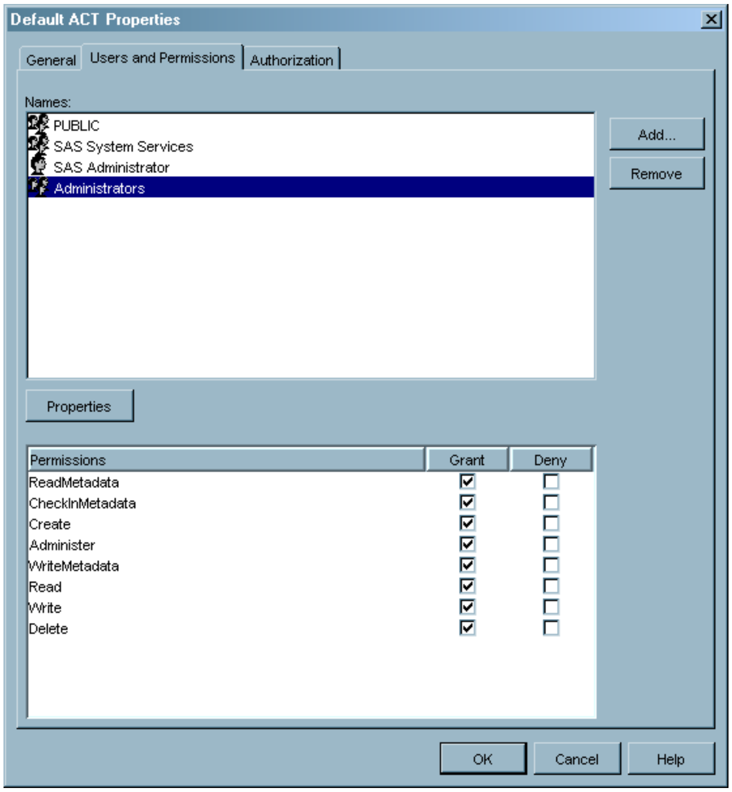
5 Define the group's default access to the repository.

- **a** In the navigation panel, select **Environment Management ▶ Authorization Manager ▶ Access Control Templates ▶ Default ACT**.

  Note: In SAS Management Console, the repository ACT is represented by a blue cylinder icon and is named "Default ACT" by default. △

- **b** From the menu bar, select **File ▶ Properties** to open the Default ACT Properties dialog box.
- **c** On the **`Users and Permissions`** tab, click **`Add`**.
- **d** In the Add Users and/or Groups dialog box, move the Administrators group to the **`Selected Identities`** list and then click **`OK`**.
- **e** On the **`Users and Permissions`** tab, select **`Administrators`** in the **`Names`** list and grant all permissions.*

Display 4.2 Repository ACT Settings for the Administrators Group



Names:
- PUBLIC
- SAS System Services
- SAS Administrator
- Administrators

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

To populate the new group, complete these steps for each administrator:

* You can choose to set the Read, Write, Create, Delete, CheckInMetadata, and Administer permissions on specific resources rather than as default permissions for the entire repository.

1 If the user does not already have an account on the metadata server machine, create an operating system account (or an LDAP or Active Directory account if you are using one of these providers). On Windows platforms, give the user the **`Log on as Batch`** right.

Note: If you have a mixed authentication environment, create any additional accounts that the administrator needs in order to access other servers (such as workspace servers, stored process servers, or database servers). △

2 Give the user status as an *administrative user* of the metadata server by adding the user ID of the primary account that you created in step 1 to the **`adminUsers.txt`** file. For details, see "How to Designate an Unrestricted, Administrative, or Trusted User" on page 93.

3 Create a metadata definition for the user and assign the user to the Administrators group.

a Log on to SAS Management Console by opening a metadata profile with an *unrestricted user* account (sasadm), and access the foundation repository.
b In the navigation panel, select **`User Manager`**.
c Open the New User Properties dialog box by selecting this path from the menu bar: **Actions ▶ New ▶ User**.
d On the **`General`** tab, enter the user's name in the **`Name`** field. The other fields on the **`General`** tab are optional.
e On the **`Logins`** tab, add a login that contains the fully qualified form of the user ID for the primary account that you created in step 1. If this login will be used to provide access to other servers from applications that do not cache credentials, include the password and specify the DefaultAuth authentication domain.

Note: If you have a mixed authentication environment, add other logins as needed to support additional authentication. These logins would contain the credentials for any additional individual user accounts that you created in step 1. △

f On the **`Groups`** tab, make the user a member of the Administrators group.
g Click **`OK`** to save and close the user definition.

Note: It is important to restrict access to this group definition as explained in "Protect Group Definitions" on page 76. △

Setting Explicit Protections for Security-Related Resources

Initial Settings for Security-Related Resources

By default, ACTs and user-defined groups are protected only by the access controls that these resources inherit from the repository ACT. When you examine the **`Authorization`** tab for one of these resources, you will see inherited (grey background color) grants and denials that come from the settings on the repository ACT. Because you will eventually grant wider access on the repository ACT, you should protect specific resources with additional, *direct* controls. You set direct access controls on the **`Authorization`** tab of each resource that you want to protect. This ensures that the resources will remain protected as you expand access to the repository.

Protect Access Control Templates

To enable only members of the Administrators group to modify or delete an ACT, take WriteMetadata permission away from PUBLIC and then give WriteMetadata permission back to the Administrators group. For example, to protect the repository ACT, complete these steps:

1 In SAS Management Console, select the repository ACT under **Environment Management ▶ Authorization Manager ▶ Access Control Templates**.

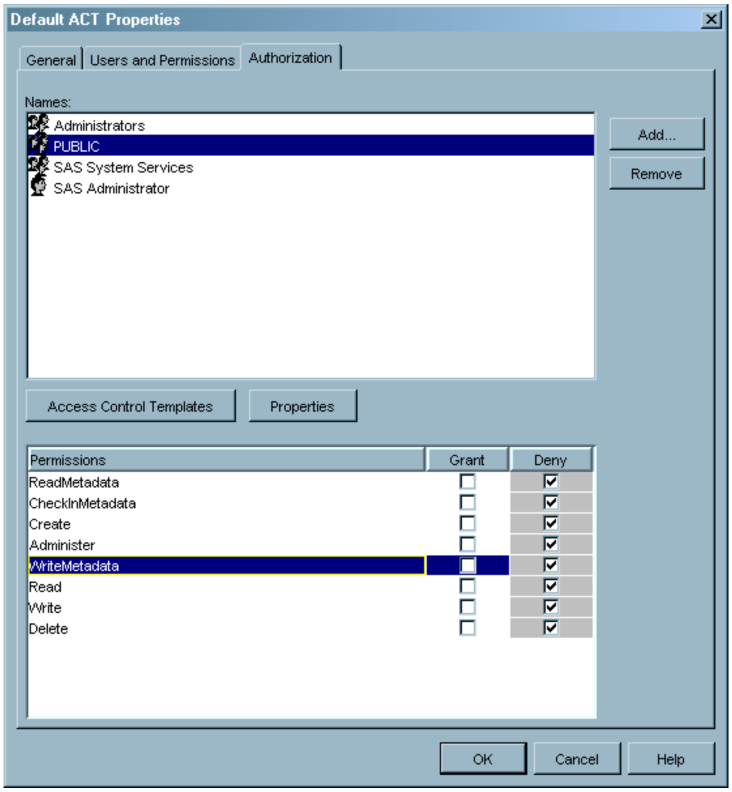
2 Right-click and select **`Properties`** from the pop-up menu. Then select the **`Authorization`** tab.

   Note: You use the **`Authorization`** tab (rather than the **`Users and Permissions`** tab) because you want to control who can make changes to this ACT. △

3 On the **`Authorization`** tab:

   a In the **`Names`** list, select **`PUBLIC`**. In the permissions list for the PUBLIC group, the deny WriteMetadata check box should already be selected and have a gray background color. This denial comes from the pattern that you defined on the **`Users and Permissions`** tab of the repository ACT.

Display 4.3 Initial Setting: PUBLIC Has Inherited Denials



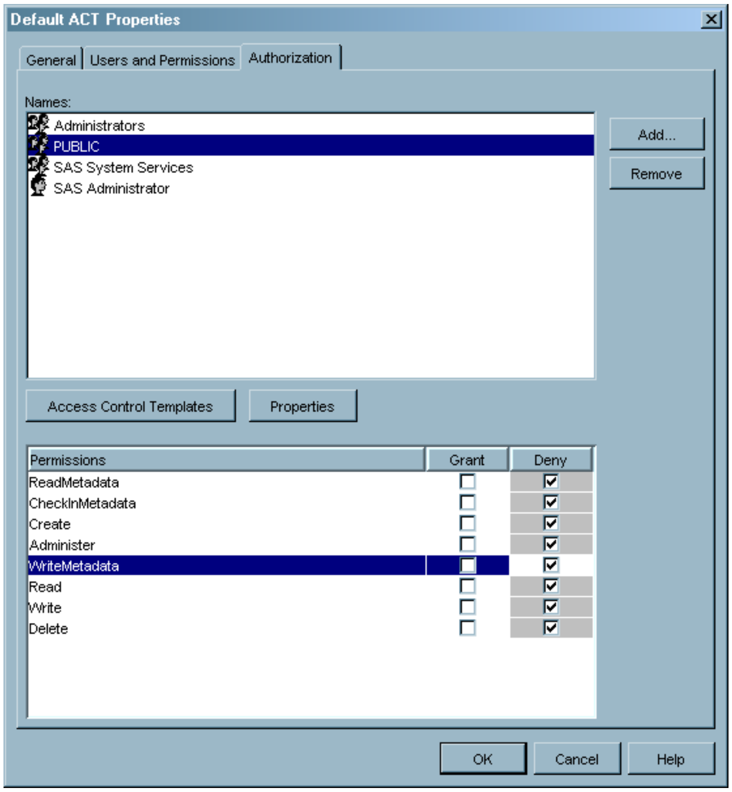
Names:

Administrators
PUBLIC
SAS System Services
SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	□	☑
Write	□	☑
Delete	□	☑

   b Select the (already selected) **`Deny WriteMetadata`** check box to add a directly assigned denial of WriteMetadata permission for the PUBLIC group. The directly assigned denial is indicated by the absence of a background color.

Display 4.4 Revised Setting: PUBLIC Has Explicit Denial of WriteMetadata



Names:

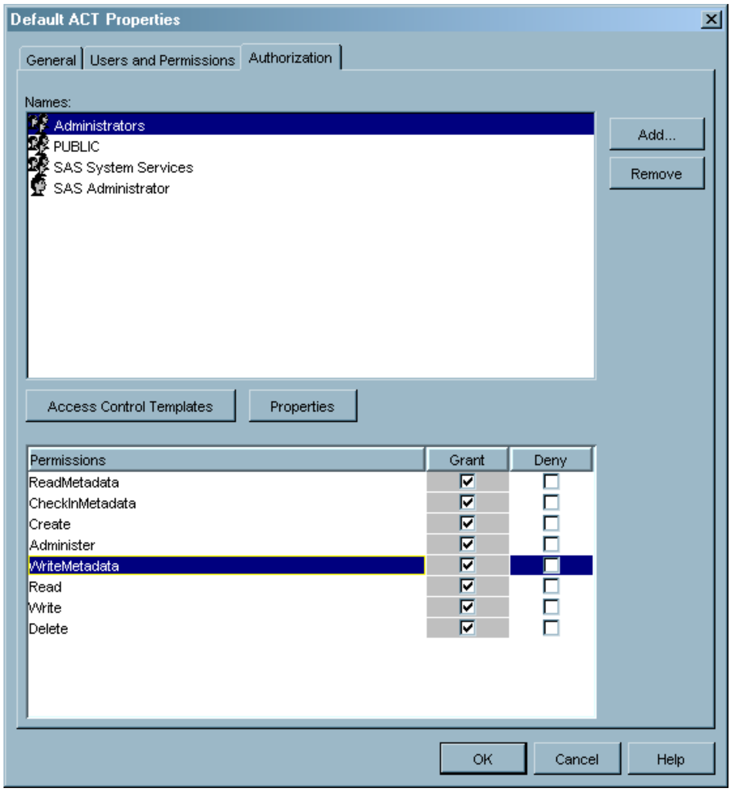
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	□	☑
Write	□	☑
Delete	□	☑

The directly assigned denial ensures that the current resource will remain protected as you expand default WriteMetadata access to the repository. Explicit controls have precedence over inherited controls, so the PUBLIC group's explicit denial of WriteMetadata will override any grants of WriteMetadata that come from the repository ACT.

c In the **`Names`** list, select **`Administrators`**. In the permissions list for the Administrators group, the **`Grant WriteMetadata`** check box should already be selected and have a gray background color. This grant comes from the pattern that you defined on the **`Users and Permissions`** tab of the repository ACT.

Display 4.5 Initial Setting: Administrators Has Inherited Grants



Names:

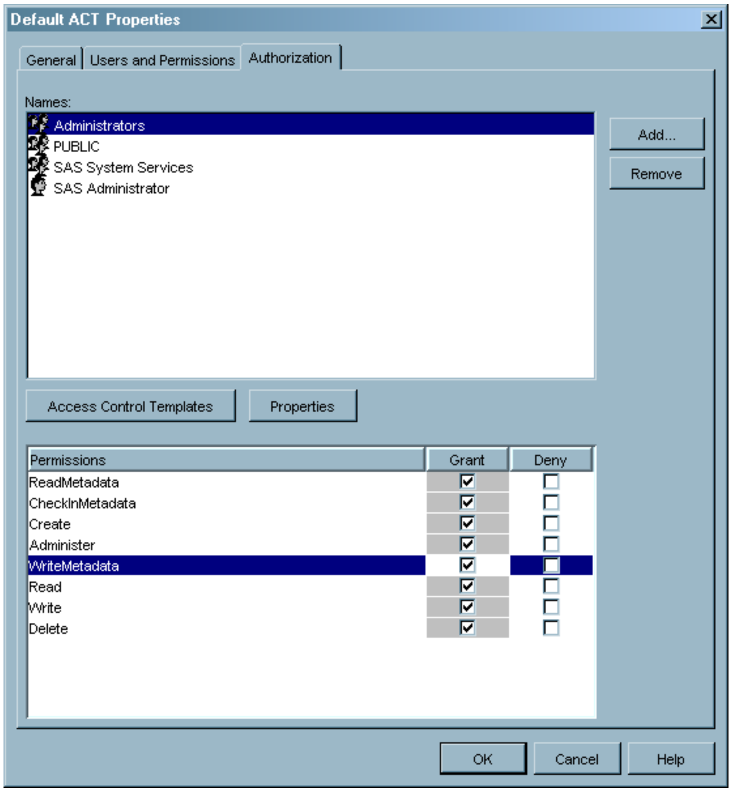
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

Because members of the Administrators group are also members of the PUBLIC group, they are affected by the direct denial of WriteMetadata permission that you set in step 3b. To enable members of the Administrators group to modify or delete this resource, you must give the Administrators group a direct grant of WriteMetadata permission.

Select the (already selected) **`Grant WriteMetadata`** check box to add a directly assigned grant of WriteMetadata permission for the Administrators group. The directly assigned grant is indicated by the absence of a background color.

Display 4.6 Revised Setting: Administrators Has Explicit Grant of WriteMetadata



Names:

- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

4 In the properties dialog box, click **`OK`** to save the settings and close the ACT.

Now the repository ACT is protected. You will have to repeat these steps for any other ACTs that you create.

Example: Use a Custom ACT

As an alternative to setting individual access control entries (ACEs) on the **`Authorization`** tab of every ACT, you can use this approach:

1 Create a custom ACT that has a reusable identity/permission pattern on its **`Users and Permissions`** tab.

- **a** In SAS Management Console, navigate to **Environment Management ▶ Authorization Manager ▶ Access Control Templates**. Right-click and select New Access Control Template from the pop-up menu.
- **b** On the **`General`** tab in the New Access Control Template dialog box, enter a name such as **`ACT Securing ACTs`**. Or, if you plan to also use this ACT to protect a variety of resource types, you might want to enter a more general name such as **`Exclusive WriteMetadata for Administrators`**.
- **c** On the **`Users and Permissions`** tab, click **`Add`** and add the PUBLIC and Administrators groups to the **`Names`** list.
- **d** On the **`Users and Permissions`** tab, select **`PUBLIC`** and deny WriteMetadata permission. Then select **`Administrators`** and grant WriteMetadata permission.
- **e** Click **`OK`** to save the new ACT.

2 Apply the new ACT on the **`Authorization`** tab of each resource that you want to protect with this particular identity/permission pattern. For example, you can apply the custom ACT to protect itself by completing these steps:

a Open the properties dialog box for the custom ACT.

b On the **Authorization** tab, examine the settings for the Administrators and PUBLIC groups. You will see inherited grants and denials (with grey background colors). To apply the new custom ACT on top of these default settings, click **Access Control Templates**.

c In the Add/Remove Access Control Templates dialog box, move the custom template to the **Currently Using** box and click **OK**.

On the **Authorization** tab, examine the settings for the Administrators and PUBLIC groups again. For the Administrators group, the grant of WriteMetadata now has a green background color. This indicates that the grant comes from the directly applied ACT. Similarly, the WriteMetadata denial for the PUBLIC group now has a green color. The green color assures you that this is an explicit, directly assigned control that will be preserved when default WriteMetadata access to the repository is expanded.

This approach gives you a more centralized way to manage access to your ACTs. If you change your mind about which groups or users should be able to modify or delete your ACTs, you can make the change in one place (on the custom ACT) rather than revisiting every ACT to change the individual ACEs that you set on each resource.

Note: If you want different groups to be able to make changes to particular ACTs, then you cannot use a single ACT to manage access to all of these objects. You use an ACT to manage access when you want to apply the *same* identity/permission pattern to multiple resources. △

Note: You can apply this custom ACT to any resource for which these permission settings are appropriate. For example, you might also use this custom ACT to protect group definitions. △

Protect Group Definitions

It is essential that you protect all of your group definitions. The manual approach is similar to the approach that was used to protect ACTs:

1 Locate each group definition (under the **User Manager** node in SAS Management Console).

2 Access the **Authorization** tab for each group definition.

3 Set direct controls that deny WriteMetadata permission to PUBLIC and grant WriteMetadata permission to the Administrators group.

As an alternative to manually setting these permissions on each group definition, you can use an ACT to manage access. SAS provides a macro to simplify this process. See “Macro for Protecting Group Definitions” on page 95.

Minimizing the Availability of Accounts

The following list provides suggestions for reducing exposure of privileged accounts, service accounts, and regular user accounts:

- □ Make *trusted user* capabilities available only where these special privileges are needed. *Trusted user* capabilities are needed in these circumstances:
  - □ The deployment includes a SAS OLAP Server.
  - □ The deployment uses Web authentication.
  - □ The deployment uses batch reporting processes.

If your deployment does not make use of *trusted user* functionality, you can remove the user ID for the sastrust account from the **`trustedUsers.txt`** file. This makes the sastrust account a normal account that has no special privileges.

If your deployment requires *trusted user* functionality, you can reduce exposure of the privileged sastrust account by eliminating its use in situations that do not require *trusted user* privileges. For example, an account other than sastrust can be used as the pool administrator for pooled workspace servers.

□ Look for operating system-specific ways to minimize availability of accounts. For example, on Windows systems the *unrestricted user* and *trusted user* accounts can be local accounts on the metadata server machine rather than network accounts (these accounts are not authenticated on any other machines).

□ On Windows systems, reduce availability by granting the **`Log on as Batch`** right on only those accounts and machines where it is needed. Any user who is authenticated by a SAS server that is running on Windows and using host authentication must have this right on that machine.

□ If you are using workspace server pooling with SAS Web Report Studio, then protect the **`Foundation Services Manager User Service`** definition by completing these steps:

1. In SAS Management Console, navigate to **Foundation Services Manager ▸ Query and Reporting ▸ BIP Core Services ▸ Platform User Service ▸ Properties ▸ Authorization tab**.
2. Set direct controls that deny ReadMetadata and WriteMetadata permissions to the PUBLIC group and grant these permissions to the SAS System Services group and the Administrators group.

   Note: To set these permissions, follow a process similar to step 3 in "Protect Access Control Templates" on page 72. △

CHAPTER

5

Customizing the Authentication Configuration

Overview of Customizing the Authentication Configuration

This chapter contains instructions for setting up specific configurations. You should perform only those tasks that are applicable for your deployment. See “Making Preliminary Decisions about Authentication” on page 52 for help determining which configurations are appropriate for your environment.

Modifications to Support a Mixed Authentication Environment

Initial Authentication Configuration

With the assumption that all of your servers share a single authentication provider, the default configuration creates a single authentication domain named DefaultAuth. If your deployment uses more than one authentication provider, it will not be fully functional until you set up more authentication domains.

How to Create an Authentication Domain

You can create a new authentication domain while you are defining a server or a login in SAS Management Console. Instead of selecting an existing authentication domain, click **New** to access the New Authentication Domain dialog box. The name of an authentication domain should be meaningful to the people who create the logins and server definitions that will be associated with that authentication domain. For each authentication domain, you must define associations to the appropriate servers and logins. "How Authentication Domains Are Used" on page 9 explains the relationships between authentication domains, servers, and logins.

Note: There is no direct method for deleting an authentication domain from SAS Management Console. △

How to Change an Authentication Domain Assignment

In order to modify existing authentication domain assignments, you need to know how to locate those assignments in SAS Management Console.

□ The authentication domain for a server is specified on the `Options` tab of each of the server's connection definitions. For example, to access the authentication domain assignment for a SAS OLAP Server, you will select a path such as: **Server Manager ▶ SASMain ▶ SASMain Logical OLAP Server ▶ SASMain OLAP Server ▶ Connection: SASMain OLAP Server ▶ Properties ▶ Options**.

□ The authentication domain for a login is specified on the `Logins` tab of the user or group definition to which the login is assigned. You can see a login that is assigned to another user only when you log on to SAS Management Console as an *unrestricted user*.

Modifications to Support Additional Servers

How to Manage Authentication for an Additional Server

For each additional server that you register in the repository, complete these steps to support authentication:

1 In the server definition, associate the new server with an appropriate authentication domain.

   □ If the server uses the same authentication provider as an existing server, associate the new server with the authentication domain for that provider.

   □ If the server does not use the same authentication provider as an existing server, create a new authentication domain when you register the new server in the metadata.

2 In the new server's authentication provider, establish user accounts. These can be either individual accounts or shared accounts.

3 In the metadata repository, provide exactly one login that contains credentials for accessing the new server for each user who will access that server.

How to Use Shared Accounts

You can enable multiple users to access a server with the same account. Using shared accounts provides these advantages:

□ minimizes the number of user accounts you have to create in the operating system (or other authentication provider)

□ minimizes the number of user credentials you must store in the metadata to support single sign-on

- □ improves performance by facilitating the pooling of workspace servers for SAS Web applications

Using shared accounts has these disadvantages:

- □ reduces individual accountability
- □ reduces your ability to make access distinctions between users
- □ requires you to carefully coordinate credentials in the metadata so that no user has more than one login for any authentication domain

To store the user ID and password for an account that several users share, you add that user ID and password to a *group* definition. This enables all the members of that group to use those credentials.

Example: Managing Authentication to a Database Server

For example, to manage authentication to an Oracle server that is using database authentication, you would perform these tasks:

1. Create a new authentication domain for the Oracle server when you register that server in the metadata. The new authentication domain is necessary because users do not access the Oracle server with the same credentials that they use for any other server in the deployment.
2. In the Oracle authentication database, set up user accounts. You can use any of these approaches:
    - □ Give each user an individual account for Oracle. This provides the greatest accountability, but also necessitates 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 in the metadata, but also results in a loss of individual accountability and control.
    - □ Create a few Oracle accounts, each of which will be used by several users. This middle-of-the-road approach enables you to make some access distinctions in the Oracle authorization layer and still store only a few Oracle user IDs and passwords in the metadata.
3. In the metadata, store the credentials that users need in order to authenticate to the Oracle server. Depending on the approach you selected in step 2, this will involve one of these tasks:
    - □ If you created individual accounts on the Oracle server, then you must add an Oracle login to each user definition. The login must include the Oracle user ID and password. The login must be associated with the authentication domain that you created for the Oracle server.
    - □ If you created one shared account on the Oracle server, then you must identify (or create) a group that contains the users who will access the Oracle server. For that group, you add a login that includes the user ID and password for the Oracle shared account. You associate that login with the authentication domain that you created for the Oracle server.
    - □ If you created several shared accounts on the Oracle server, then you must identify (or create) a user group in the metadata for each shared account. You give each group one login for the Oracle server, and you assign users who will share an account to the user group that owns the login for that account.

Modifications to Support Alternative Authentication Mechanisms

Using Web Authentication

Configuring your SAS Web applications to use the Web server's authentication provider (rather than the metadata server's authentication provider) enables you to avoid creating accounts for your Web application users on the machine on which your metadata server is running. However, this configuration has these potential disadvantages:

- □ Switching to Web authentication can result in increased credential management work. To determine whether this is the case in your environment, you need to understand how Web authentication works (see "Initial Authentication on a Web Application Server" on page 13) and understand the relationships between authentication providers, servers, and credential management requirements (see "Authentication Scenarios" on page 21).
- □ If you use Web authentication for the SAS Information Delivery Portal, you will not have a public kiosk.

The configuration changes involve setting certain properties for each Web application and redeploying those applications. For instructions, see "Changing to Trusted Web Authentication" in the chapter "Setting Up and Managing Middle-Tier Security" in the *SAS Intelligence Platform: Web Application Administration Guide*.

Using LDAP or Active Directory

Best Practice for Using LDAP or Active Directory

At many sites, the host authentication process makes use of LDAP or Active Directory as a back-end authentication mechanism. This is the preferred way to use LDAP or Active Directory in the SAS Intelligence Platform, and no additional configuration is required to support this scenario. From the perspective of the SAS servers, this is host authentication, because the SAS server relies on the host (which relies in turn on LDAP or Active Directory).

If you have LDAP or Active Directory accounts, we recommend that you determine whether your host authentication process already interacts with these providers or can be modified to interact with these providers. For example:

- □ On Windows, Active Directory is almost always the back-end authentication mechanism behind the host.
- □ On UNIX, Pluggable Authentication Modules (PAM) can enable a host to use LDAP or Active Directory as a back-end authentication mechanism.

Direct Use of LDAP or Active Directory

If your host authentication process cannot interact with your LDAP or Active Directory provider, you can choose to enable the metadata server and the OLAP server to use these alternate providers *directly*. Before you perform the additional configuration that this scenario requires, review the preceding topic and consider these points:

- □ Direct use of LDAP or Active Directory is not an alternative to storing user information in a SAS Metadata Repository. The security model requires you to maintain user definitions in the metadata, regardless of your choice of authentication provider.

□ Only the metadata server and the OLAP server can directly use alternate providers. Stored process servers and workspace servers always rely on the host operating system for authentication, so cached LDAP and Active Directory credentials cannot provide access to these servers. Direct use of an alternate provider can increase the need to store credentials in the metadata.

If you want to use an alternate provider *directly*, complete these steps:

1 Verify that user accounts are in place with the alternate provider. It is not necessary to create LDAP or Active Directory entries for the required accounts. These instructions anticipate that all required accounts will continue to exist with the host authentication provider.

2 Add system environment variables for the metadata server and OLAP server.

Table 5.1 System Environment Variables for Direct Use of LDAP

Variable	**Value**
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_HOST	The host name of the machine where LDAP is running.
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 5.2 System Environment Variables for Direct Use of Active Directory

Variable	**Value**
AD_PORT	The port number for Active Directory. The default is 389.
AD_HOST	The host name of the machine where Active Directory is running.

For example, on Windows you can add these settings to **`sasv9_MetadataServer.cfg`** and **`sasv9_OLAPServer.cfg`**:

```
-set LDAP_HOST myhost
-set LDAP_BASE "ou=emp,o=us"
-set LDAP_PORT 389
```

For example, on UNIX you can add these settings to **`MetadataServer.sh`** and **`OLAPServer.sh`**:

```
LDAP_HOST=myhost
export LDAP_HOST
LDAP_BASE="ou=emp,o=us"
export LDAP_BASE
LDAP_PORT=389
export LDAP_PORT
```

Note: 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 for the metadata server or OLAP server, 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. △

3 In the invocation command for the metadata server and the OLAP server, use the AUTHPROVIDERDOMAIN option (authpd) to indicate which credentials should be sent to your alternate provider. For example:

-authpd LDAP:sas
causes the server to send credentials for users who log on as *anything*`@sas` to LDAP for authentication.

-authpd ADIR:sas
causes the server to send credentials for users who log on as *anything*`@sas` to Active Directory for authentication.

Note: You can use another value instead of `sas`. The examples in these instructions use `@sas` to make it easy for users to remember what to append to their LDAP or Active Directory user IDs when they log on to SAS applications. △

4 Tell users that when they log on to a SAS application, they must specify their user ID as follows:

□ For LDAP, use *myLDAPidentifier@authpdLDAPvalue* (for example, `tom@sas`).

Note: The SAS server uses the user's *myLDAPidentifier* value to find the user's DN. By default, the SAS server expects this value to be the user's LDAP uid. To enable users to use a different LDAP attribute, set the LDAP_IDATTR system environment variable. △

□ For Active Directory, use *myQualifiedWindowsID@authpdADIRvalue* (for example, `OrionNT\Tom@sas` or `Tom@OrionNT@sas`).

Note: The SAS server sends the user's *myQualifiedWindowsID* value to Active Directory for authentication. △

5 In the metadata, ensure that each user definition includes a matching inbound login in which the user ID is specified as follows:

□ For LDAP, the format in the login is the same as the format that the user submits: *myLDAPidentifier@authpdLDAPvalue* (for example, `tom@sas`).

□ For Active Directory, the format in the login is an abbreviated version of the format that the user submits: *myQualifiedWindowsID* (for example, `OrionNT\Tom`).

Note: The password is not specified in this login unless the login is also used for outbound purposes. △

Note: The authentication domain for this login is usually the authentication domain that you are using for the OLAP server and the metadata server. Typically, you will associate this login with a new authentication domain with a name such as `LDAPAuth` or `ADIRAuth`. △

6 In the metadata, ensure that host credentials are stored to enable users to access stored process servers and workspace servers. Typically, this is achieved in one of these ways:

□ adding a password to each user's existing host login

□ creating a new login for each user to store the user ID and password for the user's host account

□ adding the user ID and password for a shared host account to a group definition

This table illustrates credential information in a deployment where LDAP is the alternate provider for a metadata server and an OLAP server running on Windows. In this example, the workspace server and stored process server are running on UNIX.

Table 5.3 Example: Credentials If Using LDAP: sas

Source of Credentials	User ID	Password	Authentication Domain
interactive log on by Tom	tom@sas	provided by Tom	LDAPAuth
login (inbound)	tom@sas	not stored	LDAPAuth
login (outbound)	unxtom	stored	UNIXAuth

This table illustrates credential information in a deployment where Active Directory is the alternate provider for a metadata server and an OLAP server running on UNIX. In this example, the workspace server and stored process server are running on Windows.

Table 5.4 Example: Credentials If Using ADIR: sas

Source of Credentials	User ID	Password	Authentication Domain
interactive log on by Tom	OrionNT\Tom@sas	provided by Tom	ADIRAuth
login (inbound and outbound)	OrionNT\Tom	stored	WINAuth

In this example, using Active Directory as an alternate provider enables you to avoid creating UNIX accounts for your Active Directory users. Note that in this configuration, a user ID that is cached when the user logs on cannot be reused to provide access to workspace servers or stored process servers, because that ID is not in a valid format for host authentication on a Windows machine. For this reason, the login in this example has these attributes:

□ To support inbound use, the user ID in the login is in the abbreviated format that matches the authenticated user ID as returned by Active Directory.

□ To support outbound use, the login includes a password and is associated with the authentication domain of the workspace server and stored process server.

Related Topics:

"How to Use Shared Accounts" on page 80

"How to Create an Authentication Domain" on page 79

Chapter 6, "User and Group Management," on page 89

"AUTHPROVIDERDOMAIN" in the chapter "SAS System Options" in the *SAS Language Reference: Dictionary*

"The PWENCODE Procedure" in the *Base SAS Procedures Guide*

PART 3

Identity Management

CHAPTER

6

User and Group Management

About the User and Group Management Tasks

This chapter explains how to use SAS Management Console to create and manage user and group definitions in the metadata. For an alternate method for creating metadata identity information, see Appendix 2, “Bulk-Load Processes for Identity Management,” on page 185.

Organizing Users Into Groups

Identifying Related Tasks

Make a list of the business tasks that each user performs and the content domain (such as the business unit, job title, or geographic region) in which each user operates. For example, your list might include these activities:

- □ viewing reports in a particular content domain (such as human resources)
- □ creating reports in a particular content domain
- □ scheduling jobs
- □ defining objects that represent computing resources in a metadata repository
- □ creating or maintaining user and group definitions

Organize your list of user tasks into logical groups. In this process, look for variations in

- □ the content domain
- □ the computing resources that are involved
- □ the type of access that is required
- □ the level of sensitivity of the underlying data

Analyze "Standard Group Metadata Identities" on page 182 and your task lists to identify which additional user groups you need. In this process, keep in mind your security goals. If you do not intend to define different levels of access for two sets of users, then you usually do not need to create separate user groups to represent each of those sets of users.

For example, if you want everyone to be able to read all of your data and you want to limit write access by job function, your list of tasks and groups might look like the tasks in the following table.

Table 6.1 A Simple Tasks-to-Groups Mapping

Tasks	Group
Create users and groups Administer servers and repositories Set repository level security	Administrators
Define metadata for data resources Define ETL processes Schedule jobs	ETL Developers
Create and maintain information maps	Information Architects
Create and publish standard reports	Report Creators
View all reports Make ad hoc changes to reports Save modifications to reports	Power Report Consumers
View all reports	Report Consumers

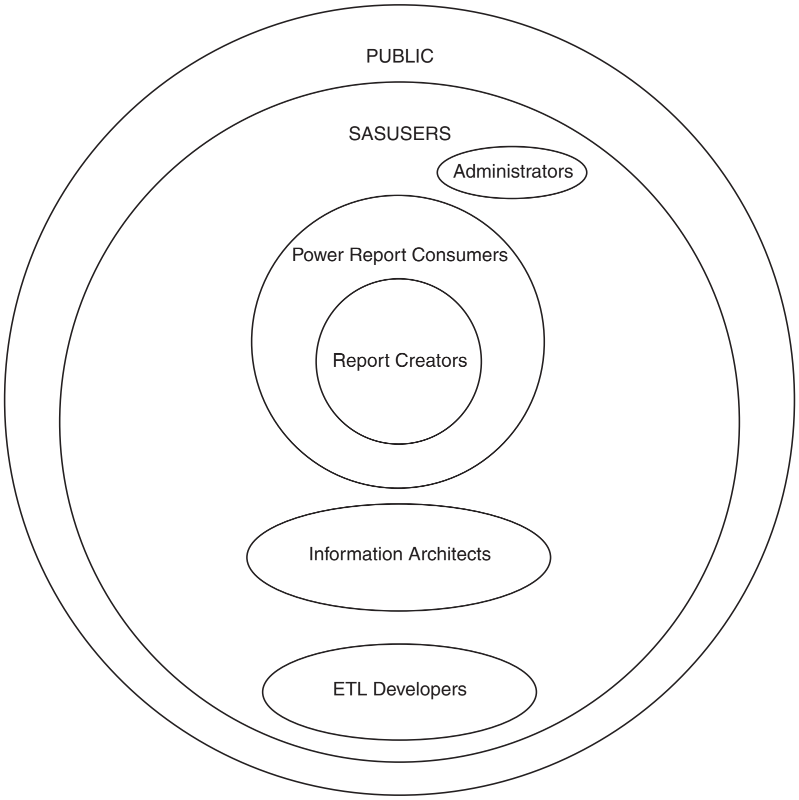
Note: Membership in a user group that is named Administrators does not enable a user to perform tasks that require status as an *administrative user* of the metadata server. For details, see "How to Designate an Unrestricted, Administrative, or Trusted User" on page 93. △

Defining the Group Structure and Membership

Decide how the groups that you create should relate to each other. In the metadata layer, one group can be a member of other groups. For example, a regional sales group can be a member of a worldwide sales group.

The following figure depicts one way you could structure the user groups that were identified in the simple tasks-to-groups mapping in the previous section.

Figure 6.1 Example of a Group Structure



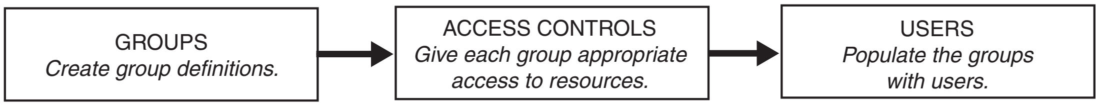
This group structure simplifies the process of defining and maintaining access controls. For example, if you make the Report Creators group a member of the Power Report Consumers group, you simplify the process of giving the report creators a superset of the access that you will give to the report consumers. You can assign permissions that you want both groups to have to the Power Report Consumers group, and the permissions that you want only the creators to have to the Report Creators group.

Determine which individuals in your organization should be assigned to each of the user-defined groups that you identified. You will usually assign each user definition to one or more group definitions in the metadata. Remember that you do not always have to directly add individual users to every user group; groups can also be members of other groups.

Sequence for Populating a Deployment

You can use the following sequence to populate your deployment.

Figure 6.2 Sequence for Expanding Access to Your Deployment


GROUPS
Create group definitions.

ACCESS CONTROLS
Give each group appropriate access to resources.

USERS
Populate the groups with users.

In this sequence, you create user groups and define access controls before you add individual users to the deployment. This centralizes management of access controls by assigning permissions to user groups rather than to individual users. This also enables you to separate tasks that are typically performed by a security architect (such as designing the user group structure and the access control strategy) from the more administrative tasks (such as adding individual users to a deployment).

You can perform the entire sequence separately for each group (or set of groups) in the deployment. This phased approach is appropriate for a gradual rollout by job function. For example, a typical sequence is ETL developers, then information map creators, then report creators, and then report consumers. Or, you can perform the entire sequence one time for the entire deployment. In this approach, you set up all of the groups, then create all of the access controls, and then add users to the deployment. This enables you to do a small (but comprehensive) pilot by adding a few users to each group and then verifying that you get the security behaviors that you expect.

How to Create a User Group

Most environments will have several user-defined groups in addition to the two standard user groups (PUBLIC and SASUSERS) that have implicit membership.

To create a user group in the metadata, complete these steps:

1 Log on to SAS Management Console and access the foundation repository. You should create all of your user and group definitions in a single foundation metadata repository.

2 In the navigation panel, select `User Manager`.

3 Open the New Group Properties dialog box by selecting this path from the menu bar: **Actions ▶ New ▶ Group**.

4 Create the group definitions. For each group definition, complete these steps:

   a On the `General` tab, enter a name for the group.

   b On the `Logins` tab, add a login to the group definition only if you are using the group to provide access to servers using a shared account. A login on a group definition should contain the user ID and password for a shared account that has been established with an authentication provider. A login on a group definition should be assigned to an appropriate authentication domain.

   For example, you can enable multiple users to share a database account by storing the credentials for that account in a login on the `Logins` tab for a group definition. All users who are members of that group definition can use that login to access the database server.

   c On the `Members` tab, add any other groups that are members of the current group, in accordance with your planned group structure.

   Unless you define groups in a particular order, you might have to return to a group's `Members` tab to add other groups as members. For example, if you want GroupA to be a member of GroupB, you must do either of these things:

   - Create GroupA before you create GroupB. This enables you to add GroupA as a member when you define GroupB.
   - Create GroupA after you create GroupB. This requires you to return to the `Members` tab of GroupB after you create GroupA to add GroupA as a member.

   d Click `OK` to save and close the group definition.

5 Secure each group definition with directly assigned permissions. If your goal is to enable only members of the Administrators group to make changes to your group

definition, you will take WriteMetadata permission away from PUBLIC and then give WriteMetadata permission back to the Administrators group. For instructions, see "Protect Group Definitions" on page 76.

How to Add a User

To create the necessary accounts, metadata identities, and group membership assignments for a regular user, complete these steps for each user:

1 If the user does not already have an account that enables the user to access the metadata server machine, create such an account. This can be an operating system account on the metadata server host, an LDAP or Active Directory account (if you are using an alternate authentication provider), or an account with a Web authentication provider (if you are using Web authentication). On Windows platforms, give the user the **`Log on as Batch`** right. You can do this by assigning the user to the SAS Server Users group in the operating system.

   Also create any other individual accounts that the user needs to access servers such as unpooled workspace servers, stored process servers, or database servers.

2 Create a metadata identity for the user (unless the user's access needs can be met by membership in the PUBLIC group).

   a Log on to SAS Management Console by opening a metadata profile with your *administrative user* account (or with the *unrestricted user* account). Access the foundation repository. You should create all of your user and group definitions in a single foundation metadata repository.

   b In the navigation panel of SAS Management Console, select **`User Manager`**.

   c Open the New User properties dialog box by selecting this path from the menu bar: **Actions ▶ New ▶ User**.

   d On the **`General`** tab, enter the user's name in the **`Name`** field. The other fields on this tab are optional.

   e On the **`Logins`** tab, perform these steps:

      □ Add a login that the metadata server can use to determine the user's metadata identity. This login must contain the fully qualified form of the user ID for the primary account that you created in step 1a. If this login will be used to provide access to other servers from applications that do not cache credentials, include the password and specify the DefaultAuth authentication domain.

      □ Add other logins as needed to support additional authentication. These logins would contain the credentials for any additional individual user accounts that you created in step 1.

   f On the **`Groups`** tab, define the user's group memberships. Each user can belong to multiple groups.

   g Click **`OK`** to save and close the user definition.

   Note: By default, only *administrative users*, *unrestricted users*, and the user who is represented by a particular user definition can make changes to that user definition. △

How to Designate an Unrestricted, Administrative, or Trusted User

Unrestricted, *administrative*, and *trusted users* are users who have special status on the metadata server. You can assign special user status listing user IDs in the

adminUsers.txt file or the **trustedUsers.txt** file. By default, these files are located in the **MetadataServer** directory under the **SASMain** configuration directory. The metadata server recognizes three types of special users:

administrative user
: a user ID that can perform tasks such as creating repositories, creating user definitions, stopping the metadata server, refreshing repositories, and enabling Applications Response Management logging. Like a regular user, an *administrative user* is subject to metadata layer access controls. *Administrative users* do not have unrestricted access to metadata.

  A user ID that is listed in the **adminUsers.txt** file without an initial asterisk is an *administrative user*.

trusted user
: a user ID that acquires credentials and acts on behalf of other users in a multi-tier server environment.

  A user ID that is listed in the **trustedUsers.txt** file is a *trusted user*.

unrestricted user
: a user ID that has unrestricted access to all metadata, regardless of any access controls that have been specified. An *unrestricted user* can also perform all of the tasks that an *administrative user* can perform. An *unrestricted user* can overwrite password information, but cannot retrieve passwords from the repository. For this reason, an unrestricted account should be used only within SAS Management Console and only for tasks that do not require retrieval of passwords.

  A user ID that is listed in the **adminUsers.txt** file with an initial asterisk is an *unrestricted user*.

When you make changes to the **adminUsers.txt** file or the **trustedUsers.txt** file, follow these guidelines:

□ Specify one user ID per line.

□ List the fully qualified user ID. For example, for a Windows network account, specify *<Windows-domain>**userID*. For a local account, specify *<machine-name>**userID*.

□ In the **adminUsers.txt** file, distinguish an *unrestricted user* from an *administrative user* by prepending an asterisk to the user ID that has unrestricted status. For example, ***sasadm@netdomain**.

□ In addition to being listed in the appropriate file, these user IDs must be able to authenticate to the metadata server.

□ You must stop and restart the metadata server to make these changes take effect.

How to Remove a User

Administrative users and *unrestricted users* can use User Manager to delete user definitions from a metadata repository. It is recommended that you log on with the *unrestricted user* account (SAS Administrator) in order to delete a user definition. Detailed instructions for deleting user definitions are provided in the online Help for User Manager.

In addition to removing the user definition from the metadata, you might need to remove the account that gives the user access to the metadata server. If you do not remove this account, then the user will still be a member of the PUBLIC group in the metadata. Even after you delete a person's metadata identity and user accounts, that person can still view the public kiosk of the SAS Information Delivery Portal.

Macro for Protecting Group Definitions

Overview of the %MDUGRPAC Macro

You should limit WriteMetadata permission for group definitions, because you must prevent regular users from making accidental or deliberate changes such as deleting a group definition or changing group membership assignments. By default, a group definition is protected only by the settings on the repository ACT, so it is necessary to set additional direct controls to protect these objects.

As an alternative to manually setting permissions on the **`Authorization`** tab of every group definition, you can use the %MDUGRPAC autocall macro (available in the SAS autocall macro library). This macro enables you to centrally manage access to group definitions. The %MDUGRPAC macro locates group definitions that are not protected by direct access controls and associates those definitions to an ACT (by default the ACT is named **`ACT Securing Groups`**).

Note: On the **`Users and Permissions`** tab of the ACT that is created by this macro, there are no default settings. See the numbered list below for recommended settings. △

For example, the following code creates an ACT and associates each unsecured IdentityGroup object in the specified repository to that ACT.

```
options metaserver=mymachine
metaport=9999
metauser='winnt\userid'
metapass='xxxyyyzzz1'
metarepository=Foundation;

%mdugrpac();
```

After you run this macro for first time, you must perform these tasks:

1 On the Users and Permissions tab of **`ACT Securing Groups`**, deny WriteMetadata permission to the PUBLIC group, and then grant this permission back to only those users who maintain your group definitions. For example, you might grant WriteMetadata permission to an Administrators group. Or you might choose not to grant WriteMetadata permission to anyone, which leaves an *unrestricted user* as the only identity that can make changes to group definitions. Do not deny ReadMetadata permission to anyone.

2 On the **`Authorization`** tab of the ACT, set permissions to control who can make changes to the ACT. It would be appropriate to simply deny WriteMetadata permission to PUBLIC, so that only an *unrestricted user* can make changes to this special ACT.

The following topic contains reference information about this macro.

Syntax of the %MDUGRPAC Macro

Here is the syntax for this macro:

```
%mdugrpac();
```

or

```
%mdugrpac(ACTName="Name of the ACT"
          scope= ALL | IMPORTED | NONIMPORTED
```

```
        mode= EXECUTE | REPORT);
```

ACTName

specifies the name of the ACT that will be used to manage access to IdentityGroup objects (group definitions). By default, the macro creates an ACT named **`ACT Securing Groups`** (if that ACT does not already exist). If you want to associate unsecured group definitions to an existing ACT, use this option to indicate which ACT you want to use.

scope

specifies which IdentityGroups can be secured by the ACT. Associations are not created for objects for which direct access controls are already defined. The default scope is ALL.

ALL	All IdentityGroups in the foundation repository are candidates for inclusion.
IMPORTED	Only IdentityGroups that were created by a bulk-load process are candidates for inclusion.
NONIMPORTED	Only IdentityGroups that were not created by a bulk-load process are candidates for inclusion.

mode

controls the behavior of the macro. The default mode is EXECUTE.

EXECUTE	causes the macro to create the associations and the new ACT (if it does not already exist), and to generate a list of the changes that were made.
REPORT	causes the macro to only generate a list of the IdentityGroups that would be modified if you execute the macro.

CHAPTER 7

User ID and Password Management

About the User ID and Password Management Tasks

In the SAS Intelligence Platform, all user IDs and passwords are for accounts with external systems. For this reason, user ID and password management activities for the SAS Intelligence Platform are driven by changes that occur in these external systems. For example:

- After you create an account for a user in a database system, you must store that user ID and password in the metadata to support single sign-on access to the database server from a SAS application.
- When the password for an external user account is stored in the metadata repository to support single sign-on, the stored password must match the actual password. Any change to the actual password in the external authentication provider must be followed by an update to the stored password.
- Passwords for required accounts (such as sastrust or saswbadm) are stored in additional locations, both in the metadata repository and in configuration files. It is necessary to update the password in all of these locations after the password changes in the operating system (or other authentication provider).

How to Store User IDs and Passwords in the Metadata

You use SAS Management Console's User Manager to store user IDs and passwords. In the metadata repository, user IDs and passwords are stored in login objects as part of a user definition or a group definition.

- To store the user ID and password for a person's individual account, you add that user ID and password on the **Logins** tab of a user definition.
- To store the user ID and password for an account that several users share, you add that user ID and password on the **Logins** tab of a *group* definition. This enables all the members of that group to use those credentials.

The following list explains how logins can be added and removed:

- Each user can add logins to his or her user definition by using either the SAS Personal Login Manager or SAS Management Console. Web server authentication is not available for these applications; both of these applications require the user to have an account with the metadata server's authentication provider.

- □ *Administrative users* and *unrestricted users* can use SAS Management Console to add logins to any user definition.
- □ Any user who has WriteMetadata permission for a group definition (and for the repository) can add logins to that group definition.
- □ A group member who has WriteMetadata permission for a group definition (and for the repository) can remove logins from that group definition.
- □ An *unrestricted user* can add or remove logins from any user or group definition.

Related Topics:

"How Logins Are Used" on page 8

"Which User IDs and Passwords Will be Stored in the Metadata?" on page 57

How to Update Passwords for Users

Users can use SAS Management Console or the SAS Personal Login Manager to update their stored passwords. In order to use either of these desktop applications, the user must have an account with the metadata server's authentication provider. An *unrestricted user* can use SAS Management Console to reset a password for any user. The *unrestricted user* can overwrite the existing password but cannot view the password.

How to Update Passwords for Required Accounts

This topic documents the tasks that you must perform when you change the operating system passwords for the required accounts (such as SAS Administrator, SAS Trusted User, SAS General Server User, SAS Web Administrator, and SAS Guest User).

Note: For SAS Financial Management and SAS Strategic Performance Management, do not use these instructions. Instead, use the instructions that are provided with the password utility that is available for those solutions. △

To update the passwords for the required accounts, complete these steps:

1 Stop all SAS server sessions and services other than the SAS Metadata Server.

2 Reset the passwords in the operating system (or other authentication provider).

3 Update the passwords that are stored in the metadata repository. Log on to SAS Management Console as an *unrestricted user* (sasadm) in order to make these changes.

   a Update the password in the sassrv login by navigating to **User Manager ▶ SAS General Servers Group ▶ Logins**. To access the `Password` and `Confirm Password` fields, select a login and click `Modify`.

   b If you have SAS Web Report Studio, update the password for the account that provides access to the WebDAV content server (by default this is the saswbadm account). To access the `Password` and `Confirm Password` fields, navigate to **Environment Management ▶ BI Manager ▶ BIP Tree ▶ Properties ▶ General**.

   c If you have SAS Web Report Studio and you are using pooled workspace servers, update the password for the pool administrator (by default, this is the sastrust account). To access the `Password` and `Confirm Password` fields, navigate to **Foundation Services Manager ▶ Query and Reporting**

Services ▶ BIP Core Services ▶ User Service ▶ Properties ▶ Service Configuration. On the **Users** tab of the User Service Configuration dialog box, select the sastrust account and click **Edit**.

d If you have the SAS Information Delivery Portal and you are using pooled workspace servers, update the password for the pool administrator (by default, this is the sastrust account). To access the **Password** and **Confirm Password** fields, navigate to **Foundation Services Manager ▶ ID Portal Local Services ▶ BIP Local Services OMR ▶ User Service ▶ Properties ▶ Service Configuration**. On the **Users** tab of the User Service Configuration dialog box, select the sastrust account and click **Edit**.

4 Update the passwords that are stored in the file system. In this step you are working with passwords that are encrypted or encoded. Always overwrite the *entire* password, including the encoding-type indicator **{sasXXX}** and any trailing equal signs (**=**).

a Encrypt the new passwords using SAS proprietary 32-bit encryption. For example, to encrypt a password of "SAStrust1" you would submit this code in the SAS Program Editor:

```
proc pwencode in='SAStrust1' method=sasenc;
run;
```

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

b In the configuration files for your deployment, replace the old encrypted passwords with the new encrypted passwords. “Configuration Files That Include Passwords” on page 100 lists the changes that are needed for each account.

c If you have programs that include these passwords and that do not prompt for credentials, replace the old encrypted passwords with the new encrypted passwords in those files. For example:

- □ Scheduled jobs that you create include passwords. These jobs must be updated and redeployed after a password is updated. By default, these jobs are stored in **Lev***N***/<***server-context***>/SASEnvironment/SASCode/Jobs**.
- □ The replication programs that you create to make backups include passwords.

5 If you are using the LSF schedule server with SAS Web Report Studio, complete these steps after you change the operating system password for the account that you are using as the LSF user:

a Update the password in the LSF server by using the lspasswd command.

b In the $LSF_PASSWORD$ field of the **wrs.config** file, enter the new password.

6 Reconfigure and redeploy your SAS Web applications. It is not necessary to re-import the local service deployment files after you update passwords.

7 Restart the SAS server sessions and services, following these guidelines:

- □ Start the metadata server before starting other SAS servers, spawners, and the SAS Services Application.
- □ Start the SAS Services Application and your WebDAV server before starting your servlet container or J2EE application server.

Configuration Files That Include Passwords

The information in the following tables will help you complete step 4 in the previous topic. The tables list the standard location and name for each configuration file that must be updated. Context information is provided to indicate where within each file the passwords are located. In each configuration file that is applicable to your deployment, replace the old encrypted password with the new encrypted password. *Not all deployments use all accounts or include all components.*

Table 7.1 Configuration Files That Include the Password for sasadm

Component	Location of Password for the SAS Administrator (sasadm)
SAS Metadata Server (Windows)	Lev1\SASMain\MetadataServer\MetadataServer.bat Context: Set OMAAPW=*password*
Xythos WebFile Server 4.0.48	*xythos-install-location*\wfs-4.0.48\saswfs.properties Context: com.sas.wfs.metadata.pw
Xythos WebFile Server 4.2	*xythos-install-location*\custom\classes\saswfs.properties Context: com.sas.wfs.metadata.pw
SAS Analytics Platform Server*	*install-location*\sasapcore\conf\server.config Context: omr_password=*password*

* As an alternative to manually updating the password for this component, you can run the Analytics Platform Configuration tool (see the *SAS Analytics Platform Administrator's Guide*).

Table 7.2 Configuration Files That Include the Password for sastrust

Component	Location of Password for the SAS Trusted User (sastrust)
SAS Object Spawner	Lev1\SASMain\ObjectSpawner\OMRConfig.xml Context: <Login Name="Metadata Login" UserId="&apos;sastrust&apos;" Password="*password*" />
SAS/CONNECT Server	Lev1\SASMain\ConnectServer\OMRConfig.xml Context: <Login Name="Metadata Login" UserId="&apos;sastrust&apos;" Password="*password*" />
SAS OLAP Server (UNIX)	Lev1/SASMain/OLAPServer/OLAPServer.sh Context: metapass=*password*
SAS OLAP Server (Windows)	Lev1\SASMain\OLAPServer\sasv9_OLAPServer.cfg Context: -metapass "*password*"
SAS Information Delivery Portal	*install-location*\Web\Portal2.0.1\PortalConfigure\install.properties Context: $SERVICES_OMI_USER_PASSWORD$
SAS Web OLAP Viewer 3.1	*install-location*\SASWebOlapViewerforJava\3.1\Configure\install.properties Context: $TRUSTED_USER_ID$
SAS Web Report Studio 3.1	*install-location*\SASQueryandReportingServices\3.1\OutputManagementConfigTemplate.xml Context: <Password>*password*</Password>

Component	Location of Password for the SAS Trusted User (sastrust)
SAS Marketing Automation (WebLogic)	Lev1\web\webapps\exploded\sas.analytics.crm.ma.webapp.war\ WEB-INF\web.xml Context: <param-name>OmrServerPassword</param-name><param-value> *password*</param-value>
SAS Marketing Automation (WebSphere)	WAS_HOME\AppServer\installedApps\<machine-name>\ sas_analytics_crm_ma_webapp_war.ear\sas.analytics.crm.ma.webapp.war\ WEB-INF\web.xml Context: <param-name>OmrServerPassword</param-name><param-value> *password*</param-value>
SAS BI Web Services for Java	*install-location*\Web\WebServicesforJava\1.0\Configure\install.properties Context: $SERVICES_OMI_USER_PASSWORD$
Xythos WebFile Server 4.0.48	*xythos-install-location*\wfs-4.0.48\saswfs.properties Context: com.sas.wfs.metadata.trpw
Xythos WebFile Server 4.2	*xythos-install-location*\custom\classes\saswfs.properties Context: com.sas.wfs.metadata.trpw

Table 7.3 Configuration Files That Include the Password for sassrv

Component	Location of Password for the SAS General Server User (sassrv)
SAS/SHARE Server (UNIX)	Lev1/SASMain/ShareServer/ShareServer.sh Context: metapass=*password*
SAS/SHARE Server (Windows)	Lev1\SASMain\ShareServer\sasv9_ShareServer.cfg Context: -metapass "*password*"
SAS Information Delivery Portal	*install-location*\Web\Portal2.0.1\PortalConfigure\install.properties Context: $GENERAL_SERVERS_PASSWORD$
SAS Marketing Automation (WebLogic)	Lev1\web\webapps\exploded\sas.analytics.crm.ma.core.ear\ sas.analytics.crm.ma.core.jar (extract ejb-jar.xml) Context: <env-entry-name>MAConfig/OmrServerPassword</ env-entry-name> ... <env-entry-value>*password*</env-entry-value>

Component	Location of Password for the SAS General Server User (sassrv)
SAS Marketing Automation (WebSphere)	WAS_HOME\AppServer\installedApps\machine-name\Marketing Automation 4.1.ear\sas.analytics.crm.ma.core.jar (extract ejb-jar.xml) Context: <env-entry-name>MAConfig/OmrServerPassword</ env-entry-name> ... <env-entry-value>*password*</env-entry-value>
SAS BI Web Services for Java	*install-location*\Web\WebServicesforJava\version\Configure\install.properties Context: $GENERAL_SERVERS_PASSWORD$

Table 7.4 Configuration Files That Include the Password for saswbadm

Component	Location of Password for the SAS Web Administrator (saswbadm)
SAS Information Delivery Portal	*install-location*\Web\Portal2.0.1\PortalConfigure\install.properties Context: $PORTAL_ADMIN_PASSWORD$
SAS Web Report Studio 2.1 or 3.1	*install-location*\SASWebReportStudio*version*\wrs.config Context: $WEB_ADMIN_PASSWORD$ Context: $SERVICES_OMI_USER_PASSWORD$
SAS Web Report Viewer 2.1 or 3.1	*install-location*\SASWebReportViewer*version*\wrv.config Context: $WEB_ADMIN_PASSWORD$ Context: $SERVICES_OMI_USER_PASSWORD$
SAS Web Report Studio 1.1	*install-location*\SASWebReportStudio\9.1\wrs.config Context: $SERVICES_OMI_USER_PASSWORD$
SAS Web Report Viewer 1.1	*install-location*\SASWebReportViewer\9.1\wrv.config Context: $SERVICES_OMI_USER_PASSWORD$
SAS Web OLAP Viewer	*install-location*\SASWebOlapViewerforJava*version*\Configure\install.properties Context: $SERVICES_OMI_USER_PASSWORD$
SAS BI Web Services for Java	*install-location*\Web\WebServicesforJava\version\Configure\install.properties Context: $PORTAL_ADMIN_PASSWORD$

Table 7.5 Configuration Files That Include the Password for sasguest

Component	Location of Password for the SAS Guest User (sasguest)
SAS Information Delivery Portal	*install-location*\Web\Portal2.0.1\PortalConfigure\install.properties Context: $PORTAL_GUEST _PASSWORD$
SAS Web Report Studio 2.1*	*install-location*\SASWebReportStudio\2.1\config\WebReportStudioProperties.xml.orig Context: <pw> for publicUserSurrogate
SAS Web Report Studio 3.1*	*install-location*\SASWebReportStudio\3.1\wrs.config Context: $PUBLIC_USER_SURROGATE_PW$
SAS Web Report Viewer 2.1*	*install-location*\SASWebReportViewer\2.1\config\WebReportViewerProperties.xml.orig Context: <pw> for publicUserSurrogate

Component	Location of Password for the SAS Guest User (sasguest)
SAS Web Report Viewer 3.1*	*install-location*\SASWebReportViewer\3.1\wrv.config Context: $PUBLIC_USER_SURROGATE_PW$
SAS BI Web Services for Java	*install-location*\Web\WebServicesforJava\version\Configure\ install.properties Context: $PORTAL_GUEST_PASSWORD$

* Applicable only if you are using sasguest as a surrogate public user. If the password is in the application's LocalProperties.xml file, update the password in that location instead.

PART 4

Access Management

CHAPTER

8

Using the Metadata Authorization Layer

About the Access Management Tasks

Immediately after installation, the protections that are described in “Protecting the Foundation Repository” on page 67 should have been set. When you no longer need a severely restricted environment, you can expand access to support a fully functional deployment. Typically, this expansion happens after you have added users and set some specific metadata layer controls. This chapter explains how to expand and manage access and provides some simple examples of using access controls.

How to Manage ReadMetadata and WriteMetadata Access

Repository ACT Settings for ReadMetadata and WriteMetadata Permissions

Almost all users in the SAS Intelligence Platform need global ReadMetadata and WriteMetadata permissions to the foundation repository. Granting these permissions at a lower level (such as on a folder or a specific resource) is not a sufficient alternative. For example, even if you have WriteMetadata permission to a report folder, you cannot add a report to that folder unless you also have WriteMetadata permission to the repository.

To establish typical repository ACT settings for the ReadMetadata and WriteMetadata permissions, complete these steps in SAS Management Console:

1 In the navigation panel, select **Environment Management ▶ Authorization Manager ▶ Access Control Templates ▶ Default ACT**.

   Note: In SAS Management Console, the repository ACT is represented by a blue cylinder icon and is named "Default ACT" by default. △

2 From the menu bar, select **File ▶ Properties** to open the Default ACT Properties dialog box. Then select the **`Users and Permissions`** tab.

   CAUTION:

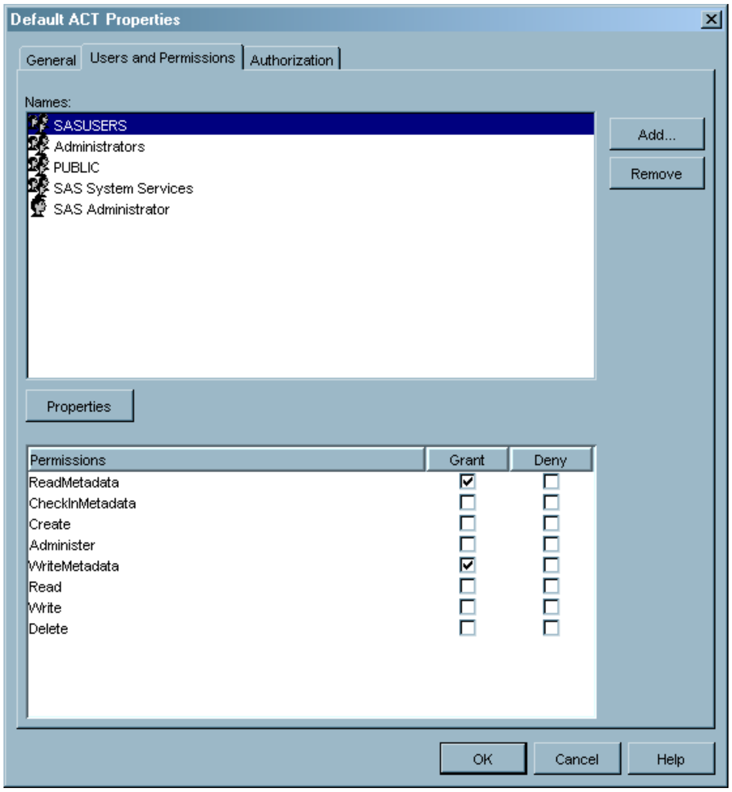
   The Users and Permissions tab looks very similar to the Authorization tab. When you want to set default controls for a repository, be sure that you are on the **`Users and Permissions`** tab. △

3 On the **`Users and Permissions`** tab, click **`Add`**.

4 In the Add Users and/or Groups dialog box, move the SASUSERS group to the **`Selected Identities`** list and then click **`OK`**.

5 On the **`Users and Permissions`** tab, select **`SASUSERS`** in the **`Names`** list and grant ReadMetadata and WriteMetadata permissions to that group.

Display 8.1 Repository Settings for SASUSERS



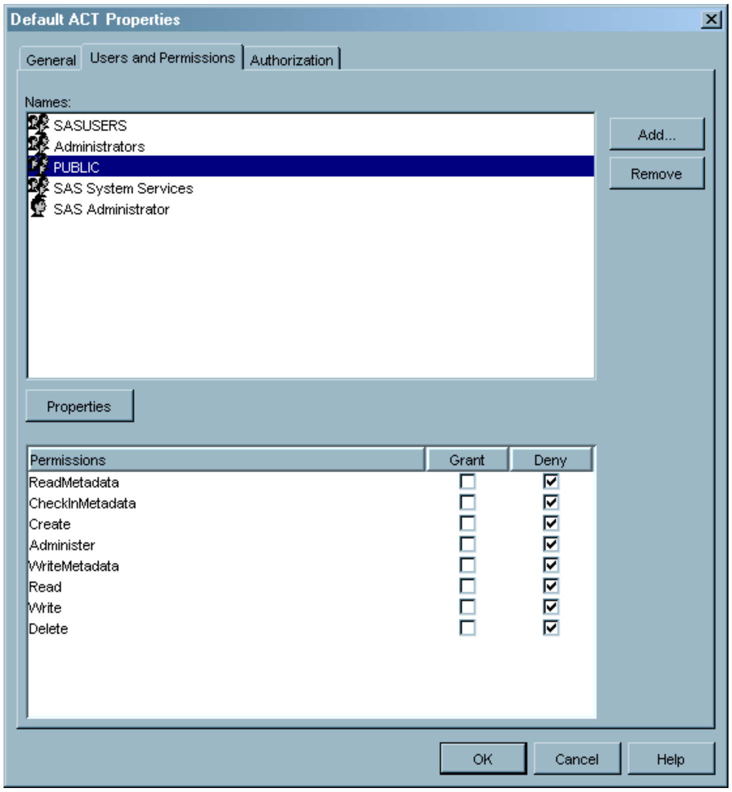
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	□	□
Create	□	□
Administer	□	□
WriteMetadata	☑	□
Read	□	□
Write	□	□
Delete	□	□

6 On the **`Users and Permissions`** tab, select **`PUBLIC`** in the **`Names`** list and verify that these permissions are denied.

Display 8.2 Repository Settings for PUBLIC



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☐	☑
CheckInMetadata	☐	☑
Create	☐	☑
Administer	☐	☑
WriteMetadata	☐	☑
Read	☐	☑
Write	☐	☑
Delete	☐	☑

7 Click **OK** to save the changes.

With these settings, every user who has a metadata identity has a default grant of ReadMetadata and WriteMetadata for the entire repository. Here are some possible variations on these settings:

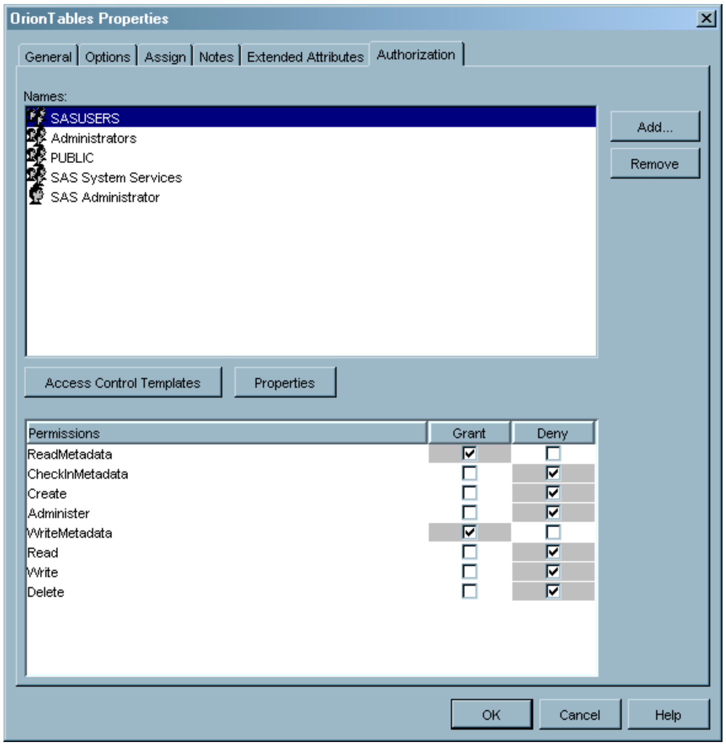
- You can choose to be more liberal and grant ReadMetadata permission (and even WriteMetadata permission) to the PUBLIC group, which includes everyone who can access the metadata server—regardless of whether they have metadata identities.
- You can choose to be more restrictive and grant WriteMetadata permission to a user group that you create. Users who are not members of that group will have very limited capabilities.
- Users of SAS Data Integration Studio who are working in a change-managed environment should have CheckInMetadata permission rather than WriteMetadata permission. This enables these users to interact with resources through a change-managed process and prevents these users from directly modifying resources in the foundation repository.

With the necessary broad grants in place, you must manage ReadMetadata and WriteMetadata access by exclusion; you have to set denials of these permissions on individual resources or containers of resources. The following examples demonstrate how you can set specific denials. The examples assume that you have the typical repository ACT settings that are described in this topic.

Example: Preventing Most Users from Viewing a Particular Library Definition

The goal in this example is to prevent everyone other than administrators from viewing a library definition. To do this, you will broadly deny ReadMetadata permission on the library definition and then grant ReadMetadata permission back to the Administrators group. To begin, navigate to the library definition under the **Data Library Manager** in SAS Management Console and access the properties dialog box for the library. On the library's **Authorization** tab, examine the default settings for the SASUSERS group. As you might expect, the SASUSERS group has an inherited grant of ReadMetadata permission for the library, as depicted in the following display.

Display 8.3 Initial Library Definition Settings for SASUSERS



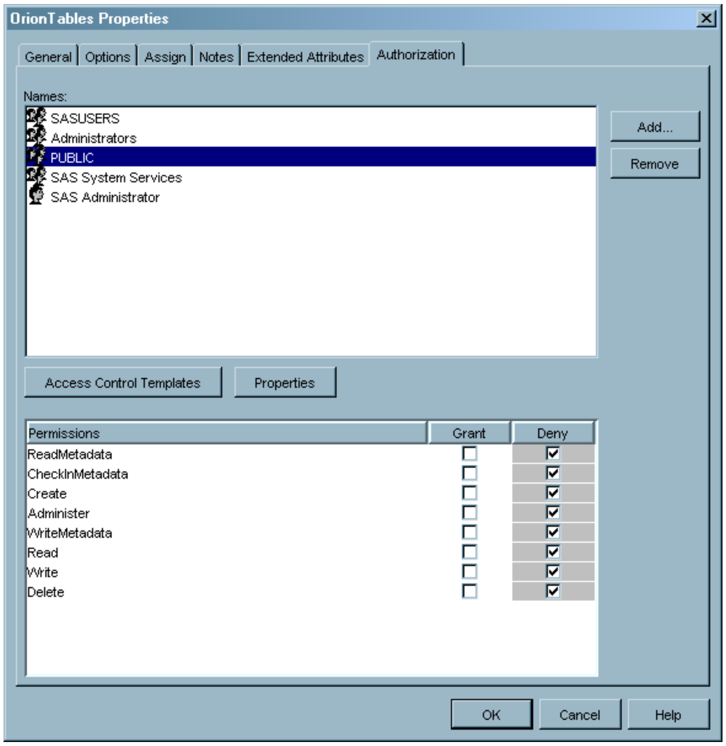
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	□	☑
Write	□	☑
Delete	□	☑

You could simply check the ReadMetadata **Deny** check box to add an explicit denial of this permission for the SASUSERS group. However, this control would not apply to users who do not have individual metadata identities (because those users are not members of the SASUSERS group). Even though those users currently have an inherited denial (from the Default ACT setting for the PUBLIC group), it is a good practice to also include those users in any specific protections that you create. For this reason, we recommend that you assign your blanket denials to the PUBLIC group. On the **Authorization** tab, examine the settings for the PUBLIC group. You will see the inherited denial of ReadMetadata permission.

Display 8.4 Initial Library Definition Settings for PUBLIC



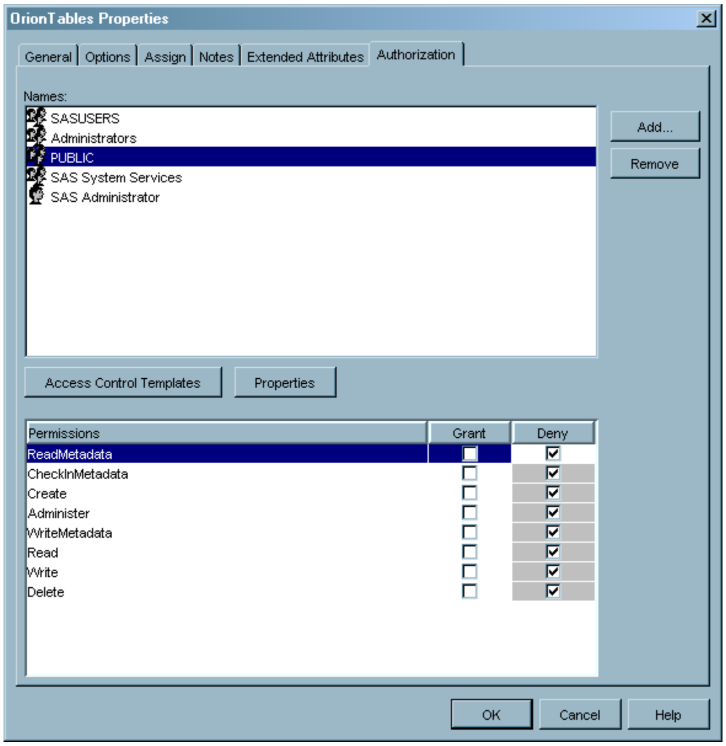
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	□	☑
Write	□	☑
Delete	□	☑

Click the (already checked) ReadMetadata **`Deny`** check box to add an explicit denial. The explicit denial will have a white background color, as depicted in the following display.

Display 8.5 Revised Library Definition Settings for PUBLIC



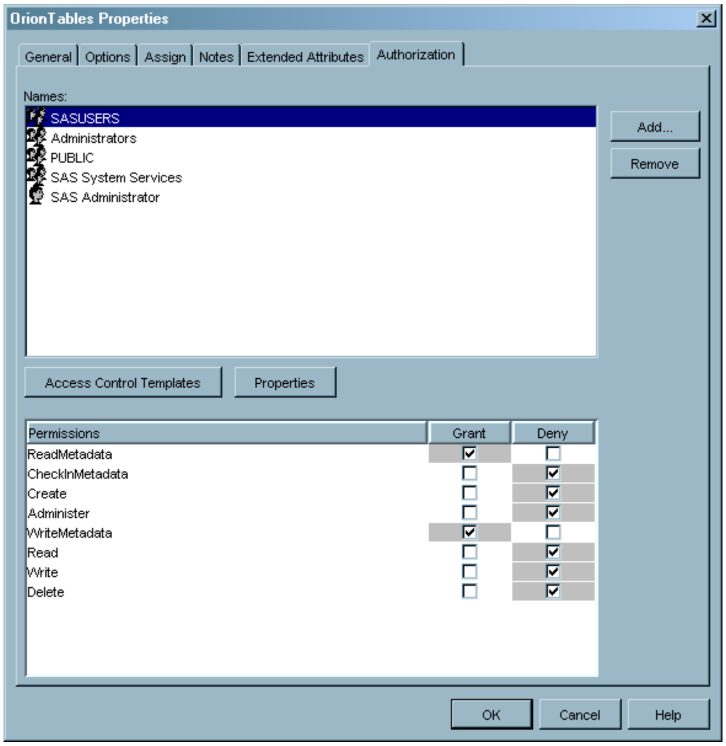
Names:

SASUSERS
Administrators
PUBLIC
SAS System Services
SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☐	☑
CheckInMetadata	☐	☑
Create	☐	☑
Administer	☐	☑
WriteMetadata	☐	☑
Read	☐	☑
Write	☐	☑
Delete	☐	☑

The denial that you set for PUBLIC applies to members of SASUSERS because all members of SASUSERS are also members of PUBLIC. Since the denial is set directly on the library definition, the denial overrides any grants that come from the Default ACT. However, if you examine the SASUSERS permissions list again, you will note that there is no visible evidence of the direct denial. Only the inherited grant is displayed, as depicted in the following display.

Display 8.6 Revised Library Definition Settings for SASUSERS



Names:

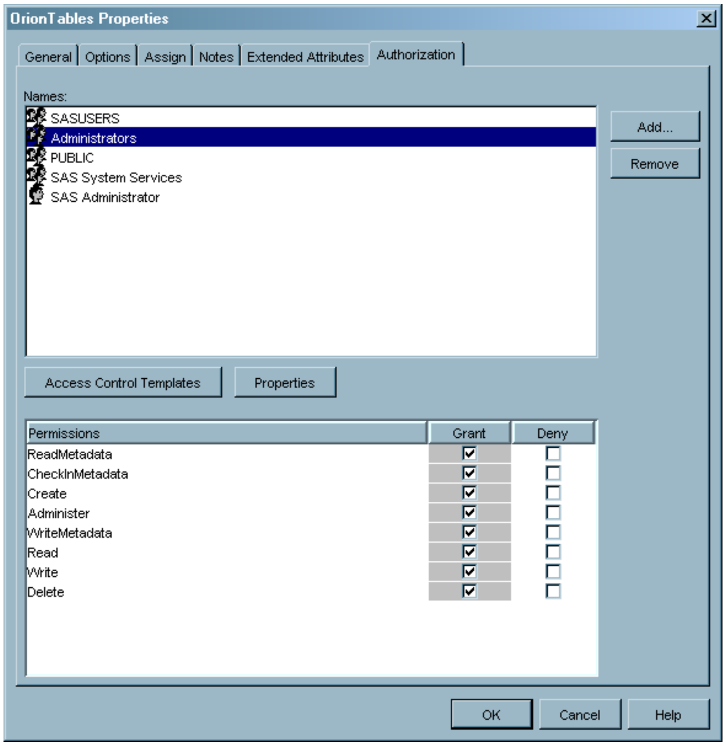
- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	□	☑
Write	□	☑
Delete	□	☑

In fact, members of SASUSERS cannot view the library definition. In the current release, the **`Authorization`** tab does not reflect effective permissions when there are direct controls that apply to an identity because of the identity's group memberships.

Now that you have an explicit, blanket denial of ReadMetadata permission for the library, you need to enable your Administrators group to view the library definition. When you examine the settings for the Administrators group on the library's **`Authorization`** tab, you will see the inherited grant of ReadMetadata permission, as depicted in the following display.

Display 8.7 Initial Library Definition Settings for Administrators



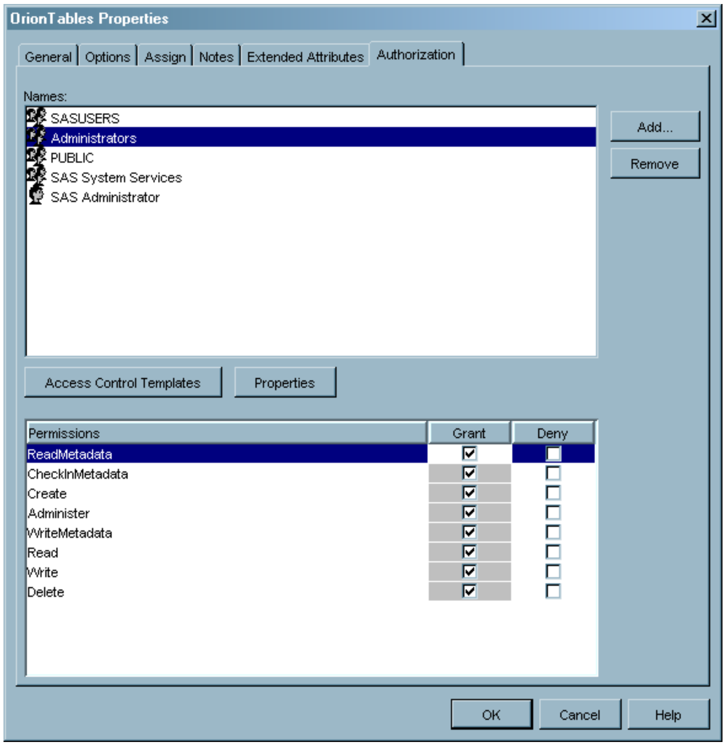
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

You know that there is a direct denial for PUBLIC that defeats this inherited grant (as explained above, the **Authorization** tab does not depict this fact). To enable the Administrators to escape the blanket direct denial, give them a direct grant. To add the direct grant, click the (already checked) ReadMetadata **Grant** check box. The check box should now have a white background, as depicted in the following display.

Display 8.8 Revised Library Definition Settings for Administrators



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	☐
CheckInMetadata	☑	☐
Create	☑	☐
Administer	☑	☐
WriteMetadata	☑	☐
Read	☑	☐
Write	☑	☐
Delete	☑	☐

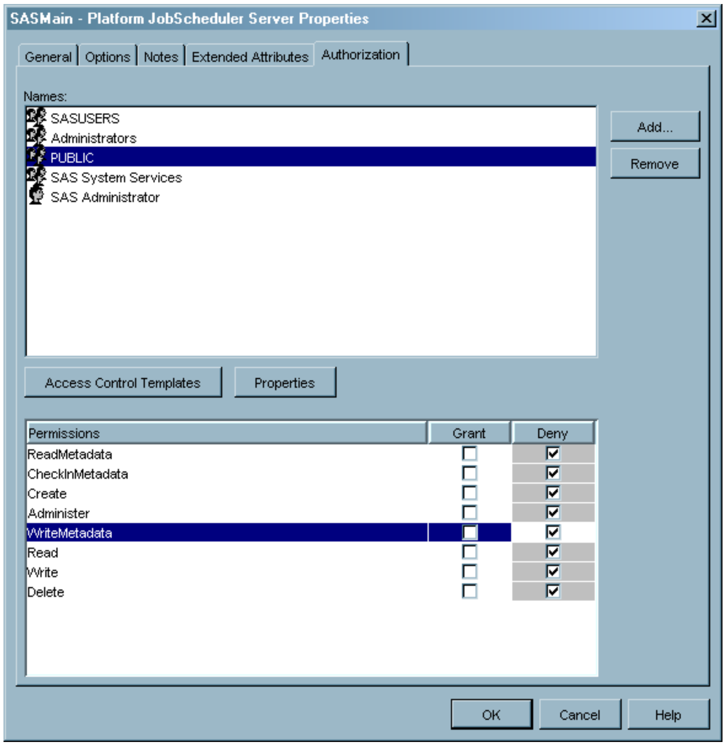
Example: Preventing Most Users from Modifying a Particular Server Definition

The goal in this example is to prevent users from modifying or deleting the server definition for the scheduling server. To do this, you will broadly deny WriteMetadata permission to the server definition and then grant WriteMetadata permission back to the Administrators group. The process in this example is very similar to the process in the previous example, so some of the details have been omitted. To begin, navigate to the scheduling server definition, which is located directly under the **`Server Manager`** in SAS Management Console.

Note: Denying WriteMetadata access to a server definition prevents users from associating resources with that server. This is an important consideration when you deny WriteMetadata permission to any server definition. In this example, users will still be able to schedule jobs, because those resources are associated with the SAS Main application server (rather than the scheduling server). △

As you did in the previous example, you will assign a direct denial to the PUBLIC group, as depicted in the following display.

Display 8.9 Revised Server Definition Settings for PUBLIC



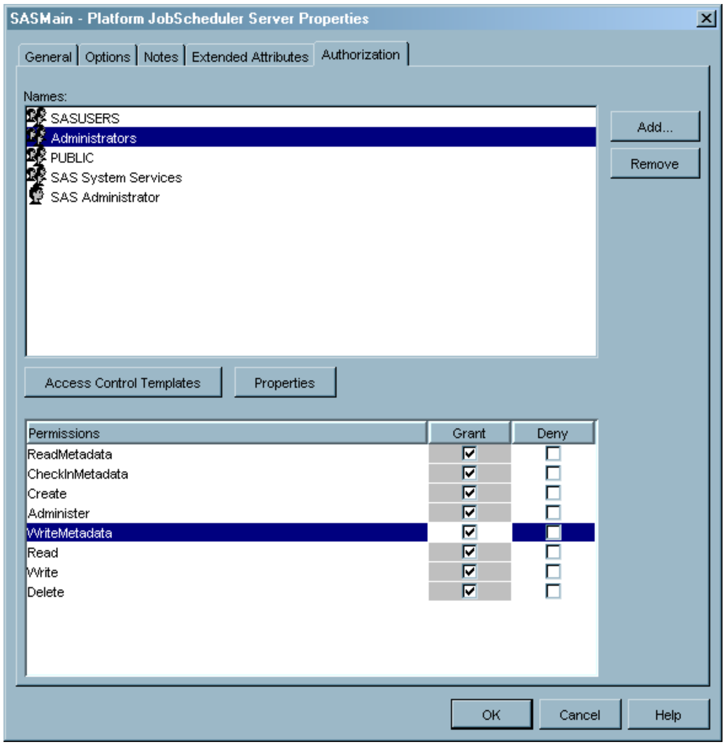
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☐	☑
CheckInMetadata	☐	☑
Create	☐	☑
Administer	☐	☑
WriteMetadata	☐	☑
Read	☐	☑
Write	☐	☑
Delete	☐	☑

Next, restore access for the Administrators group by giving them a direct grant, as depicted in the following display.

Display 8.10 Revised Server Definition Settings for Administrators



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	☐
CheckInMetadata	☑	☐
Create	☑	☐
Administer	☑	☐
WriteMetadata	☑	☐
Read	☑	☐
Write	☑	☐
Delete	☑	☐

Note that these settings (denial of WriteMetadata permission for PUBLIC, grant of WriteMetadata permission for Administrators) are identical to the settings of the ACT that is defined in "Example: Use a Custom ACT" on page 75. As an alternative to setting the individual controls that are described in this example, you could simply apply that custom ACT to this server definition.

Example: Preventing Most Users from Adding Reports to a Particular Folder

The goal in this example is to prevent most users from adding reports to a particular folder. To do this, you will broadly deny WriteMetadata permission to the folder and then grant WriteMetadata permission back to a Power Report Creators group. The process is very similar to the process in the previous example, so some details and all screenshots have been omitted. These are the steps:

1. In SAS Management Console, locate the folder that you want to protect by navigating under **Environment Management ▶ BI Manager ▶ BIP Tree ▶ ReportStudio ▶ Shared ▶ Reports**.
2. On the **Authorization** tab for the folder, set an explicit (white background) denial of WriteMetadata permission for PUBLIC, as you did for the server definition in the previous example.
3. Click **Add** and then add the Power Report Creators group to the **Names** list on the **Authorization** tab. Set an explicit (white background) grant of WriteMetadata permission for this group.

In the absence of additional, more specific controls, these settings create an environment in which only members of the Power Report Creators group can add, modify, or delete folders or reports under this particular folder. If you examine the **Authorization** tab of a subfolder or a specific report, you will see these settings conveyed as inherited grants and denials.

How to Manage Read Access

Introduction to Managing Read Access

While the ReadMetadata permission controls the ability to view metadata objects, the Read permission controls the ability to view the data that is described by those metadata objects. Management of the Read permission differs from management of ReadMetadata and WriteMetadata permissions in these ways:

- The Read permission is relevant only for OLAP data, data that is accessed through the metadata LIBNAME engine, and data that is accessed through information maps.
- In the interest of greater security, Read permission is not granted in the initial configuration. Therefore, before users can perform actions such as querying OLAP cubes or executing reports that are based on information maps, you will need to grant Read permission using an approach that is appropriate for your site. Sites with minimal security requirements can simply grant the Read permission to the PUBLIC group on the repository ACT. Sites with stricter security requirements typically control Read access on specific folders and schemas.
- The Read permission does not have to be granted at the global level (on the foundation repository). This means you can choose whether to manage this permission by exclusion or by inclusion, as explained in the following topics.

Managing Read Access by Exclusion

Introduction to Managing Read Access by Exclusion

You can choose to manage Read access by granting the permission at the repository level and then excluding access for specific resources as appropriate for your environment. For a site that has minimal security requirements, this is the easiest approach. This is a liberal approach because it creates a default grant of Read permission for all SAS OLAP data, all data that is accessed through an information map, and all data that is accessed through the metadata LIBNAME engine. For a site that has stricter requirements, this can still be an acceptable approach, because you can set selective denials on particular resources or sets of resources that you want to protect. However, this approach requires careful, active management to ensure that specific denials are created where they are needed.

Example: Giving Everyone Default Read Permission to Everything

To broadly grant default Read permission to the repository, complete these steps in SAS Management Console:

1 Log on as a user who has WriteMetadata permission to the repository ACT (or as an *unrestricted user*).

2 In the navigation panel, select **Environment Management ▶ Authorization Manager ▶ Access Control Templates ▶ Default ACT**.

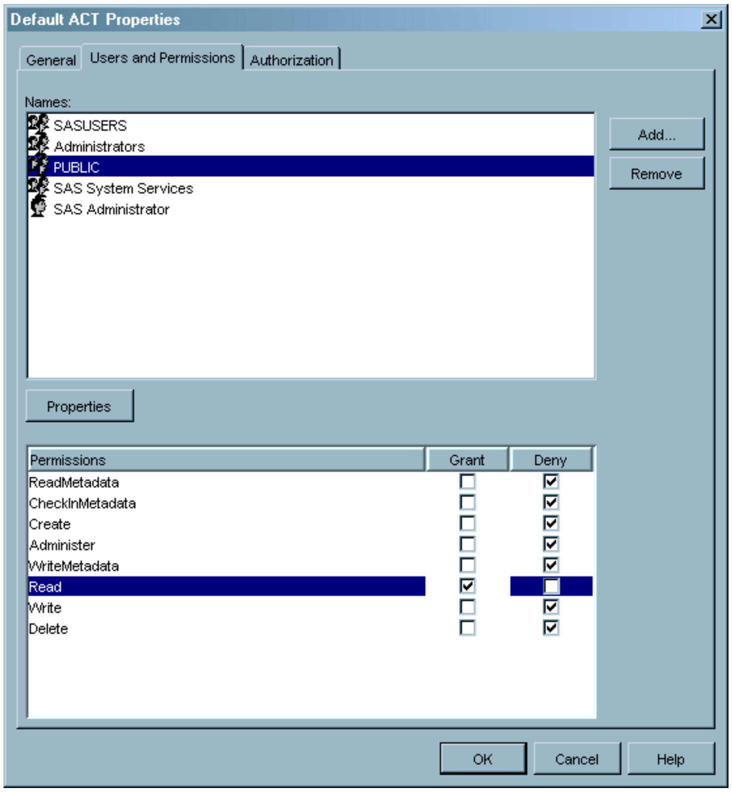
   Note: In SAS Management Console, the repository ACT is represented by a blue cylinder icon and is named "Default ACT" by default. △

3 From the menu bar, select **File ▶ Properties** to open the Default ACT Properties dialog box. Then select the `Users and Permissions` tab.

   CAUTION:
   The Users and Permissions tab looks very similar to the Authorization tab. When you want to set default controls for a repository, be sure that you are on the `Users and Permissions` tab. △

4 On the `Users and Permissions` tab, select `PUBLIC` in the `Names` list and grant Read permission to that group.

Display 8.11 Repository Settings for PUBLIC



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	☑	□
Write	□	☑
Delete	□	☑

5 Check the settings for each of the other identities in the **`Names`** list (especially **`SASUSERS`** if that group is listed) to ensure that the Read permission is not denied.

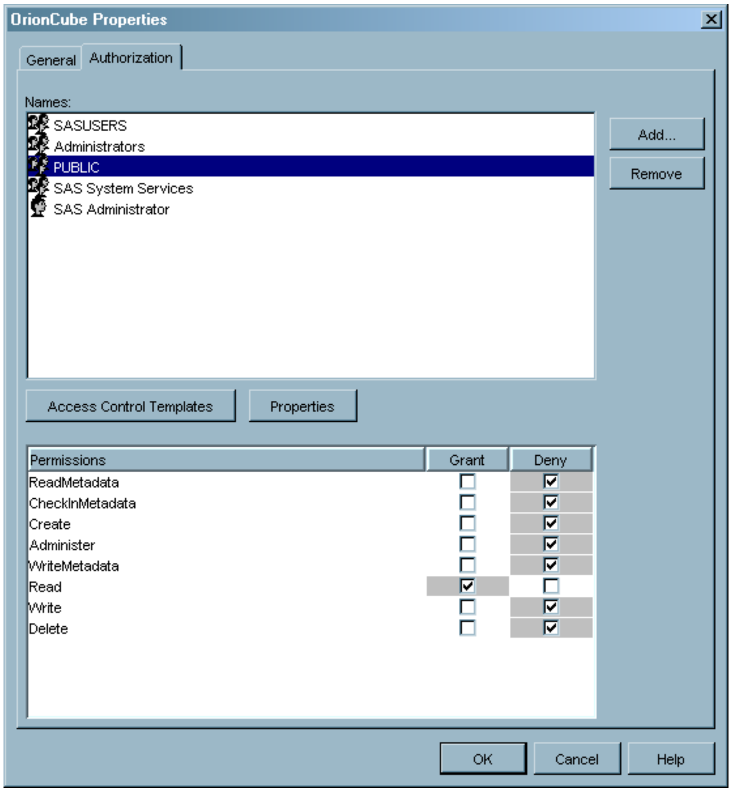
6 Click **`OK`** to save the changes.

Example: Denying Most Users Read Permission to a Particular OLAP Cube

This example assumes that you have the repository settings that are described in the previous example and that your goal is to protect a particular cube. To do this, you will take Read permission for the cube away from PUBLIC and then grant it back to the Administrators group. To set these permissions, complete these steps in SAS Management Console:

1 Navigate to the cube under **Environment Management ▶ Authorization Manager ▶ Resource Management ▶ By Location ▶ <SAS Main> ▶ <SAS Main – OLAP Schema>**. On the cube's **`Authorization`** tab, examine the default settings for PUBLIC. You will see the inherited grant of Read permission that comes from the repository ACT, as depicted in the following display.

Display 8.12 Initial Cube Settings for PUBLIC



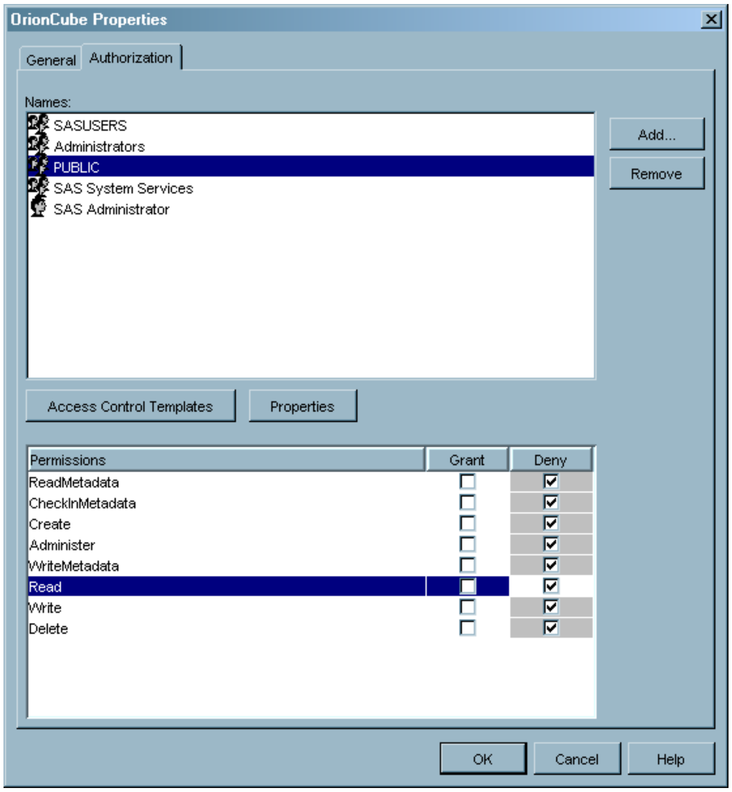
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	☑	□
Write	□	☑
Delete	□	☑

2 Add an explicit denial of Read permission for PUBLIC by checking the Read `Deny` check box. The directly assigned permission will have a white background color, as depicted in the following display.

Display 8.13 Revised Cube Settings for PUBLIC



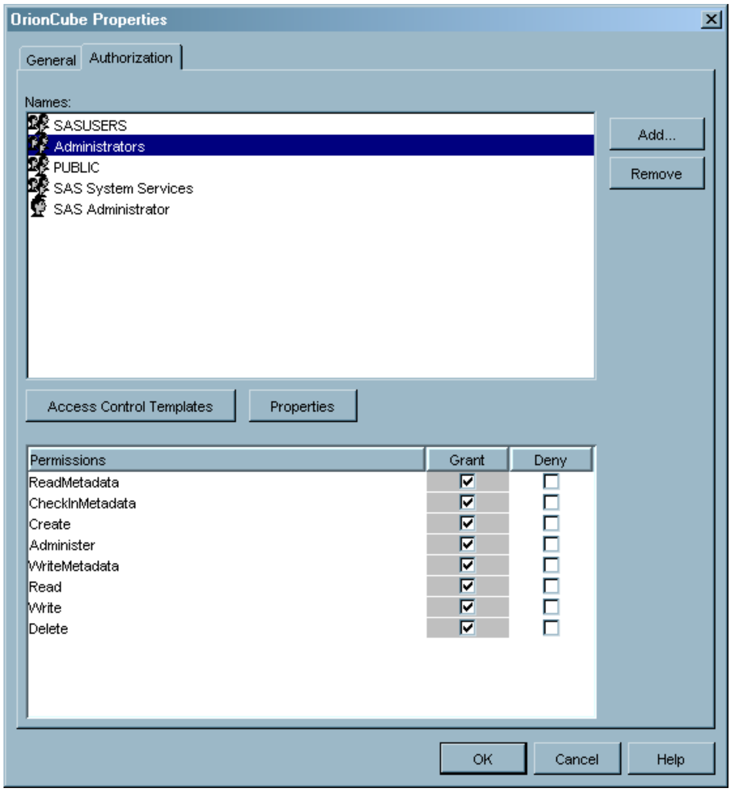
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	□	☑
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	□	☑
Read	□	☑
Write	□	☑
Delete	□	☑

3 Examine the default settings for Administrators on the cube's **`Authorization`** tab. You will see the inherited grant of Read permission.

Display 8.14 Initial Cube Settings for Administrators



Names:

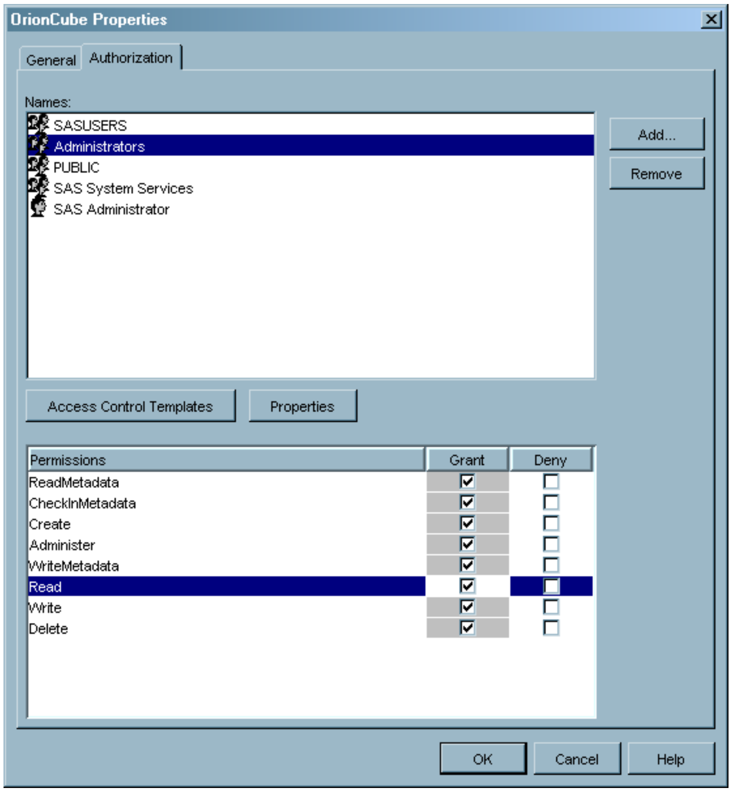
- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

Note: The direct denial of this permission to PUBLIC is not reflected in the display, but it will prevent the Administrators from accessing the data in the cube. △

4 To restore the Administrators access to the cube, you must balance the direct denial that you gave to the PUBLIC group with a direct grant to the Administrators group. To add a direct grant of Read permission, click the (already checked) Read **`Grant`** check box. The directly assigned permission has a white background, as depicted in the following display.

Display 8.15 Revised Cube Settings for Administrators



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	☑	□
Administer	☑	□
WriteMetadata	☑	□
Read	☑	□
Write	☑	□
Delete	☑	□

Managing Read Access by Inclusion

Introduction to Managing Read Access By Inclusion

As an alternative to managing by exclusion, you can set a repository-level denial of this permission and then selectively grant it as needed on specific resources or sets of resources. This can involve more work, but it does have the advantage of making decisions about opening up Read access to particular resources more explicit.

Example: Denying Most Users Default Read Permission to Everything

Repository-level denials of the Read permission should already be set, with the possible exception of the Administrators group. Examine the settings on the **`Users and Permissions`** tab of the repository ACT to verify that Read access is denied to PUBLIC and is not granted to any identities other than the Administrators group.

Example: Granting Most Users Read Permission to a Set of Information Maps

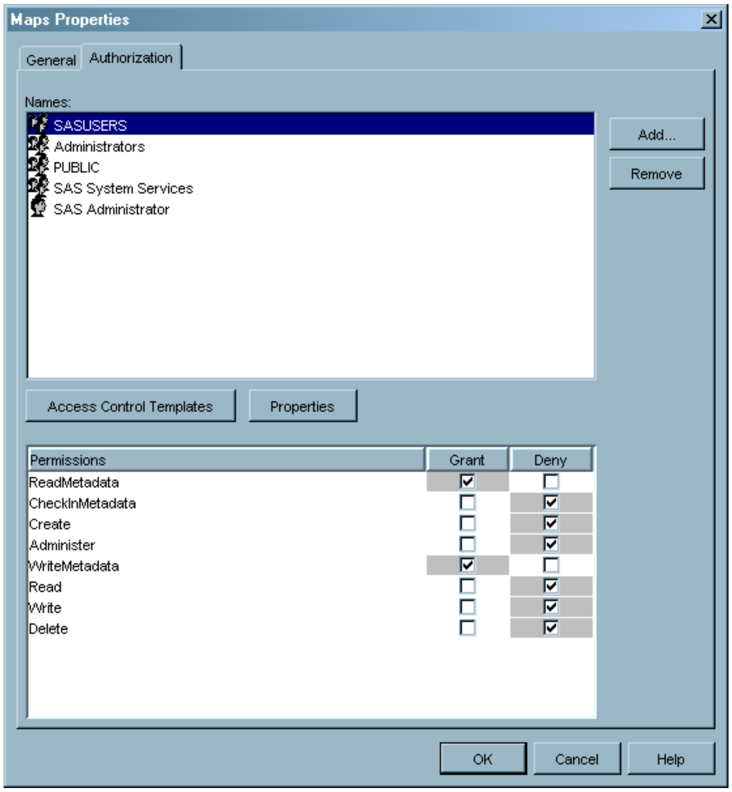
If you have the repository-level denials of Read permission that are described in the previous example, then you will have to grant this permission on certain resources as appropriate for your site. In this example, the goal is to enable all users who have a metadata identity to view reports that are based on information maps. To do this, you will grant Read permission at the top of the folder tree that contains information maps that are used by SAS Web Report Studio. To begin, navigate to that folder in SAS

Management Console by selecting this path: **Environment Management ▶ Authorization Manager ▶ Resource Management ▶ By Application ▶ BIP Tree ▶ Report Studio ▶ Maps**.

Note: This is the standard top-level folder for information maps that are used by SAS Web Report Studio. Information maps can be stored in other locations. △

In the properties dialog box for the Maps folder, access the **`Authorization`** tab and examine the default settings for **`SASUSERS`**. You will see the inherited denial of Read permission that comes from the repository ACT, as depicted in the following display.

Display 8.16 Initial Folder Settings for SASUSERS



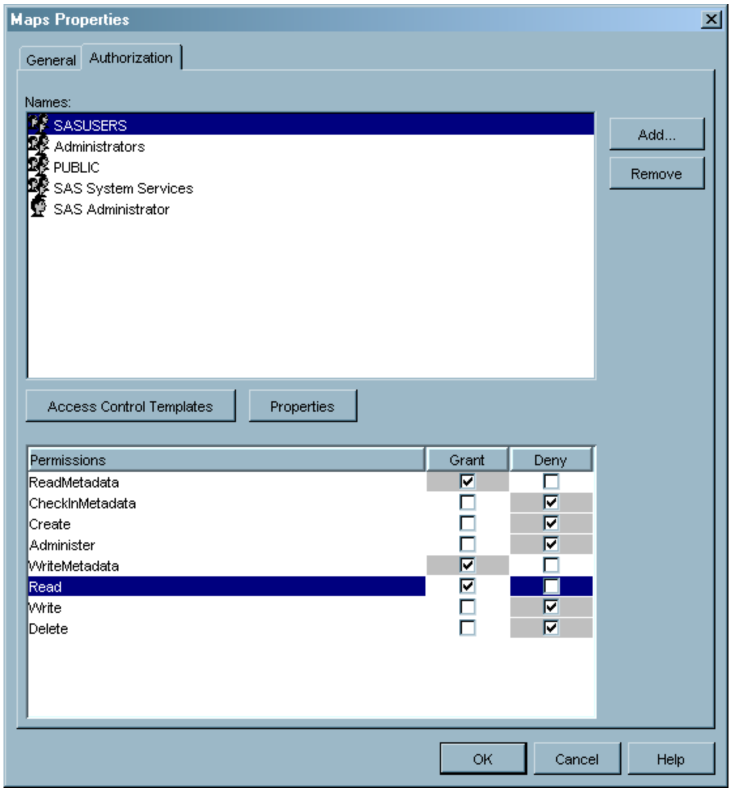
Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	□	☑
Write	□	☑
Delete	□	☑

To enable SASUSERS to read data through the information maps in this folder tree, click the Read **`Grant`** check box. This adds a direct grant with a white background, as depicted in the following display.

Display 8.17 Revised Folder Settings for SASUSERS



Names:

- SASUSERS
- Administrators
- PUBLIC
- SAS System Services
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	□	☑
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	☑	□
Write	□	☑
Delete	□	☑

In this example, you are not granting the Read permission on each individual information map. Instead, you are relying on the fact that the information maps will inherit the grant through the folder tree. For this reason, if there is an explicit denial of the Read permission on a lower folder or on a specific information map, then you will have to grant the permission at that level (or remove the denial) if you want to enable access. Using access controls that are inherited through the folder structure to manage access to information maps is an appropriate and efficient technique. This approach also enables you to secure specific folders or information maps without having to explicitly manage access for every item.

How to Manage the Other Permissions

The following list provides recommendations for managing permissions other than ReadMetadata, WriteMetadata, and Read.

□ The Administer permission controls the ability to access the administrative interfaces of certain SAS servers, such as the SAS OLAP Server, the SAS Stored Processes Server, and IOM spawners. In Chapter 4, "Securing a Deployment," on page 63, you granted your administrators this permission at the repository level and denied this permission to the PUBLIC group. No further adjustments are necessary.

□ The CheckInMetadata permission controls the ability to check in metadata from a project repository and check out metadata to a project repository. This permission is applicable only to SAS Data Integration Studio users who are working in a change-managed environment. For more information, see "Setting Up Change

Management” in the chapter “Administering SAS Data Integration Studio” in the *SAS Intelligence Platform: Desktop Application Administration Guide*.

□ Like the Read permission, the Write, Create, and Delete permissions can be managed by inclusion or by exclusion. In the current release, the Write, Create, and Delete permissions are enforced only if the SAS metadata LIBNAME engine is used to access the data. For more information, see “Pre-Assigning Libraries to Use the MLE” in the chapter “Assigning Libraries” in the *SAS Intelligence Platform: Data Administration Guide*.

Tip: Interpreting the Authorization Tab

The permission settings on an **`Authorization`** tab for a resource determine who can access that particular resource. For example, the **`Authorization`** tab for a group definition controls who can view, modify, or delete that definition; these settings do not determine the group’s access to other objects.

To view the access controls for a resource, locate the resource in SAS Management Console and display the Properties dialog box for the resource. On the **`Authorization`** tab, examine the permissions that are assigned to each identity (user or group) that is listed in the **`Names`** list.

The default settings for a resource come from the **`Users and Permissions`** tab of the repository ACT and from any access controls that are conveyed by the resource’s parent objects. These settings have a gray background color. Settings that have been explicitly added on the resource have a white background color (indicating a directly assigned ACE) or a green background color (indicating a directly applied ACT). Permissions that are directly assigned on a resource always have precedence over permissions that are inherited from the repository ACT or other parent objects.

CAUTION:

In the current release, the permissions list does not always display effective permissions. The permissions list does not display direct access controls that apply because of group memberships. For example, if a person has an inherited (gray background) grant of WriteMetadata permission, and the PUBLIC group has a direct (no background color) denial of that permission, then the PUBLIC group’s directly assigned denial overrides the person’s inherited grant. However, when you view the person’s permissions list, there is no visual indication of the PUBLIC group’s directly assigned denial. △

CHAPTER
9

Access Guidelines and Requirements

Guidelines for Managing Access

Effective access control requires an organized approach to setting permissions. For example, an appropriate user group structure will greatly simplify the process of assigning and maintaining permissions. In any environment, access controls that are defined for individual users in an ad hoc fashion quickly become difficult, if not impossible, to manage.

The following measures can enhance the effectiveness of the protections that you establish in the metadata layer:

□ To control access to the computing resources that are represented by metadata objects, grant and deny metadata permissions in pairs. In the current release, the strongest protections come from the ReadMetadata, WriteMetadata, and CheckInMetadata permissions. The following table documents an approach that provides the best protections in the current release and the best compatibility for future releases.

Table 9.1 Recommended Use of Permissions

Action You Want to Control	Permissions to Grant or Deny
Reading a metadata object	ReadMetadata
Reading the data that is described by a metadata object	Read and ReadMetadata
Modifying a metadata object	WriteMetadata
Modifying data that is described by a metadata object	Write and WriteMetadata
Creating a new metadata object	WriteMetadata
Creating new data	Create and WriteMetadata

Action You Want to Control	Permissions to Grant or Deny
Deleting a metadata object	WriteMetadata
Deleting data that is described by a metadata object	Delete and WriteMetadata

- Use other authorization layers, such as operating system permissions and relational database controls, to secure data.
- Understand how permissions are evaluated in the metadata authorization layer. Remember that this layer supports multiple inheritance, and that the inheritance rules make it much easier to establish an effective grant of a permission than to establish an effective denial.
- Use caution when moving objects that inherit permissions from their folders. For example, moving a report or information map from one folder to another might change the effective access controls for that object.
- Remember that your effective permissions are limited to access that is allowed in all applicable authorization layers.
- Remember that managing security is an ongoing process; you will need to define more access controls as you register additional resources in the metadata environment.

An important efficiency goal is to minimize the number of access controls that you have to set and maintain. Tactics that will help you achieve this goal include the following:

- Assign permissions to the highest appropriate user group in the group hierarchy.
- Use access control templates (ACTs) to centralize management of identity/permission patterns that you will apply to multiple resources.
- Assign permissions at the highest appropriate level in the resource inheritance structure.
- Use dedicated folders to manage access to resources. For example, if you have SAS Data Integration Studio, you can use dedicated folders in the ETL custom tree to manage access to the metadata that describes data.
- For permissions other than ReadMetadata and WriteMetadata, consider whether it is more efficient to manage permissions by inclusion or by exclusion.
  - When you manage permissions by inclusion, you begin by denying all access to resources, and then you selectively grant permissions where they are needed. This approach is typically used when you are following the rule of least privilege so that you grant only as much access as is required to do the job.
  - When you manage permissions by exclusion, you begin by granting broad access to resources, and then you selectively deny permissions where there is a need to protect resources or information. This approach is typically used when you are following the rule of least protection.
- When making decisions about your environment, consider the balance between deployability, usability, maintainability, and security.

Access Requirements by Type of Resource

Access Requirements for Server Definitions

The members of the SAS System Services group are used to connect to various servers, so these identities must be able to access the configuration information that is stored in server definitions. During installation, the SAS System Services group is granted ReadMetadata permission to the repository. This gives the SAS System Services group ReadMetadata access to all objects in the repository, including the server definitions.

Do not block the SAS System Services group's access to any server definition. If you choose to limit access to a logical server definition, you might need to set an additional access control to preserve the SAS System Services group's access. For example, you might set these direct access controls on the **`Authorization`** tab of an OLAP server definition:

□ deny the PUBLIC group ReadMetadata permission to the server definition

□ grant ReadMetadata permission back to a user group that accesses data on that server

When you deny ReadMetadata permission to PUBLIC directly on the server definition, the denial overrides the grant to SAS System Services that comes from the repository ACT, so the SAS System Services group will not be able to obtain configuration information about the SAS OLAP Server from the metadata server.

You can remedy this situation by adding a direct grant of ReadMetadata permission to the permissions list for **`SAS System Services`** on the **`Authorization`** tab for the server definition. In the **`Names`** list, select SAS System Services. In the permissions list, the group's repository ACT grant of ReadMetadata permission is indicated by a checked box with a gray background. To add a direct grant on top of the repository ACT grant, select the check box. The gray background is removed and the check box is still selected. This indicates that the SAS System Services group now has a direct grant of ReadMetadata permission to the server definition.

Note: Because the SAS Object Spawner attempts to read server definitions only during initialization, you must stop and restart the spawner after making these changes. △

For an example of managing WriteMetadata permission for a server definition, see "Example: Preventing Most Users from Modifying a Particular Server Definition" on page 115.

Access Requirements for Libraries and Tables

Users who use SAS Data Integration Studio, the metadata LIBNAME engine, or SAS Management Console to define and manage metadata that describes data sources must have permissions that enable them to interact with that metadata. This topic describes the required metadata layer permissions for working with data.

Note: Permissions in other authorization layers are also required for most of these tasks. For example, although no metadata layer permissions to a data source are required in order to register the data source, this task involves reading some information from the source, so some permissions in the data source and operating system authorization layers are usually required. △

The column headings in the following table are the metadata objects that are most frequently involved when you define and use metadata that describes data sources. For each metadata object that is listed in the column headings, the table shows the permissions that are required for a particular task.

Table 9.2 Access Requirements for Common Data Tasks

Task	Foundation Repository	Metadata Object That Describes the Data Source
Create metadata that describes a data source	RM, CheckInM (or WM)*	Not applicable
View the metadata that describes a data source	RM	RM
Modify or delete metadata that describes a data source	RM, CheckInM (or WM)*	RM, CheckInM (or WM)*
View the data within a registered data source	RM	RM, R**

* If you are using SAS Data Integration Studio in a change-managed environment, then you must have CheckInMetadata permission. Otherwise, you must have WriteMetadata permission.

** In the current release, the Read permission is not always required, because not all applications enforce this permission.

In a change-managed environment, the owner of each project repository should also have ReadMetadata and WriteMetadata permissions for the entire project repository. To learn about change management, see “Administering SAS Data Integration Studio” in the *SAS Intelligence Platform: Desktop Application Administration Guide*.

Note: In order to navigate to a metadata object, you must also have ReadMetadata permission to the folder that contains the metadata object, and to all of that folder’s parent folders. If you cannot see a folder, you cannot browse the objects that the folder contains. △

Access Requirements for OLAP Data

This table summarizes the permissions that required users should have in order to support access to SAS OLAP data:

Table 9.3 Supporting Access to OLAP Data

Requirement	Recommended Approach
The SAS Trusted User must have the ReadMetadata permission for all SAS OLAP cubes and schemas.	On the SAS OLAP Server definition, grant the ReadMetadata permission to the SAS System Services group. Be sure to preserve this access. For example, if you directly deny the PUBLIC group ReadMetadata permission to a particular cube, you should directly grant that permission back to the SAS System Services group.
The SAS Guest User must be able to access any SAS OLAP data that is displayed in the public kiosk of the SAS Information Delivery Portal.	On each appropriate resource, grant the Read and ReadMetadata permissions to the SAS Guest User.

Regular users need the following permissions in order to perform common OLAP tasks:

□ ReadMetadata permission for a cube, hierarchy, dimension, level, or measure is required in order to view the metadata for those resources. ReadMetadata permission for schemas and cubes is a prerequisite for accessing data within those resources.

□ The OLAP server requires users to have Read permission for cubes, hierarchies, dimensions, levels, and measures in order to view the data in those resources. If you deny access to a measure that participates in a calculated measure, the calculated measure might produce unintended results.

□ WriteMetadata permission for a schema is required in order to add or remove cubes from that schema. To delete a cube, users must also have WriteMetadata permission for the cube.

□ WriteMetadata permission for a cube is required in order to rebuild, modify, or change the permission settings for that cube.

□ Administer permission for the parent application server (such as SASMain) is required in order to perform tasks such as viewing user sessions, terminating sessions, and refreshing cubes. These tasks are performed by using the SAS OLAP Server Monitor in SAS Management Console.

Related Topics:

"Inheritance in OLAP Data" on page 42

Chapter 11, "OLAP Member-Level Permissions," on page 171

"Example: Denying Most Users Read Permission to a Particular OLAP Cube" on page 119

Access Requirements for Information Maps

Users who define and use information maps must have permissions that enable them to interact with those information maps. An information map is a metadata object that contains a view of one or more data sources, with an added layer of business metadata. Information maps are used as the data sources for reports.

The column headings in the following table are the metadata objects that can be involved when you define and manage information maps. For each metadata object that is listed in the column headings, the table shows the permissions that are required for a particular task.

Table 9.4 Access Requirements for Common Information Map Tasks

Task	Foundation Repository	Folder That Contains the Information Map*	Information Map	Stored Process (Optional)	Data Sources
Create and save an information map	RM, WM	RM, WM	Not applicable	RM	RM, R**
View an information map	RM	RM	RM	RM	RM
Edit and overwrite an existing information map	RM, WM	RM	RM, WM, R	RM	RM, R**
Move an information map to another folder	RM, WM	RM, WM	RM, WM	None	None
Rename or delete an information map	RM, WM	RM, WM	RM, WM	None	None
Run queries using an information map	RM	RM	RM, R	RM	RM, R**

* Users who navigate to an information map must have RM permissions to the folder that contains the information map and to all of that folder's parent folders. If you cannot see a folder, you cannot browse the objects that the folder contains. Users who access an information map by searching do not have to have RM for the folder that contains the information map.

** You must be able to read the data in order to test a query or set a filter value from the data source. In the current release, Read permission for the target data is not always required, because not all applications enforce this permission.

Each information map should be created for a particular set of report creators and report consumers for the following reasons:

□ If you attempt to create a report that includes any column to which you do not have access, then the entire report creation fails.

□ If you attempt to view a report without having access to all of the underlying data sources, then only those report items (such as tables or graphs) that contain data to which you have access are displayed in the report.

Users need Read permission for an information map in order to access data through that information map. This requirement does not replace or override access controls that are set on the data sources; this is an additional requirement. To learn how you can meet this requirement, see "How to Manage Read Access" on page 117.

Access Requirements for Reports

Users who view, create, modify, and delete reports must have permissions that enable them to perform those activities. The access requirements for working with reports vary, depending on the relationship the report has to its underlying data. For example:

- □ *Automatically refreshed* reports run queries to get current data every time the report is accessed. Data in an automatically refreshed report is live data.
- □ *Manually refreshed* reports are generated and cached on a demand basis. Data in a manually refreshed report is static data that can be updated by a user action in the report viewer.

The column headings in the following table are the metadata objects that can be involved when you work with reports in a repository. For each metadata object that is listed in the column headings, the table shows the permissions that are required for a particular task.

Table 9.5 Access Requirements for Common Report Tasks

Task	Foundation Repository	Folder That Contains the Report*	Report	Stored Processes	Information Maps	Data Source
Create and save a new report	RM, WM	RM, WM	Not applicable	RM	RM, R	RM, R**
View a report	RM	RM	RM	RM	RM, R	RM, R**
View a pre-generated report	RM	RM	RM	None	None	None
Edit and overwrite an existing report	RM	RM	RM, WM	RM	RM	RM
Move a report to another folder	RM	RM, WM	RM, WM	None	None	None
Delete or rename a report	RM	RM, WM	RM, WM	None	None	None
Refresh a pre-generated report	RM	RM	RM	RM	RM, R	RM

* Users who navigate to a report must have RM permissions to the folder that contains the report and to all of that folder's parent folders. If you cannot see a folder, you cannot browse the

objects that the folder contains. Users who access a report by searching do not have to have permissions to the folder that contains the report.

** In the current release, the Read permission is not always required, because not all applications enforce this permission.

CAUTION:

Access to certain types of reports does not require access to the underlying stored processes, information maps, or data sources. In addition to securing the underlying components, you should also secure reports—especially reports that contain cached data. △

Each report should be created for a particular audience of report consumers. If a user attempts to view a report without having access to all of the underlying data items, only those objects (such as tables or graphs) that contain data to which the user has access are displayed. In addition to using the metadata authorization layer, consider the other authorization layers (such as operating system permissions, data source controls, or WebDAV controls) when planning security for reports. For more information, see "Managing Access to Reports" in the chapter "Managing SAS Web Report Studio Content and Users" in the *SAS Intelligence Platform: Web Application Administration Guide*.

Access Requirements for Stored Processes

Users who define and use metadata that describes stored processes must have permissions that enable them to interact with that metadata. A stored process is a SAS program that generates output, such as a data set, a table, or a graph. A stored process can be associated with an information map or with a report.

The column headings in the following table are the metadata objects that are involved when you register and manage stored processes. For each metadata object that is listed in the column headings, the table shows the permissions that are required for a particular task.

Table 9.6 Access Requirements for Common Stored Process Tasks

Task	Repository	Folder That Contains the Stored Process*	Server That Hosts the Stored Process	Stored Process	Data Sources
Create metadata that describes a stored process	RM, WM	RM, WM	RM, WM	Not applicable	None
View metadata that describes a stored process	RM	RM	RM	RM	None
Modify metadata that describes a stored process	RM	RM	RM	RM, WM	None

Task	Repository	Folder That Contains the Stored Process*	Server That Hosts the Stored Process	Stored Process	Data Sources
Delete metadata that describes a stored process	RM	RM, WM	RM, WM	RM, WM	None
Run a stored process	RM	RM	RM	RM	RM, R**

* In order to navigate to a stored process, you must have RM permissions to the folder that contains the stored process and to all of that folder's parent folders. If you cannot see a folder, you cannot browse the objects that the folder contains. If you access a stored process by searching, it is not necessary to have ReadMetadata permission for the parent folders.

** In the current release, the Read permission is not always required, because not all applications enforce this permission.

CAUTION:

> **If a stored process runs on a stored process server (or a pooled workspace server), you must set appropriate access controls to secure the stored process.** These stored processes use the account under which the server is running to access data. Because the user's account is not being used to access the data, the user's permissions to the data are not relevant. △

The following table indicates which accounts are used to determine access when a stored process retrieves data. The accounts involved depend on which type of server executes the process and how the target library is assigned.

Table 9.7 Security Considerations for Retrieving Data with a Stored Process

Where the stored process runs	Identity whose physical layer permissions affect access[1]	Identity whose metadata layer permissions affect access
On a standard workspace server	The requesting user[2]	The requesting user's metadata identity
On a pooled workspace server	A puddle account (such as sassrv)	If the target library is pre-assigned with metaautoinit OR accessed through the metadata LIBNAME engine, then metadata access depends on the server's metadata identity. □ The metadata identity for a pooled workspace server is the metadata group to which the puddle account is assigned. □ The metadata identity for a stored process server is the SAS General Servers Group Otherwise, metadata access depends on the requesting user's metadata identity.
On a stored process server	A service account (such as sassrv)	

[1] The relevant account for physical access is the account under which the server runs.

[2] If the standard workspace server runs under a shared account, physical access depends on the permissions that are granted to that shared account.

Access Requirements for Identity and Permission Objects

The metadata authorization layer provides additional protections for user definitions, group definitions, logins, and permission objects.

For user definitions, these constraints apply:

□ Only an *unrestricted user* or an *administrative user* can add and delete user definitions.

□ By default, users can modify their own user definitions. In order to create a new e-mail, phone, or location object, a user must also have WriteMetadata permission on the `Users and Permissions` tab of the repository ACT.

□ By default, users can add, modify, and delete their own logins unless their changes affect the last instance of the user ID that the metadata server used to determine their identity.

□ In order to modify the name, description, or contact information in another person's user definition, a user must have either status as an *administrative user* or an *unrestricted user*, or must have ReadMetadata permission and WriteMetadata permission from a direct access control on that user definition.

□ In order to create additional logins for another user, a user must have status as an *unrestricted user* or an *administrative user*.

For group definitions, these constraints apply:

□ No one can delete or rename the implicit groups (PUBLIC and SASUSERS).

□ Only an *administrative user* or an *unrestricted user* can make changes to an implicit group definition.

□ Any user who has WriteMetadata permission on the `Users and Permissions` tab of the repository ACT can create a new group definition.

□ Only an *unrestricted user* or a user who has ReadMetadata and WriteMetadata permissions for a group definition can modify or delete a group definition.

□ In order to view a login that belongs to a group, a user must have membership in the group, or have ReadMetadata permission from a direct access control on the login, or have status as an *unrestricted user*.

Note: When an *unrestricted user* views logins, the passwords are not displayed. △

□ In order to add a login for a group, a user must have ReadMetadata and WriteMetadata permissions for the group definition, or have status as an *unrestricted user*.

□ In order to modify or delete a login that belongs to a group, a user must have ReadMetadata and WriteMetadata permissions for the group definition and membership in the group, or have status as an *unrestricted user*.

For permission objects, these constraints apply:

□ Only an *unrestricted user* can modify a Permission object.

□ No one can delete a Permission object from the Authorization Manager.

Note: An *unrestricted user* can use the `Delete Metadata` tab of the metadata utility in SAS Management Console to delete a Permission object. △

□ Only an *administrative user* or an *unrestricted user* can create a Permission object within a repository.

CHAPTER 10

BI Row-Level Permissions

About BI Row-Level Permissions

Introduction to BI Row-Level Permissions

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

- You can design and construct row-level filters by using a standard graphical user interface (GUI) within SAS Information Map Studio.
- You can assign row-level filters to specific identities by using the standard authorization GUI for the SAS Intelligence Platform from within SAS Information Map Studio.
- This feature is integrated with other SAS Intelligence Platform administrative functions. BI row-level permissions can be assigned to existing metadata identities, stored in the metadata repository, and evaluated by the SAS Metadata Server's authorization facility.
- This feature is practical for use with large, dimensionally modeled data marts. 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 provides flexibility in several ways:
  - BI row-level permissions work with SAS data sets and third-party relational databases.
  - BI row-level permissions do not require a specific data model.
  - BI row-level permissions can be used with dynamically generated filters. This enables you to make user-specific access distinctions without defining a separate filter for each person.
- This feature enables you to define granular access to third-party data without requiring you to maintain individual user accounts within those database systems.

If you want to use BI row-level permissions to implement row-level security, it is essential to understand these points:

- While BI row-level permissions provide filtering whenever SAS data sets or third-party relational data are accessed through an information map, comprehensive security that incorporates this filtering requires a specific, high-security configuration of SAS Web Report Studio and appropriate coarse-grained operating system or DBMS protections. For more information, see "How to Create a Secure Environment for BI Row-Level Permissions" on page 147.
- While BI row-level permissions offer many advantages for Web-based reporting, not all SAS clients require that users go through information maps in order to access data. If you need row-level security for clients such as SAS Enterprise Guide, you must use access controls in the data source layer. For example, the 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). Some third-party relational data sources can enforce row-level controls for the data that they store.

Prerequisites for BI Row-Level Permissions

BI row-level permissions require these software components:

□ SAS Intelligence Platform (version 9.1.3 with Service Pack 4 or later)

□ SAS Information Map Studio (version 3.1)

To make appropriate use of this feature, you must 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.

Note: Beginning with Service Pack 4, Read permission on an information map is required in order to access data through an information map. To learn how to meet this requirement, see “How to Manage Read Access” on page 117. △

Row-Level Permissions and Identity-Based Filtering

Understanding Row-Level Permissions

Row-level permissions provide an additional refinement of control beyond setting permissions on libraries, tables, and columns. You use row-level permissions to define access to data at a more granular level, specifying who can access particular rows within a table. Row-level permissions are typically used to 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. When row-level permissions are used, there are three possible authorization decision outcomes for a request to view data:

Grant The requesting user can access all rows.

Deny The requesting user cannot access any rows (and will get an error message).

Grant-with-conditions The requesting user can access only those rows that meet specified SQL filtering conditions.

Unlike access controls for tables or columns, row-level permissions are based on filters and rely on target data that is modeled to work with those filters. The following topics describe filtering techniques for row-level permissions and explain how these controls limit the data that is displayed when a report is generated.

Filtering Methods for Row-Level Permissions

You define row-level permissions in filters that you assign to tables within an information map. For example, you can use a filter that compares values in a target table to a specified value. This enables you to implement a rule such as *Joe can see his salary information*. You can also use a filter that compares values in the target data to a value that is dynamically derived based on the identity of each requesting user. This enables you to implement a rule such as *Each user can see his or her own salary information*. "Filters that Use Identity-Driven Properties" on page 141 provides details and examples.

In order to use any filter for security purposes, you must assign the filter as a prefilter. This prevents end users from disabling the filter and ensures that the filter is used to pre-screen the target data before any other criteria are applied. You can assign the filter in either of these ways:

□ Assign the filter as a *general prefilter*. The filter will be applied to every request and processed independently of any metadata layer access controls, serving as an additional layer of restriction. All users are subject to the filter, regardless of group membership or access controls that grant broader access.

□ Assign the filter as an *authorization-based prefilter* for one or more metadata identities. The filter will be evaluated by the authorization facility as a permission condition in coordination with other access controls, so group memberships and identity precedence can affect the outcome. For example, a filter that is assigned to the PUBLIC group can be overridden by an unconditional direct grant of Read permission that is assigned to a particular user. See "Precedence for Row-Level Permission Conditions" on page 145 for more information and examples.

The following table outlines the methods that you can use to set up filtering for security purposes. You can combine these approaches as needed to meet your business requirements.

Table 10.1 Row-Level Filtering Methods

Filter Assignment Method	Filter Is Identity-Driven	Usage Descriptions
Authorization-based prefilter	Yes	To make per-person (or per-identity) access distinctions for every member of a particular group, you can create a filter that uses an identity-driven property and assign that filter to a user group. The identity of each user in the group determines which rows the user can access. Users who are not members of the group are not subject to the filter. Because this is an authorization-based filter assignment, group memberships and identity precedence can affect the outcome. It makes sense to use this method when you want only some users to be subject to the filter, or you need to apply different filtering logic to different sets of users.
	No	To explicitly define different subsets for different identities, you can create a different filter for each subset and assign the filters to the appropriate users or groups.[1] Because these are authorization-based filter assignments, group memberships and identity precedence can affect the outcome. This method can be useful for very simple subsetting or in combination with other methods.

Filter Assignment Method	Filter Is Identity-Driven	Usage Descriptions
General prefilter	Yes	To make per-person (or per-identity) access distinctions for all users, you can create a filter that uses an identity-driven property and assign that filter as a general prefilter. All users will be subject to the filter, regardless of group memberships or access controls that grant broader access. It makes sense to use this method when the same filtering logic is appropriate for all users.
	No	To explicitly define one subset of data for all users, you can create a regular filter and assign that filter as a general prefilter. All users will be subject to the filter, regardless of group memberships or access controls that grant broader access. This method is not useful for row-level security purposes, because it does not yield different results for different requesting users. This method is useful for creating one data subset for all users.

[1] This method can require that you manage a large number of filters. You can often replace multiple regular filters with one filter that uses an identity-driven property.

Filters that Use Identity-Driven Properties

An identity-driven property is a user or group characteristic that is stored in the metadata and can be used in a filter as the value against which target data is compared. “How to Create a Filter for Row-Level Permissions” on page 157 provides instructions for creating a filter that uses an identity-driven property. When an information map that includes this type of filter is executed, an identity-specific value is substituted into the filter expression to yield a filter that is appropriate for each requesting user.

The metadata server uses the user ID with which a client is authenticated as the basis for determining other characteristics about that client. For each connecting client, the metadata server can derive identity-specific values for the following properties:

SAS.ExternalIdentity

an optional, site-specific value for the connecting client (such as employee ID). This property is often useful for filtering, because its values are likely to match user information that is already in the site’s data. If more than one external identity value is associated with the connecting client, then the first of those values is returned. If there are no associated external identity values, then a NULL (MISSING) value is returned and an error message is displayed.

As with the other identity-driven properties, the values for the ExternalIdentity property must be in the metadata so that SAS Intelligent Query Services can dynamically determine the appropriate value for each connection. However, unlike the values for other identity-driven properties, the ExternalIdentity values are not automatically populated in the metadata. If you want to use this property, you must load and maintain values for this property in the metadata repository by using the macros that are described in Appendix 2, “Bulk-Load Processes for Identity Management,” on page 185. During the identity bulk load process, ExternalIdentity values are extracted from an external enterprise identity source (such as Microsoft Active Directory Server or UNIX /etc/passwd files) and then imported into the SAS Metadata Repository. In this process, the association between each identity and the identity’s value for ExternalIdentity is preserved.

CAUTION:

If no extra precautions are taken, it is technically possible for users to change their ExternalIdentity values. To make the SAS.ExternalIdentity property reliable for identity-based filtering in a security-sensitive intranet environment, apply access controls to protect the ExternalIdentity objects. To obtain a macro that creates the appropriate protections, contact SAS technical support. As you add new users with ExternalIdentities to your deployment, you can either manually run the macro periodically or add the macro to automated identity bulk-load jobs. △

SAS.IdentityGroupName

the name of the requesting group identity, as displayed in the User Manager in SAS Management Console. If a user logs on with an ID that is part of a group login, 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 owns the connection.

For the following reasons, this property should rarely be used:

□ Unless a user logs on with the user ID that is defined for a *group* login, a NULL (MISSING) value is returned and an error message is displayed. 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 when pooled workspace servers are used.

CAUTION:

If standard precautions are not taken, users can change the names of group definitions. Group definitions should be protected as described in “Protect Group Definitions” on page 76. △

SAS.IdentityName

the name of the requesting user or group as displayed in the User Manager in SAS Management Console.

Note: To protect against misuse of this property, see the caution notes for the SAS.IdentityGroupName and SAS.PersonName properties. △

SAS.PersonName

the name of the requesting user identity, as displayed in the User Manager in SAS Management Console.

Note: In the rare situation where a user logs on with a user ID that is defined for a group login, a NULL (MISSING) value is returned and an error message is displayed. △

CAUTION:

By default, users can use SAS Management Console to change their names to any name that is not already in use. To protect against misuse of the SAS.PersonName property, ensure that the matching data mart column contains only values that are defined in the metadata. △

SAS.Userid

the authenticated user ID of the connecting client, normalized to the uppercase format USERID or USERID@DOMAIN.

Note: Because the SAS.Userid property is based on the logon ID that is used in the authentication process, users cannot change their values for this property. For this reason, filters that are based on the SAS.Userid property do not require the protection of any additional metadata access controls. △

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 runtime, SAS Intelligent Query Services asks the metadata server for the SAS.PersonName value that is associated with the connected user ID. SAS Intelligent Query Services then substitutes that identity-specific value into the filter. In this way, the query is modified as appropriate for each requesting client.

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

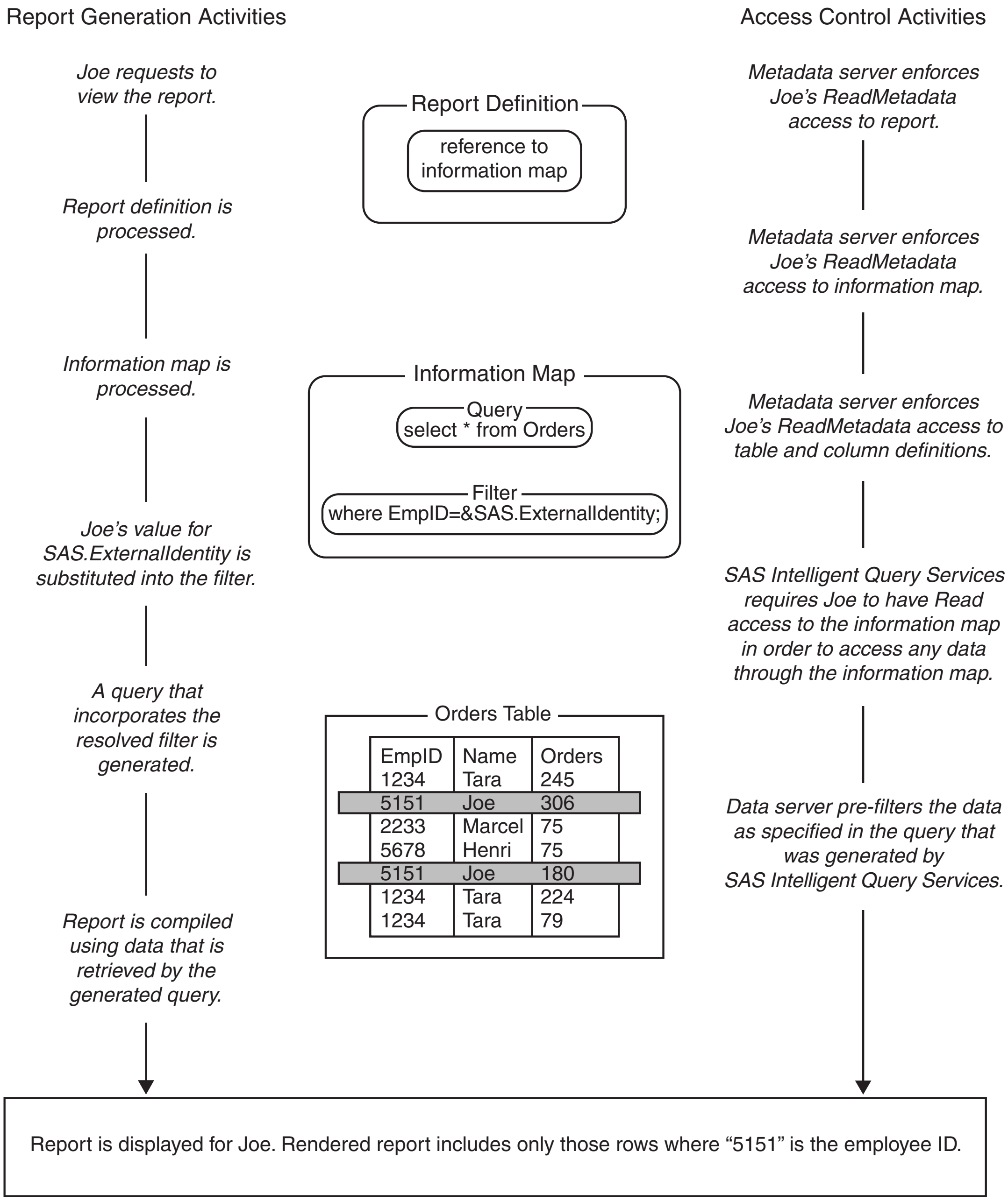
Table 10.2 Examples of Filters That Use Identity-Driven Properties

As Defined (Generic Form)	As Executed (Resolved Form)
Where EmpInfo.Name=&SAS.PersonName;	Where EmpInfo.Name="Harry Highpoint"
Where EmpInfo.Category=&SAS.IdentityGroupName;	An error message is returned because the user does not log on with a user ID that is stored as part of a group definition.
Where EmpInfo.Name=&SAS.IdentityName;	Where EmpInfo.Name="Harry Highpoint"
Where EmpInfo.WinID=&SAS.Userid;	Where EmpInfo.WinID="HIGH@WINNT"
Where EmpInfo.EmpID=&SAS.ExternalIdentity;	Where EmpInfo.EmpID="123–456–789"

How Row-Level Permissions Are Incorporated When a Report Is Generated

Row-level permissions are evaluated in coordination with controls for related resources (such as tables and columns) and controls in other authorization layers (such as physical access). Row-level permissions that are assigned to specific identities constrain only direct grants of the Read permission on information maps. The following figure depicts an example of how row-level permissions work. In the figure, a user requests access to a report that includes data for which row-level permissions have been defined by using an identity-driven property. For each step of the report-generation process, the figure depicts the access control activities in the metadata layer.

Figure 10.1 Report-Generation Process


Report Generation Activities
Access Control Activities
Joe requests to view the report.
Report definition is processed.
Information map is processed.
Joe's value for SAS.ExternalIdentity is substituted into the filter.
A query that incorporates the resolved filter is generated.
Report is compiled using data that is retrieved by the generated query.
Report Definition
reference to information map
Information Map
Query
select * from Orders
Filter
where EmpID=&SAS.ExternalIdentity;
Orders Table
Metadata server enforces Joe's ReadMetadata access to report.
Metadata server enforces Joe's ReadMetadata access to information map.
Metadata server enforces Joe's ReadMetadata access to table and column definitions.
SAS Intelligent Query Services requires Joe to have Read access to the information map in order to access any data through the information map.
Data server pre-filters the data as specified in the query that was generated by SAS Intelligent Query Services.
Report is displayed for Joe. Rendered report includes only those rows where "5151" is the employee ID.

EmpID	Name	Orders
1234	Tara	245
5151	Joe	306
2233	Marcel	75
5678	Henri	75
5151	Joe	180
1234	Tara	224
1234	Tara	79

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 runtime, 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.

Precedence for Row-Level Permission Conditions

BI row-level filters that are assigned to specific metadata identities are evaluated by the authorization facility as permission conditions. For complete information about how authorization decisions are made, see "A Closer Look: How Authorization Decisions are Made" on page 46. The access control principles that are most relevant to row-level permission conditions are summarized in the following table:

Table 10.3 Access Control Principles for Row-Level Permission Conditions

Principle	**Example**	
	Scenario	**Outcome and Explanation**
A direct access control on an information map has precedence over access controls that come from the folder that contains the information map.	A direct access control on InformationMapA denies Read permission to PUBLIC. A direct access control on the folder that contains InformationMapA grants Read permission to a particular user.	The user cannot access data through InformationMapA. The denial to PUBLIC has precedence over the grant to the user because the denial is assigned *directly* on the target resource (InformationMapA). Direct access controls always have precedence over inherited controls *regardless of who the permissions are assigned to*.
In order to assign a row-level permission filter to an identity, the identity (or a group to which the identity belongs) must have a direct grant of Read permission on the information map.	The only access control on InformationMapA is an inherited grant of Read permission to PUBLIC.	You cannot define row-level permissions for InformationMapA. The identity (or a group to which the identity belongs) must be added to the **`Authorization`** tab for Information MapA and directly granted Read permission. In the **`Authorization`** tab, a direct grant has a white background.

Principle	Example	
	Scenario	**Outcome and Explanation**
If there are multiple row-level filters 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. See "Example: Precedence for Row-Level Permission Conditions" on page 146.
If there are multiple row-level controls at the same identity level, then the outcome is the superset of rows that are allowed by either filter.	A filter on InformationMapA limits Read permission for GroupA. Another filter on InformationMapA limits Read permission for the GroupB. The user is a first level member of both GroupA and GroupB.	The user can see any row that is permitted for either GroupA or GroupB.

Example: Precedence for Row-Level Permission Conditions

This 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 row-level filter assigned to the SASUSERS group gives each user access to his or her own salary information.
- □ A row-level filter 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 through this information map, 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 cannot 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.

How to Create a Secure Environment for BI Row-Level Permissions

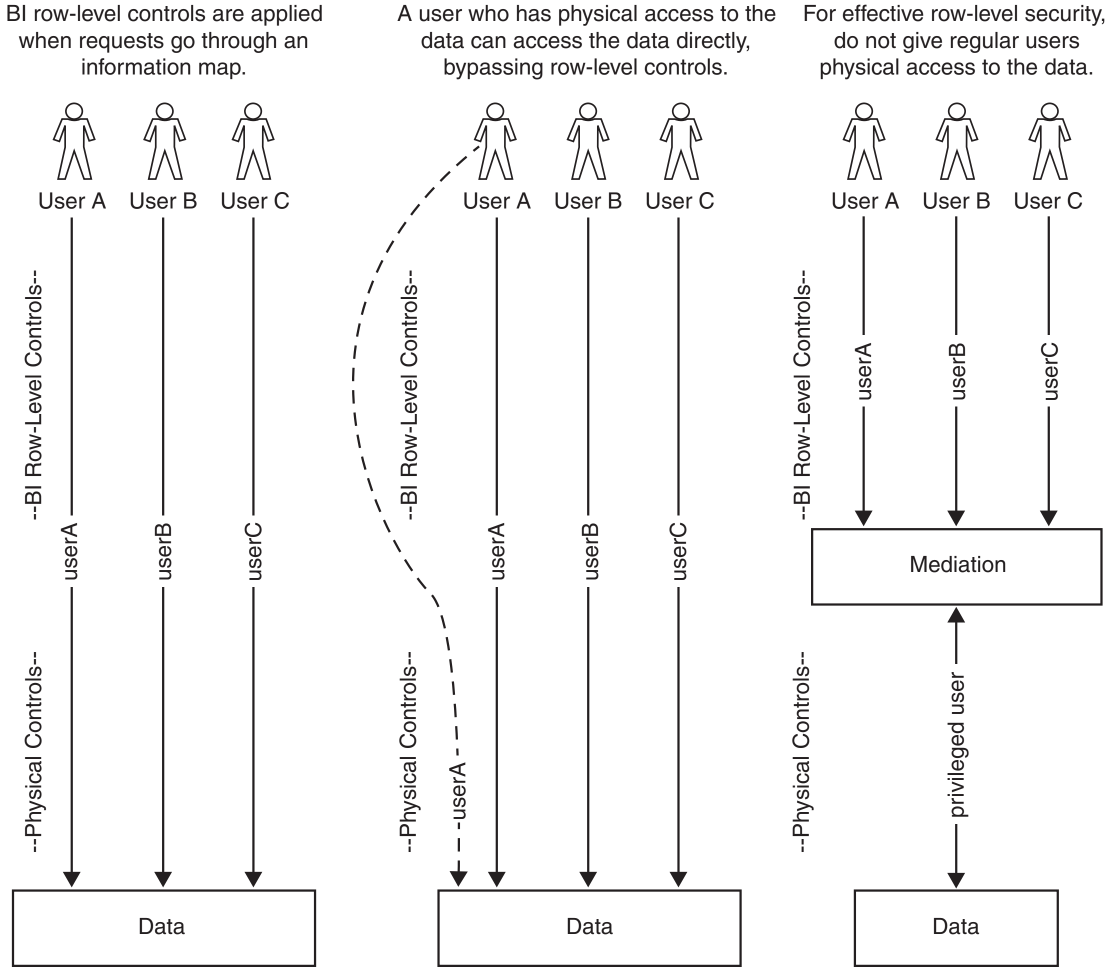
Overview of Requirements

Like any other security feature, row-level security requires that you pay careful attention to the entire environment in order to avoid vulnerabilities in other security layers. For example, if you do not limit physical access to the target data, there is a risk that users will exploit their physical access to circumvent the row-level filters that you create. If this is an acceptable risk, then no special measures are needed. For example, this can be an acceptable risk in these types of environments:

- prototype environments
- environments in which a firewall segregates untrusted users
- environments in which untrusted users do not have the tools, knowledge, or OS privileges to access files and metadata on the server tier
- environments that for other reasons do not have strict security requirements

If, on the other hand, you require strict security controls against the possibility of malicious activity on your company intranet, then a more tightly protected configuration is necessary. In such circumstances, it is important to strictly limit physical access to the target tables to prevent direct access by regular users. The goal is to enable regular users to have only mediated access to the target tables. The strategy is to ensure that participating applications use a privileged account to fetch data for requesting users, and to deny regular users physical access to the tables. The following diagram illustrates these points.

Figure 10.2 Secure Environment


BI row-level controls are applied when requests go through an information map.
A user who has physical access to the data can access the data directly, bypassing row-level controls.
For effective row-level security, do not give regular users physical access to the data.
User A
User B
User C
--BI Row-Level Controls--
--Physical Controls--
userA
userB
userC
Mediation
privileged user
Data

The mediated configuration is supported only for SAS Web Report Studio. Other applications can make use of identity-based filtering and row-level permissions but cannot provide comprehensive row-level security.

Instructions for Setting up the Recommended Environment

The mediation that is depicted in the preceding figure is provided by a pooled workspace server that is dedicated for use with SAS Web Report Studio for these reasons:

□ Using a pooled workspace server prevents the workspace server processes from running under the accounts of requesting users. Pooled workspace servers run under one or more designated accounts that are called puddle accounts.

□ Using a dedicated workspace server isolates the puddle account from applications that do not fully enforce row-level security. The pool administrator account is identified in a configuration file that is used only by SAS Web Report Studio, so workspace servers that are launched from other applications cannot use the pool.

To ensure the tightest possible security, follow these instructions.

1 Verify that the basic protections that are described in Chapter 4, “Securing a Deployment,” on page 63 are in place.

2 Create a new pooled workspace server for exclusive use by SAS Web Report Studio. For instructions, see "Configure a Pooling Workspace Server to Enforce Row-Level Security" in the chapter "Reconfiguring or Clustering Workspace or Stored Process Servers" in the *SAS Intelligence Platform: Application Server Administration Guide*.

3 Ensure that the puddle account for the restricted workspace server can physically access the target data and that regular users cannot. Regular users are people who should be able to access the data only from SAS Web Report Studio.

- □ For SAS data sets, give read access to the puddle account in the operating system layer.
- □ For third-party database tables, give read access to the data to a privileged database account. (In a later step you will make the credentials for this privileged database account available to the puddle account).

Note: Some members of your staff will also need physical access to the data. For example, the person who creates an information map based on the target data must have physical access to the data. △

4 For target data that is in third-party databases, assign libraries by using the METAAUTOINIT method of library pre-assignment. This method causes the libraries to be assigned to the metadata identity for the puddle account.

Note: When you pre-assign a library using the METAAUTOINIT method, authorization decisions are based on the metadata identity under which the workspace server connects to the metadata server. Workspace servers that use the -METAPERSON option connect under the identity that is specified by that option. Other workspace servers connect under the identity of the puddle account (pooled workspace servers) or the identity of the connecting client (standard workspace servers). These statements assume that workspace servers that do not use the -METAPERSON option connect to a metadata server that uses the TRUSTSASPEER option, which is the default configuration. △

To use the METAAUTOINIT method to assign libraries, complete these steps:

a In SAS Management Console, navigate to the restricted workspace server and select **Properties ▶ Options**.

b In the **`Object Server Parameters`** field, enter **`METAAUTOINIT`**. This tells the workspace server to connect to the metadata server to obtain information about library assignments.

Note: The workspace server connects using the account that it is running under (the puddle account). The metadata server determines that the metadata identity for the puddle account is the Restricted Puddle Access group (because you stored the puddle account user ID in a login on this group definition). This causes the libraries to be assigned to the Restricted Puddle Access group. △

c Stop and then restart the object spawner to make this change take effect.

With this method, the restricted workspace server can set up and use a target DBMS library, while attempts to assign the library under another metadata identity will fail. For example, a regular workspace server that is launched by Tara O'Toole while using SAS Enterprise Guide cannot successfully assign the DBMS library, because the workspace server's metadata identity (Tara O'Toole) does not have physical access to the library.

5 For target data that is in third-party databases, set up credentials in the metadata to enable the puddle account to access those servers. You can make credentials for a database server available to the puddle account by storing those credentials in a

login as part of the Restricted Puddle Access group definition. For example, to enable the puddle account to access a DB2 server, you would give the Restricted Puddle Access group a login that includes a DB2 user ID and password and that is associated with the DB2 server's authentication domain.

Note: As explained in step 3, some members of your staff will also need to be able to authenticate to the database server. △

How to Implement Row-Level Permissions

Process Overview for Implementing Row-Level Permissions

The process for setting up row-level permissions consists of these phases:

1 Review and summarize what you want to accomplish by defining row-level permissions.

2 Determine how you can combine row-level permissions with other metadata layer and physical controls to meet your business requirements.

3 Structure the target data to fit with the subsetting that you want to do.

4 Create filters that implement the row-level access rules that you have identified and that work with your data.

5 Test the row-level controls to verify that they function as intended.

The following topics describe these phases, using simple, abbreviated examples to explain specific points. For an integrated example of the entire process, see "Example: Using Row-Level Permissions" on page 158.

Business Requirements Phase

Business requirements often dictate 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.

In many cases, there are coarser-grained (table-level) business requirements that accompany the row-level access rules. For example, business requirements often dictate that some users (such as executives or system administrators) should be able to access all rows in a target table, while some users (such as users who do not have individual metadata identities) should not be able to access any rows.

Planning Phase

Planning for row-level security consists of these steps.

1 If your site has strict security requirements, you must perform additional deployment configuration steps to create an appropriate environment. See “How to Create a Secure Environment for BI Row-Level Permissions” on page 147.

2 Set coarse-grained controls as described in the following table.

Table 10.4 Coarse-Grained Controls

Business User Access Class	Metadata Layer		Physical Layer
	Target Table	Information Map	Target Table
All rows	Grant R, RM	Grant R, RM	Deny[1]
No rows	Deny R, RM	Grant[2] R, RM	Deny
Some rows	Grant R, RM	Grant R[3], RM	Deny[1]

[1] In a high-security environment, regular users should not have physical access to the data. In other circumstances, regular users might have physical access to the data.

[2] Grant these permissions if the "No rows" users need to access other tables through this information map.

[3] For filters that are assigned as authorization-based prefilters, this must be a direct grant of Read permission on the information map. This access will be constrained by the row-level conditions that you define in the next step.

3 Decide how you will create the data subsets that will narrow the direct grant of Read permission as appropriate for each user. Your options are explained in “Filtering Methods for Row-Level Permissions” on page 140. For example:

- □ If you want to make per-person access distinctions for the members of a particular user group, select an identity-driven property to use as the basis for an authorization-based prefilter that you will assign to that user group. For example, to give each user access to a distinct set of rows based on the user’s metadata identity, you might plan to create a filter that uses the SAS.Userid property and to assign that filter to the PUBLIC group.
- □ If you want to make a relatively small number of access distinctions, each of which will apply to one or more metadata user or group identities, design a separate filter for each class of access and plan to assign each filter to the appropriate identities. For example, to create low-, medium-, and high-security subsets of data, you might design three filters and plan to assign each of those filters to a different metadata user group.
- □ For more complex business requirements, you can use combinations of the different filtering techniques.

Your choice of filtering methods 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. When you are composing the filtering logic that you will use to meet your business requirements, consider these guidelines:

- □ For manageability, limit the total number of filters that you define. Many common business requirements can be met by using a filter that is based on an identity-driven property. There is significantly less maintenance involved in this approach than in explicitly defining a different filter for each user.
- □ For manageability, try to assign filters that you create to user groups rather than to individual users. In order to assign a filter to a user group, that user group must be defined in the metadata. Familiarity with the user group structure in your metadata will help you efficiently define row-level controls.

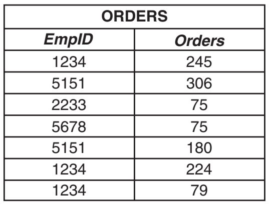
- □ For simplicity, avoid situations in which multiple filters apply to a particular user as a result of the user's group memberships. In such situations, the subset of data that is available to that user is determined by identity precedence (see "Precedence for Row-Level Permission Conditions" on page 145).

Data Modeling Phase

Overview of the Data Modeling Phase

Row-level permissions require that the target data support the subsetting that you will use to meet your business requirements. In many cases, you must modify an existing data model to include information that corresponds to the filters that you will use. As a simple example, consider a company that consists of a four-person, flat organizational structure and has a business requirement that each employee should see only his or her own order information. The order information is stored in a table that looks like this:

Figure 10.3 Orders Example: Target Table



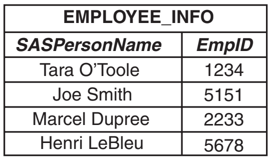
ORDERS	
EmpID	*Orders*
1234	245
5151	306
2233	75
5678	75
5151	180
1234	224
1234	79

You must supplement this existing simple data model to support and fit the filtering that you want to do. For this example, assume that the choice of filtering method is affected by these points:

- □ You do not want to manage a different filter for each user.
- □ You manually created your metadata identities, so you do not have SAS.ExternalIdentity values in the metadata that correspond the EmpID values in the ORDERS table.

In these circumstances, you will need to enhance the data to support filtering based on another identity-driven property such as SAS.PersonName. To support this subsetting, you would create an employee information table that includes a PersonName column (or add a PersonName column to an existing employee information table). In each row, you would enter a value that corresponds to the employee's name on the **`General`** tab of his or her user definition in SAS Management Console (because this is the SAS.PersonName value for the employee). A minimal version of the table that is needed looks like this:

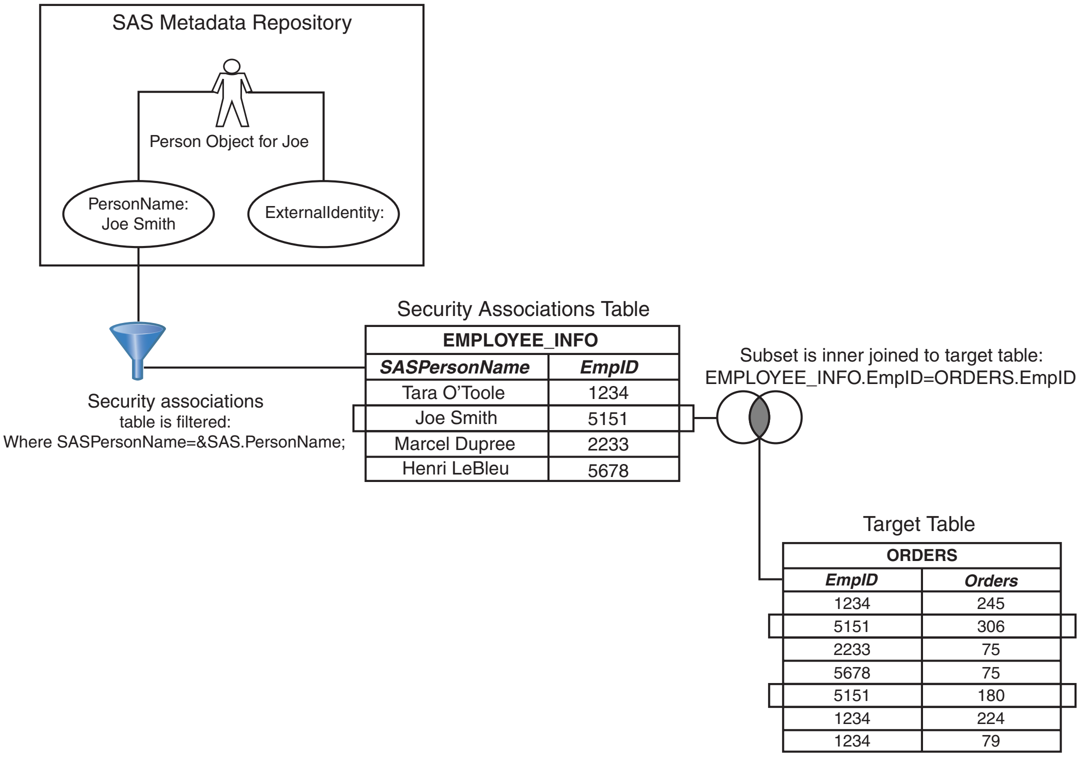
Figure 10.4 Orders Example: Security Associations Table



EMPLOYEE_INFO	
SASPersonName	*EmpID*
Tara O'Toole	1234
Joe Smith	5151
Marcel Dupree	2233
Henri LeBleu	5678

When an end user submits a query, the information map that provides access to the ORDERS table uses the employee information table to pre-screen the data. The employee information table is filtered based on each requesting user's identity and then inner joined to the ORDERS table (on the EmpID column). The following figure depicts this process:

Figure 10.5 Orders Example: How the Security Associations Table Is Used


SAS Metadata Repository
Person Object for Joe
PersonName: Joe Smith
ExternalIdentity:
Security associations
table is filtered:
Where SASPersonName=&SAS.PersonName;
Security Associations Table
Subset is inner joined to target table:
EMPLOYEE_INFO.EmpID=ORDERS.EmpID
Target Table

EMPLOYEE_INFO

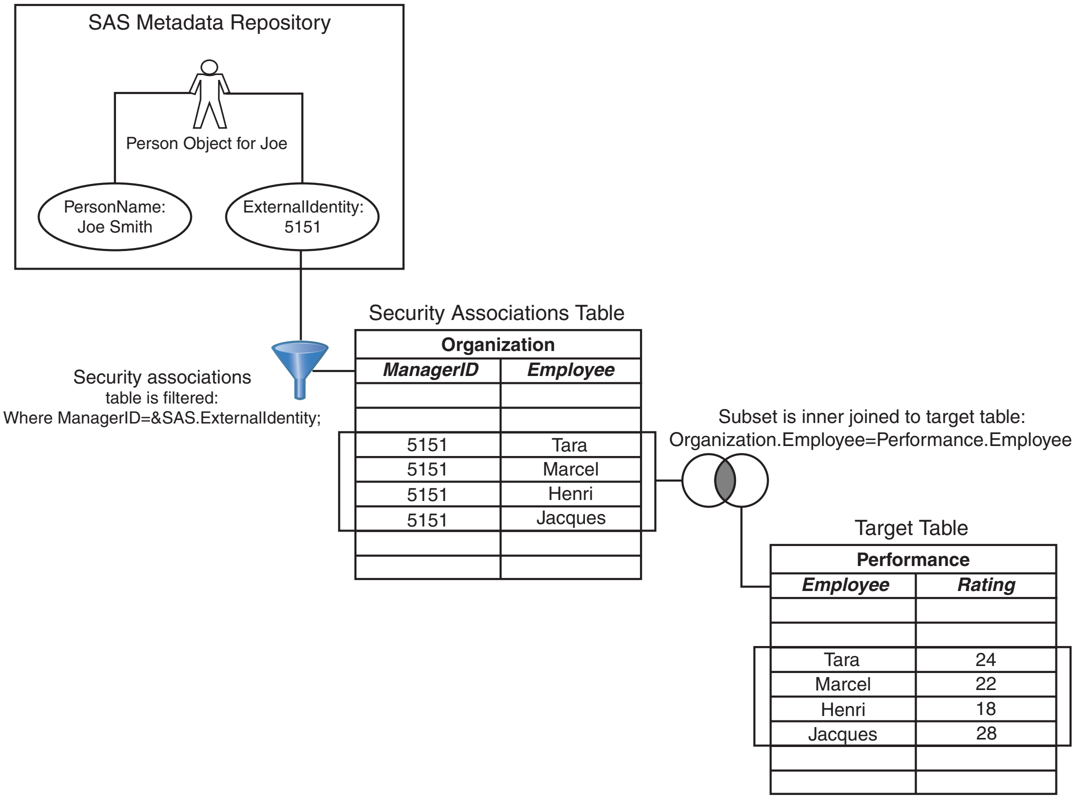
SASPersonName	***EmpID***
Tara O'Toole	1234
Joe Smith	5151
Marcel Dupree	2233
Henri LeBleu	5678

ORDERS

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

As another simple example, consider a company that has a business requirement that each manager can see performance rating information for his or her direct reports. As in the previous example, you must supplement the existing data to support and fit the filtering that you want to do. For this example, assume that the SAS.ExternalIdentity information is available in the metadata and that you choose to base your filtering on this identity-driven property. The following figure depicts a data model that supports subsetting based on each manager's value for the ExternalIdentity property.

Figure 10.6 Performance Example: How the Security Associations Table Is Used


SAS Metadata Repository
Person Object for Joe
PersonName: Joe Smith
ExternalIdentity: 5151
Security associations table is filtered: Where ManagerID=&SAS.ExternalIdentity;
Security Associations Table
Subset is inner joined to target table: Organization.Employee=Performance.Employee
Target Table

Organization	
ManagerID	*Employee*
5151	Tara
5151	Marcel
5151	Henri
5151	Jacques

Performance	
Employee	*Rating*
Tara	24
Marcel	22
Henri	18
Jacques	28

The purpose of these simplified examples is to introduce the idea of managing security associations in a separate table and to illustrate how that table is used. In most cases, the volume of data is larger and the business requirements are more complex. For example, the security associations table in the performance rating example does not enable a manager to see his or her own rating. These differences can result in additional considerations for the security associations table. The following topics address some of those considerations.

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.

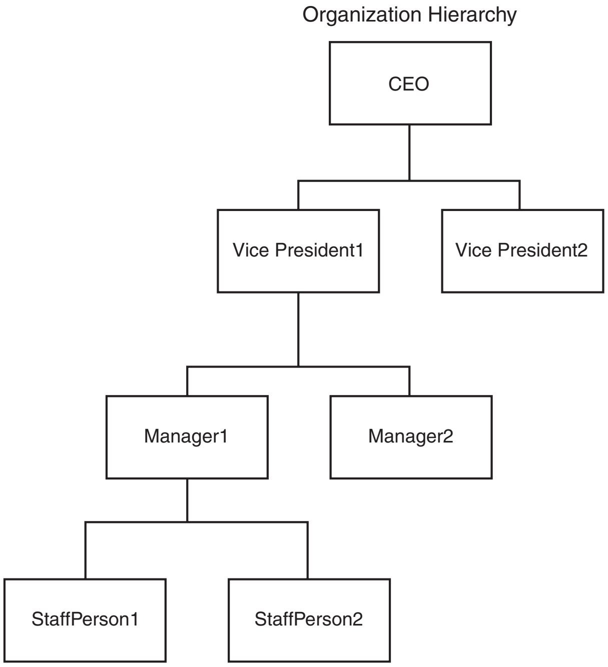
Note: In the preceding examples, the security associations tables are the EMPLOYEE_INFO table (in the orders example) and Organization table (in the performance rating example). △

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 10.7 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 greatgrandparent-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 adhere to a particular structure. This description 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 will be 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.

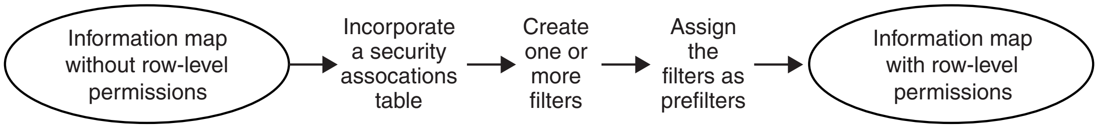
The security associations table must be maintained as security relationships change. This maintenance should be on a schedule that is appropriate for your environment. Typically, this maintenance is accomplished by a batch process (such as a nightly ETL process against the existing tables). In some cases,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

Information Map Design Phase

Overview of the Information Map Design Phase

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

Figure 10.8 Information Map Design for Row-Level Permissions


Information map without row-level permissions
Incorporate a security assocations table
Create one or more filters
Assign the filters as prefilters
Information map with row-level permissions

The following topics provide generic instructions for each of these four tasks. For a specific example with screenshots, see "Example: Using Row-Level Permissions" on page 158.

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. For example, to enhance an existing information map to include a new security associations table, you would perform these steps:

1 Register the new security associations table in the metadata.

2 In SAS Information Map Studio, open an information map and then select **Insert ▸ Table**.

3 In the Insert Table dialog box, select the table that you are using as a security associations table, and then click `OK`.

4 On the `Relationships` tab in the application's main window, create the connections between the table that you are using as a security associations table and other tables in the model. This procedure typically involves defining an inner join to connect an identifier column in the security associations table with a corresponding column in the target table (or in an intermediate dimension).

5 Make the security associations table a required table by performing these steps:

- a Select **Edit ▶ Properties ▶ Information Map**, and then select the **`Required Tables`** tab in the Information Map Properties dialog box.
- b In the **`Available tables`** list, select the table that you are using as a security associations table.
- c Use the arrow button to move the table to the **`Required tables`** list.
- d Click **`OK`**.

Note: We recommend that you do not add data items from a security associations table to an information map. Excluding these items from the information map prevents these items from surfacing when reports are created in SAS Web Report Studio. △

How to Create a Filter for Row-Level Permissions

Filters that are based on identity-driven properties can be very useful for row-level security purposes. 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 and then select **Insert ▶ Filter** to open the New Filter dialog box.
2. Enter a name and description for the filter, and then click **`Edit Data Item`**.

   Note: In these instructions, the filter uses a physical column rather than one of the business data items that are listed in the **`Data item`** drop-down list. For row-level security, we recommend that filters use physical columns, because this prevents the filters from surfacing when reports are created in SAS Web Report Studio. △
3. In the Edit Expression dialog box, select a physical column (from the table that you are using as a security associations table), and then click **`OK`**.
4. In the New Filter dialog box, the fields in the **`Values`** section are now available. From the **`Enter value(s)`** drop-down list, select **`Derive Identity Values`**. A table of identity-driven properties becomes available.
5. In the table of properties, select the row for the identity-driven property that you want to use in the filter.
6. Click **`OK`**. The new filter is now available for use in the current information map.

You can use a wide variety of filters for row-level security purposes. To learn about other filtering choices, see the Help for SAS Information Map Studio.

How to Assign a Filter for Row-Level Permissions

In order to be used for security purposes, a filter must be assigned as either an authorization-based prefilter or a general prefilter.

To assign a filter as an authorization-based prefilter, perform these steps in SAS Information Map Studio:

1. Open the information map and then select **Tools ▶ Authorization** to open the Authorization dialog box.
2. In the **`Names`** box, select an identity to which you will assign the filter, or click **`Add`** to add an identity to the **`Names`** box.
3. If Read permission is not directly granted to the selected identity, add a direct grant by selecting the **`Grant`** check box. In the **`Permissions`** list, a white background color behind a selected check box indicates that the permission is directly assigned.
4. Click **`Add Condition`** (or **`Edit Condition`**) to open the Row-Level Permission Condition dialog box.

5 In the **Selected filters** list, 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.

7 Click **OK** to apply the filter assignment and close the Row-Level Permission Condition dialog box.

8 In the Authorization dialog box, click **Close**.

9 To make your changes take effect, save the information map.

To assign a filter as a general prefilter, perform these steps in SAS Information Map Studio:

1 Open the information map and then select **Edit ▶ Properties ▶ Information Map**.

2 In the Information Map Properties dialog box, select the **General Prefilters** tab.

3 In the **Selected filters** list, select the table that you are using as a 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.

5 In the Information Map Properties dialog box, click **OK** to apply the filter assignment.

Verification Phase

Testing should be performed from an application such as SAS Web Report Studio. This testing requires that you log on to that application using different accounts.

Note: For users who 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. △

Example: Using Row-Level Permissions

Introduction, Assumptions, and Data Model

The following example demonstrates how a company could use row-level permissions to manage access to employee data. The example is based on the following assumptions:

- The company is running SAS Information Map Studio 3.1, SAS Web Report Studio 3.1, and the SAS Intelligence Platform 9.1.3 with Service Pack 4.
- 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.
- Except where otherwise noted, the company has constrained physical access to target data as described in “How to Create a Secure Environment for BI

Row-Level Permissions" on page 147, and has set appropriate protections for the target library and tables.

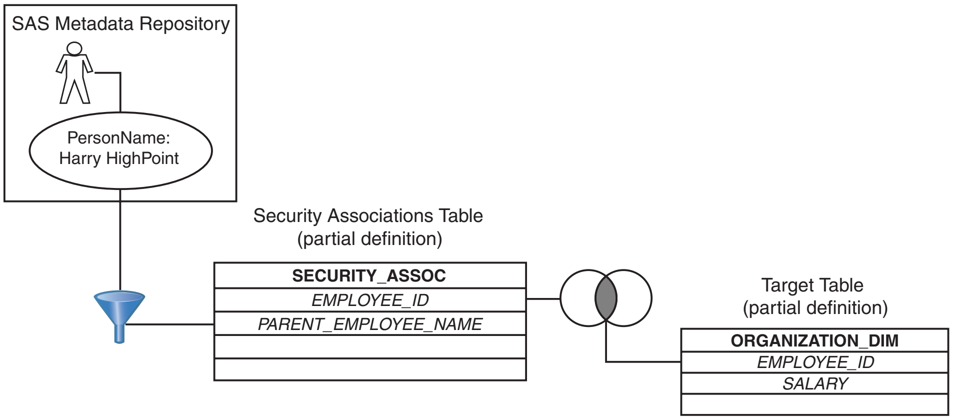
The data model for the example is a star schema that contains employee and customer data for a fictional sporting goods company. To support efficient row-level filtering, the security associations table includes both direct and indirect reporting relationships.

Note: This example uses a classic star schema design because this is a common data structure for query and reporting purposes. BI row-level permissions do not require that you use a particular data structure. △

Implementation Process

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 SAS.PersonName of each requesting user is used to filter the security associations table, based on corresponding values in the PARENT_EMPLOYEE_NAME column. This yields a subset of rows that includes all employees who report (directly or indirectly) to the requesting user. That subset of rows is then inner joined to the table that contains salary information, so that only the salaries of employees who report to the requesting user are returned. The following figure depicts this process for a requesting user who is a high-level manager in the organization. The SAS.PersonName value for this requesting user is "Harry Highpoint".

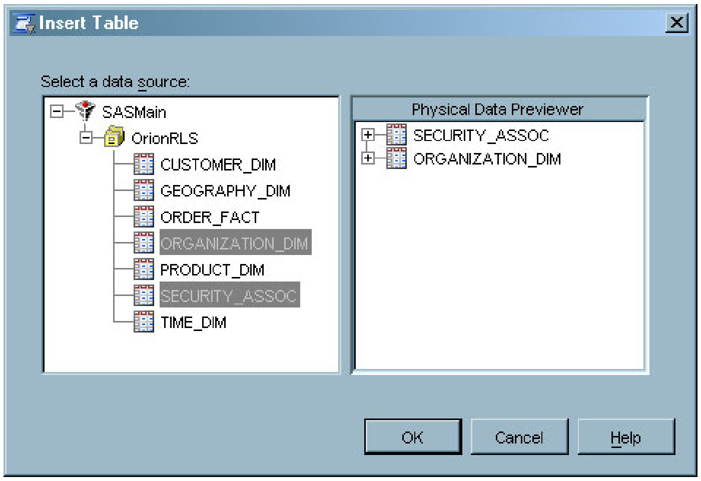
Figure 10.9 Salary Filtering with the SAS.PersonName Property



To set up these row-level permissions, complete these information map design tasks:

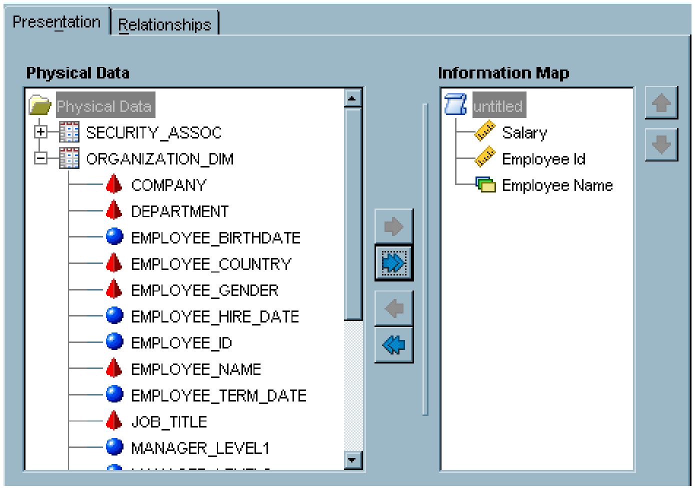
1 Create an information map that includes the salary information, security associations information, and necessary relationships.

   a In SAS Information Map Studio, select **File ▶ New** to open a new information map.
   b Select **Insert ▶ Table** to open the Insert Table dialog box, and then navigate to the library that contains the target data. In this example, the library is named **`OrionRLS`**. Select the table that contains a representation of reporting relationships (SECURITY_ASSOC) and the table that contains salary information (ORGANIZATION_DIM) and then click **`OK`**.

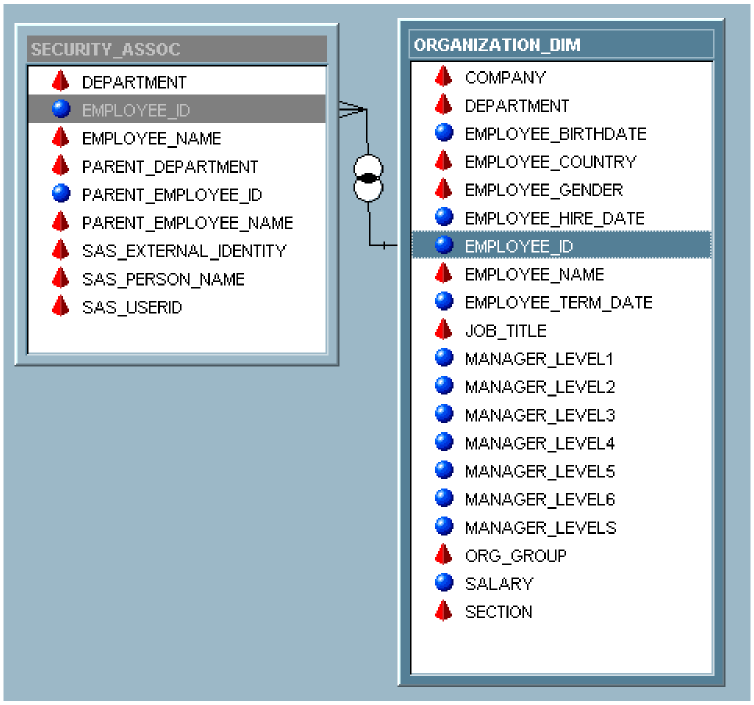


c In the main application window, on the **`Presentation`** tab, add the data items that you will need from each table:

- □ It is a good practice to not add any data items from the SECURITY_ASSOC table. You will use the PARENT_EMPLOYEE_NAME column when you create a filter, but you will use the physical item for this purpose.
- □ From the ORGANIZATION_DIM table, insert the SALARY, EMPLOYEE_ID, and EMPLOYEE_NAME columns.



d On the **`Relationships`** tab, join the two tables on EMPLOYEE_ID.



e Select **File ▶ Save**, navigate to an appropriate folder, give the new information map a name such as *SalaryByPersonName(withGeneralPrefilter)*, and click **Save**.

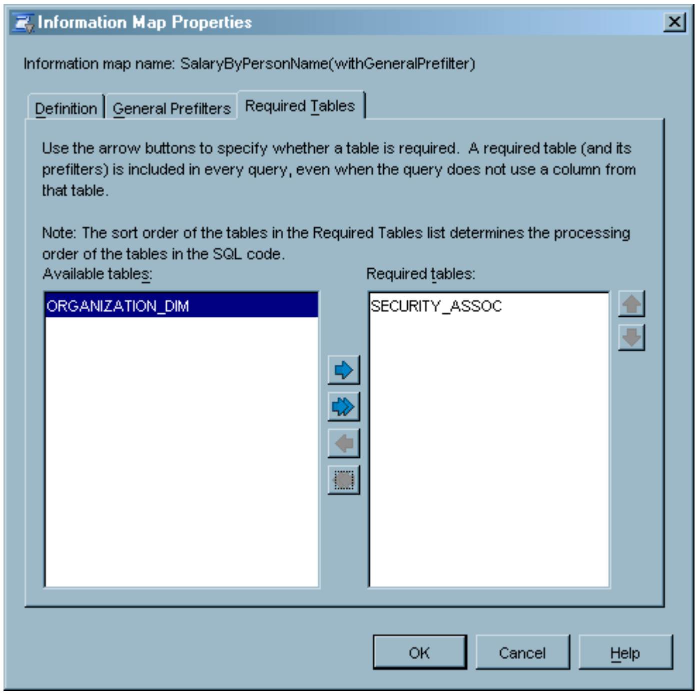
f To make the SECURITY_ASSOC table a required table, perform these steps:

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**.



Information map name: SalaryByPersonName(withGeneralPrefilter)

Use the arrow buttons to specify whether a table is required. A required table (and its prefilters) is included in every query, even when the query does not use a column from that table.

Note: The sort order of the tables in the Required Tables list determines the processing order of the tables in the SQL code.

Available tables: ORGANIZATION_DIM

Required tables: SECURITY_ASSOC

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

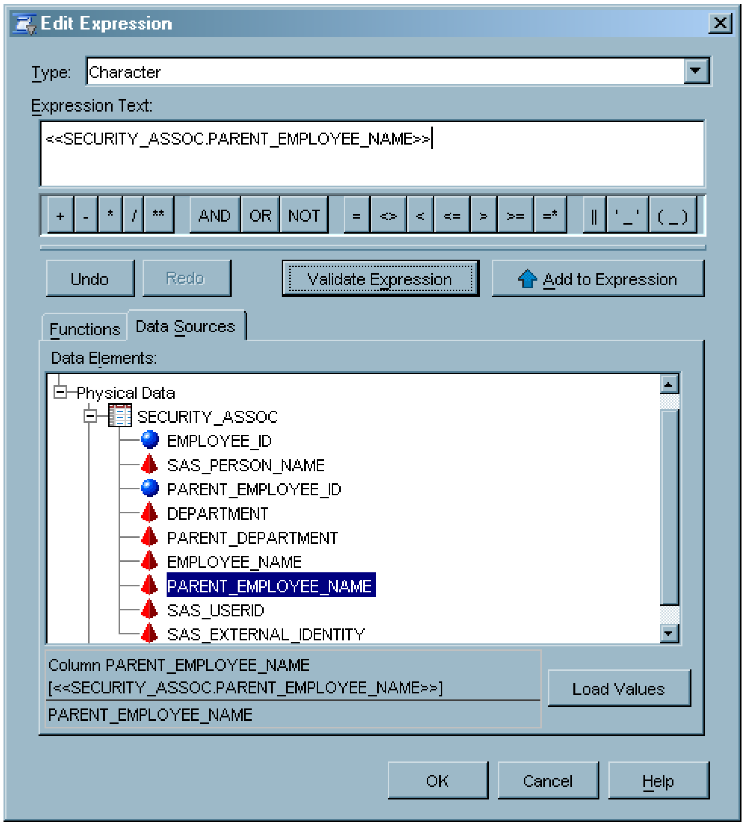
a Select **Insert** ▶ **Filter** to open the New Filter dialog box.

If this menu selection is not available, you do not have Read access for the new information map. To grant the Read permission for this information map, select **Tools** ▶ **Authorization**. As an alternative, you can set a default grant of this permission for the entire repository, as explained in "Example: Giving Everyone Default Read Permission to Everything" on page 118.

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`**.

Note: A physical column is used because this prevents the filter from surfacing when reports are created in SAS Web Report Studio. △



Type: Character

Expression Text: <<SECURITY_ASSOC.PARENT_EMPLOYEE_NAME>>

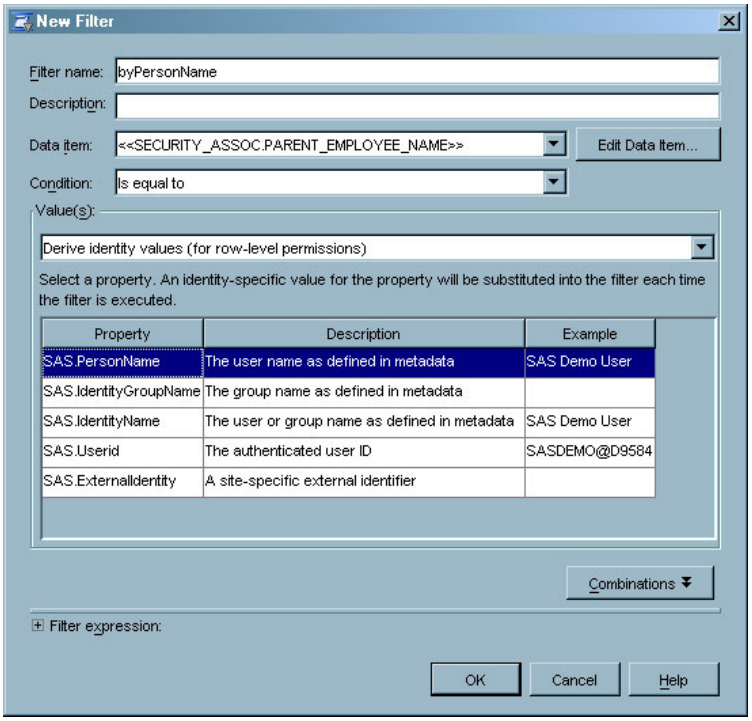
Column PARENT_EMPLOYEE_NAME
[<<SECURITY_ASSOC.PARENT_EMPLOYEE_NAME>>]

PARENT_EMPLOYEE_NAME

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**. A table of identity-driven properties becomes available.

f In the table of properties, select the SAS.PersonName row.



Filter name: byPersonName

Description:

Data item: <<SECURITY_ASSOC.PARENT_EMPLOYEE_NAME>>

Condition: Is equal to

Value(s):

Derive identity values (for row-level permissions)

Select a property. An identity-specific value for the property will be substituted into the filter each time the filter is executed.

Property	Description	Example
SAS.PersonName	The user name as defined in metadata	SAS Demo User
SAS.IdentityGroupName	The group name as defined in metadata	
SAS.IdentityName	The user or group name as defined in metadata	SAS Demo User
SAS.Userid	The authenticated user ID	SASDEMO@D9584
SAS.ExternalIdentity	A site-specific external identifier	

g Click **OK**. The byPersonName filter is now available for use in the information map.

3 To assign the filter as a general prefilter, complete these steps:

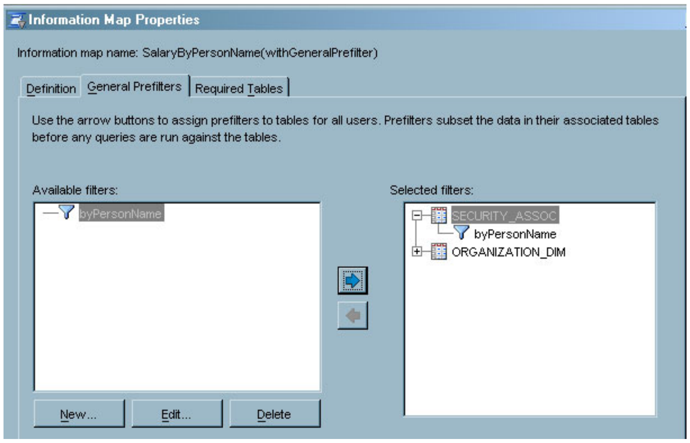
a Select **Edit ▶ Properties ▶ Information Map**.

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**.



Information map name: SalaryByPersonName(withGeneralPrefilter)

Use the arrow buttons to assign prefilters to tables for all users. Prefilters subset the data in their associated tables before any queries are run against the tables.

Available filters:
byPersonName

Selected filters:
SECURITY_ASSOC
byPersonName
ORGANIZATION_DIM

4 Select **File ▶ Save** to save the information map.

Users 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 (such as the SAS Demo User), 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, complete these steps:

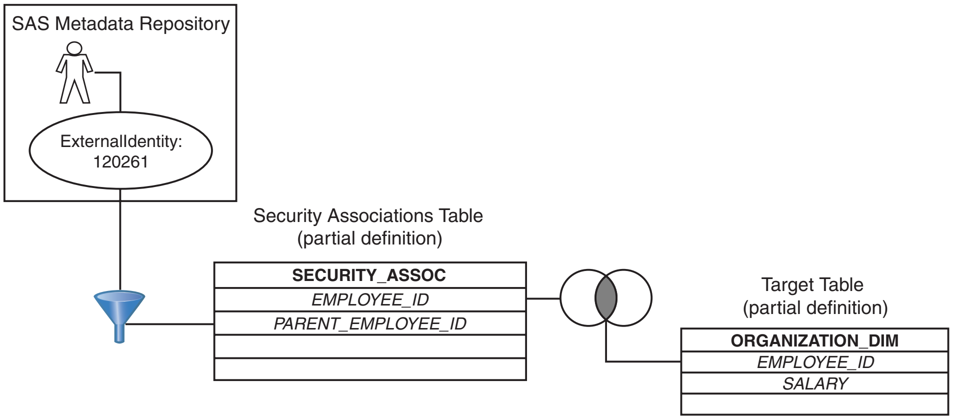
1. Select **Tools ▶ Test** from the main menu.
2. In the Test the Information Map dialog box, use the arrow button to add the **`Salary`** and **`Employee Name`** items to the **`Selected Items`** box.
3. Click **`Run Test`** and then examine the data in the Results window.
4. To test using another account,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

Note: Final verification, and verification for users who do not have physical access to the data, must be performed from within SAS Web Report Studio. △

Variation 1: Use a Different Property for Filtering

If the target data only identified parent employees by their company ID (rather than also by their employee name), then you would need to use a different identity-driven property to accomplish this filtering. The SAS.ExternalIdentity of each requesting user is used to filter the security associations table, based on corresponding values in the PARENT_EMPLOYEE_ID column. This filtering yields a subset of rows that includes all employees who report (directly or indirectly) to the requesting user. That subset of rows is then inner joined to the table that contains salary information, so that only the salaries of employees who report to the requesting user are returned. The following figure illustrates how this filtering could be accomplished:

Figure 10.10 Salary Filtering with the SAS.ExternalIdentity Property



Note: This variation assumes that bulk-load macros were used to create the metadata identities in the deployment. As part of the user import process, the company's employee IDs were added to the repository as SAS.ExternalIdentity values. △

The implementation process for this variation is very similar to the main example. The only differences are in step 2—the filter creation process. This variation differs from the preceding example in these ways:

- □ In step 2c you would select a different physical data item (PARENT_EMPLOYEE_ID rather than PARENT_EMPLOYEE_NAME).
- □ In step 2f you would select a different identity-driven property (SAS.ExternalIdentity rather than SAS.PersonName).

Note: The **`Derive Identity Values`** selection in the New Filter dialog box is available only when you are defining a filter for a character data item. △

Variation 2: Apply Different Filtering Logic to Different Groups

This variation addresses the following additional business requirements:

- □ Four people who work in a Human Resources management department must be able to view salary information for all employees. You have created a user-defined group in the metadata repository for the users (the group name is HR All Salaries).
- □ 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.

The first part of the implementation process for meeting these requirements is the same as steps 1 and 2 in the main example. To meet the business requirements in this variation, you must set some specific access controls at the level of the entire information map and then assign the filter as an authorization-based prefilter that will apply only to one particular group of users (rather than as a general prefilter, which has a universal effect).

The permissions that you will set are summarized in the following table.

Table 10.5 Information Map Controls

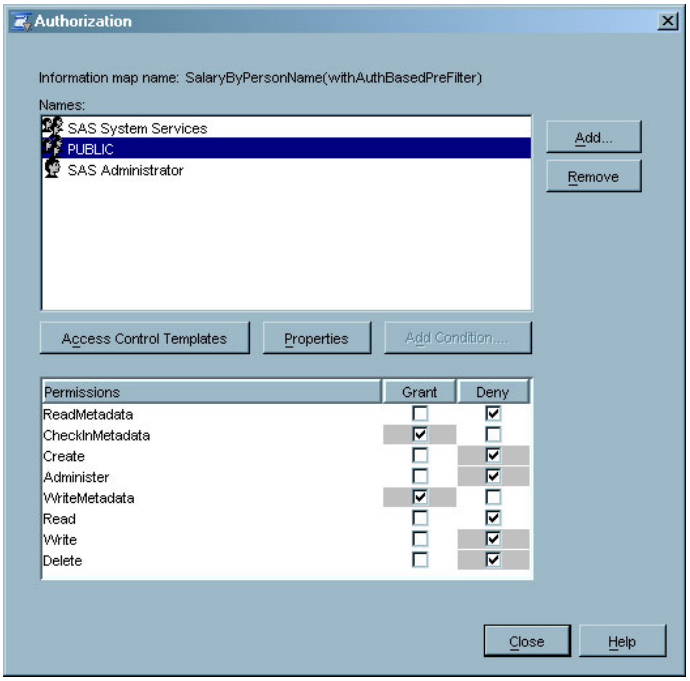
Access Class (User Group)	Information Map
All rows (Human Resources)	Grant R, RM
No rows (PUBLIC)	Deny[1] R, RM
Some rows (SASUSERS)	Grant R[2], RM

[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] To narrow this direct grant of Read permission as appropriate for each member of SASUSERS, you can use the byPersonName filter that you created in the main example.

To set these permissions, complete the following steps:

1. Prepare the information by using either of these methods:
   - □ To create a new information map for this variation, follow the instructions for steps 1 and 2 in the main example. Use a name such as *SalaryByPersonName (with AuthBasedPreFilter)* when you save the information map.
   - □ To reuse the information map from the main example, save that map with a different name and then deassign any filters that were assigned on the **`General Prefilters`** tab.
2. With the information map open, select **Tools ▶ Authorization** to open the Authorization dialog box.
3. In the **`Names`** box, select **`PUBLIC`**. In the **`Permissions`** list, select the **`Deny`** check box for the Read and ReadMetadata permissions.



Information map name: SalaryByPersonName(withAuthBasedPreFilter)

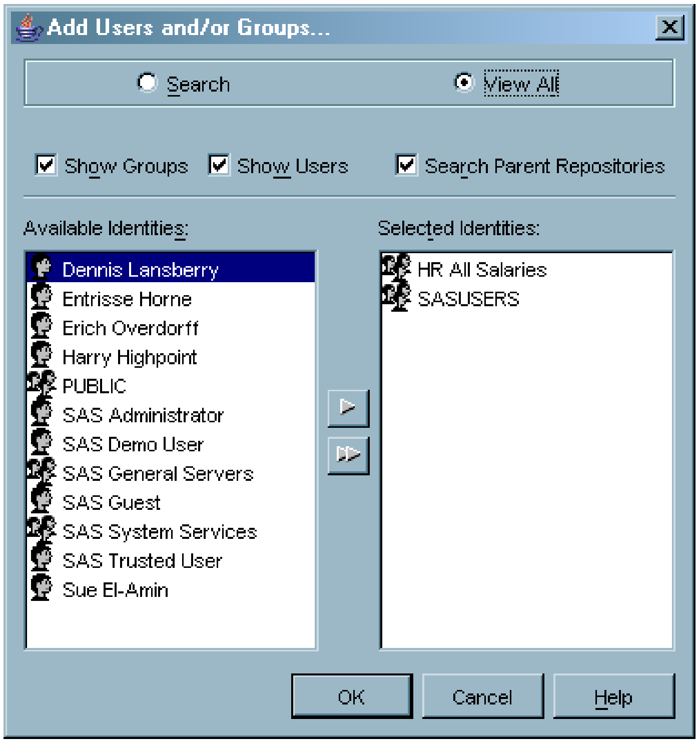
Names:

SAS System Services
PUBLIC
SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☐	☑
CheckInMetadata	☑	☐
Create	☐	☑
Administer	☐	☑
WriteMetadata	☑	☐
Read	☐	☑
Write	☐	☑
Delete	☐	☑

Note: Make sure that the check boxes for the Read and ReadMetadata permissions have a white background color. This indicates that these settings are direct permissions. △

4 To add the HR All Salaries and SASUSERS group identities to the **Names** box, click **Add**, select these groups in the Add Users and/or Groups dialog box, and then click **OK**.



○ Search ● View All

☑ Show Groups ☑ Show Users ☑ Search Parent Repositories

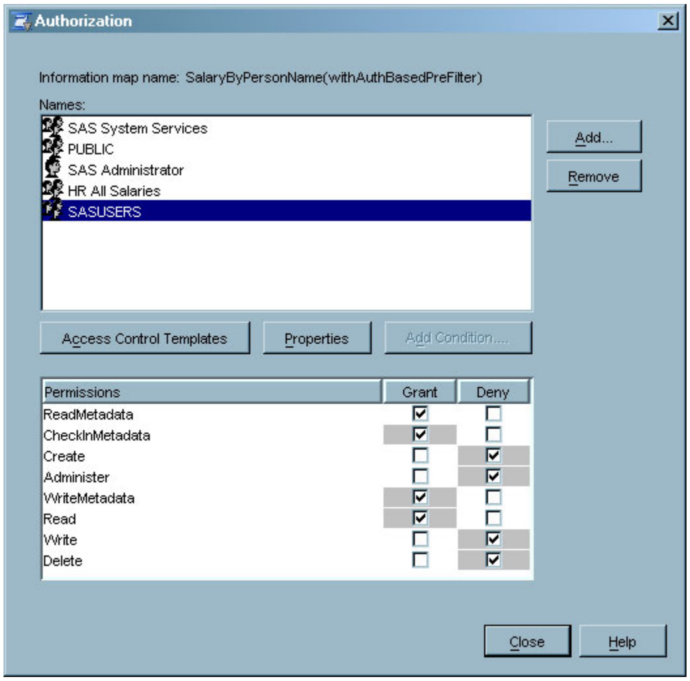
Available Identities:

Dennis Lansberry
Entrisse Horne
Erich Overdorff
Harry Highpoint
PUBLIC
SAS Administrator
SAS Demo User
SAS General Servers
SAS Guest
SAS System Services
SAS Trusted User
Sue El-Amin

Selected Identities:

HR All Salaries
SASUSERS

5 In the **Names** box, examine the settings for the SASUSERS group identity.



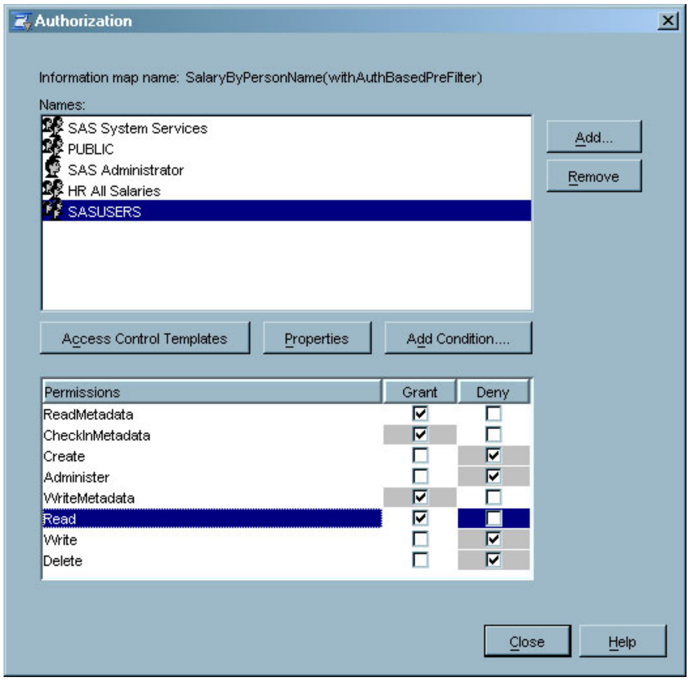
Information map name: SalaryByPersonName(withAuthBasedPreFilter)

Names:

SAS System Services
PUBLIC
SAS Administrator
HR All Salaries
SASUSERS

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	☑	□
Write	□	☑
Delete	□	☑

In the **Permissions** list, select the **Grant** check boxes to directly assign the Read and ReadMetadata permissions for this information map to the SASUSERS group.



Information map name: SalaryByPersonName(withAuthBasedPreFilter)

Names:

SAS System Services
PUBLIC
SAS Administrator
HR All Salaries
SASUSERS

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	☑	□
Write	□	☑
Delete	□	☑

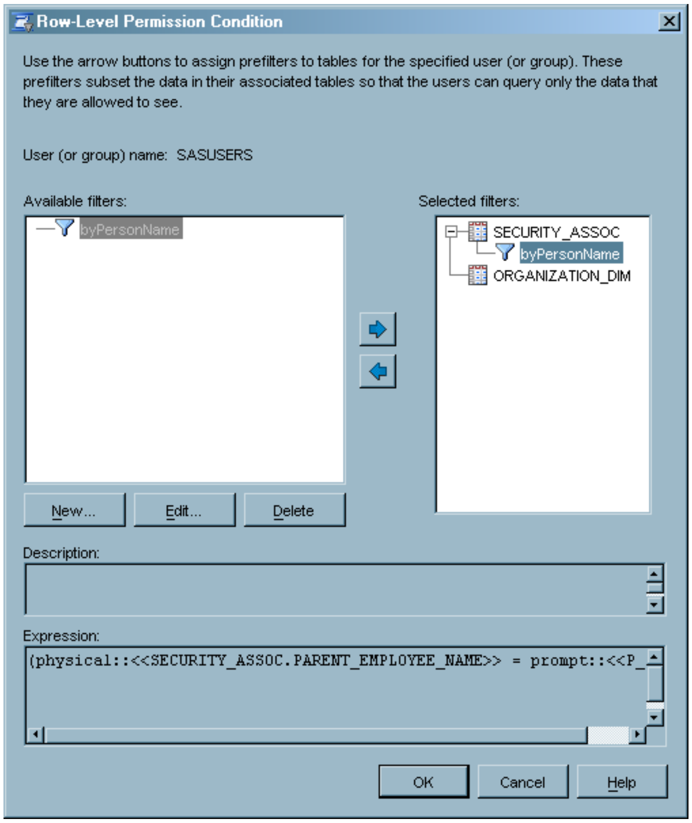
6 To limit the grant of Read permission that you just gave to the SASUSERS group, assign the byPersonName filter to that group as an authorization-based prefilter. Complete these steps:

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

Note: As the preceding figures illustrate, the **`Add Condition`** button became available when you added a direct 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.



Row-Level Permission Condition

Use the arrow buttons to assign prefilters to tables for the specified user (or group). These prefilters subset the data in their associated tables so that the users can query only the data that they are allowed to see.

User (or group) name: SASUSERS

Available filters: byPersonName

Selected filters: SECURITY_ASSOC, byPersonName, ORGANIZATION_DIM

Description:

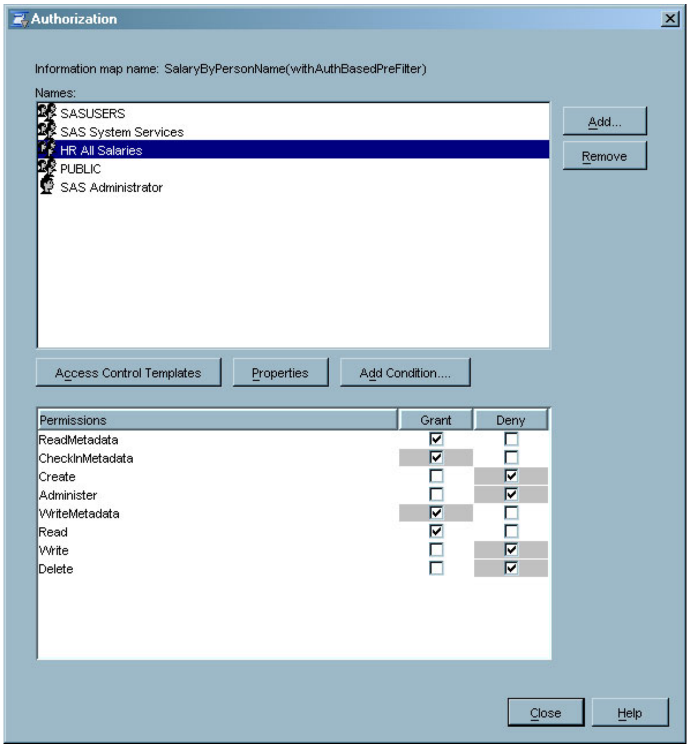
Expression:

```
(physical::<<SECURITY_ASSOC.PARENT_EMPLOYEE_NAME>> = prompt::<<P_
```

Note: Unlike a filter that you assign on the **`General Prefilters`** tab, this filter will apply 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 **`Names`** box, select the HR All Salaries group identity. In the **`Permissions`** list, select the **`Grant`** check box for the Read and ReadMetadata permissions. (white background).



Information map name: SalaryByPersonName(withAuthBasedPreFilter)

Names:

- SASUSERS
- SAS System Services
- HR All Salaries
- PUBLIC
- SAS Administrator

Permissions	Grant	Deny
ReadMetadata	☑	□
CheckInMetadata	☑	□
Create	□	☑
Administer	□	☑
WriteMetadata	☑	□
Read	☑	□
Write	□	☑
Delete	□	☑

Note: Because you want this group to be able to view all salaries, you will not constrain the direct grant of Read permission by adding a permission condition. △

8 In the Authorization dialog box, click `Close`.

9 To make your changes take effect, save the information map.

With these access controls in place, the rows that are retrieved vary as follows:

□ Users who do not have individual metadata identities will not be able to see or use the information map.

□ Users who have individual metadata identities but are not listed in the security associations table will see the information map, but will retrieve no rows.

□ Users who have individual metadata identities, are listed in the security associations table, and are not members of the HR All Salaries group will be able to view only those rows that contain data for their direct and indirect reports.

□ Users who are members of the HR All Salaries group will be able to retrieve all rows.

CHAPTER

11

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. The OLAP member-level permissions feature offers these advantages:

- This feature is integrated with other SAS Intelligence Platform administrative functions. OLAP member-level permissions are assigned to existing metadata identities, stored in the metadata repository, and evaluated by the SAS Metadata Server's authorization facility.
- This feature uses the standard authorization GUI for the SAS Intelligence Platform from within SAS Management Console.
- 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. This enables the server to leverage 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:

- Permission conditions constrain only direct ACE grants of the Read permission.
- With OLAP data, permission conditions can be specified only on dimension objects.
- If more than one permission condition applies to an identity, then the condition that is assigned at the highest level of identity precedence is applied. Other conditions that also apply to a user because of group memberships do not provide additional, cumulative access (unless there is an identity precedence tie between multiple groups at the highest level of identity precedence).
- 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.

Format for an OLAP Permission Condition

The format of the MDX expression that you use to define an OLAP permission condition varies slightly depending on whether you are filtering against the default hierarchy for the selected dimension. For example, consider a cube that has this structure:

```
Cube: OrionStar
 |
 |-Dimensions
 |  |
 |  |-Organization
 |  |  |
 |  |  |-Hierarchies
 |  |     |-Organization (All)
 |  |     |-Stores (default)
 |  |     |  |
 |  |     |  |-Levels
 |  |     |     |-Stores (All)
 |  |     |     |-East
 |  |     |     |  |
 |  |     |     |  |-Members
 |  |     |     |     |--Boston
 |  |     |     |     |--Washington D.C.
 |  |     |     |     |--Atlanta
 |  |     |     |-North
 |  |     |     |-West
 |  |     |     |-South
 |  |     |
 |  |     |-Staff
 |  |        |-Staff (All)
 |  |        |-Departments
 |  |        |-Employees
 |  |        |-Salaries
 |  |
 |  |-Products
 |  |-Sales
 |  |-Time
 |
 |-Measures
```

For this cube, you could set the following permission conditions on the Organization dimension:

- □ To enable GroupA to read only the East level of the Stores hierarchy, assign this condition to GroupA:

  ```
  {[Organization].[Stores].[East].members}
  ```

- □ To enable GroupB to read all members of the Stores hierarchy in the Organization dimension *except* for the members in the East level, assign this condition to GroupB:

```
EXCEPT({[Organization].[Stores].members}, {[Organization].[Stores].[East].members})
```

□ To limit the display of the Staff hierarchy for GroupC to show only the Departments level, assign this condition to GroupC:

```
{[Stores].defaultmember}<!--CONDITION-->{[Staff].[Departments].members}
```

Notice that because this expression filters against a hierarchy (Staff) that is not the default hierarchy for the Organization dimension, it must also include a member from the default hierarchy (Stores). The initial part of the expression, which concludes with the right angle bracket (>), meets the requirement by including the default member of the default hierarchy.

Related Topics

"MDX Queries and Syntax" in the *SAS OLAP Server: MDX Guide*

"Building Cubes" in the *SAS OLAP Server: User's Guide*

How to Assign an OLAP Permission Condition

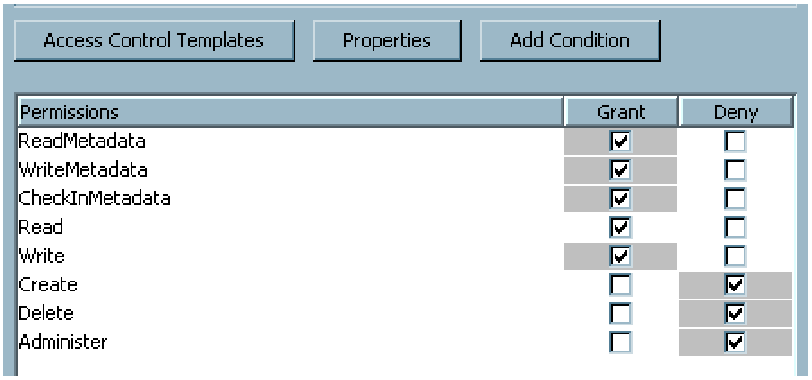
To assign an OLAP permission condition, complete these steps:

1 In SAS Management Console, navigate to **Environment Management ▶ Authorization Manager ▶ Resource Management ▶ By Location ▶ <your application server> ▶ <your OLAP schema> ▶ <your cube> ▶ Dimensions** and double-click the dimension for which you are defining a permission condition.

2 In the dimension's Properties dialog box, select the **`Authorization`** tab.

3 On the **`Authorization`** tab, select (or add) the user or group whose read access you want to limit with a permission condition.

4 In the permissions list, add a direct grant of the Read permission for the identity that you selected. The **`Add Condition`** button is now enabled.



Permissions	Grant	Deny
ReadMetadata	☑	□
WriteMetadata	☑	□
CheckInMetadata	☑	□
Read	☑	□
Write	☑	□
Create	□	☑
Delete	□	☑
Administer	□	☑

Notice that the check box for the Read permission has no added background color; this indicates that the permission is directly granted.

5 Click **`Add Condition`** to open the Permission Condition dialog box.

6 In the Permission Condition dialog box, enter an MDX expression that filters the current dimension as appropriate for the identity that you selected in step 3.

7 Click **`OK`** to save the permission condition.

Related Topics

"Access Requirements for OLAP Data" on page 130

"Example: Denying Most Users Read Permission to a Particular OLAP Cube" on page 119

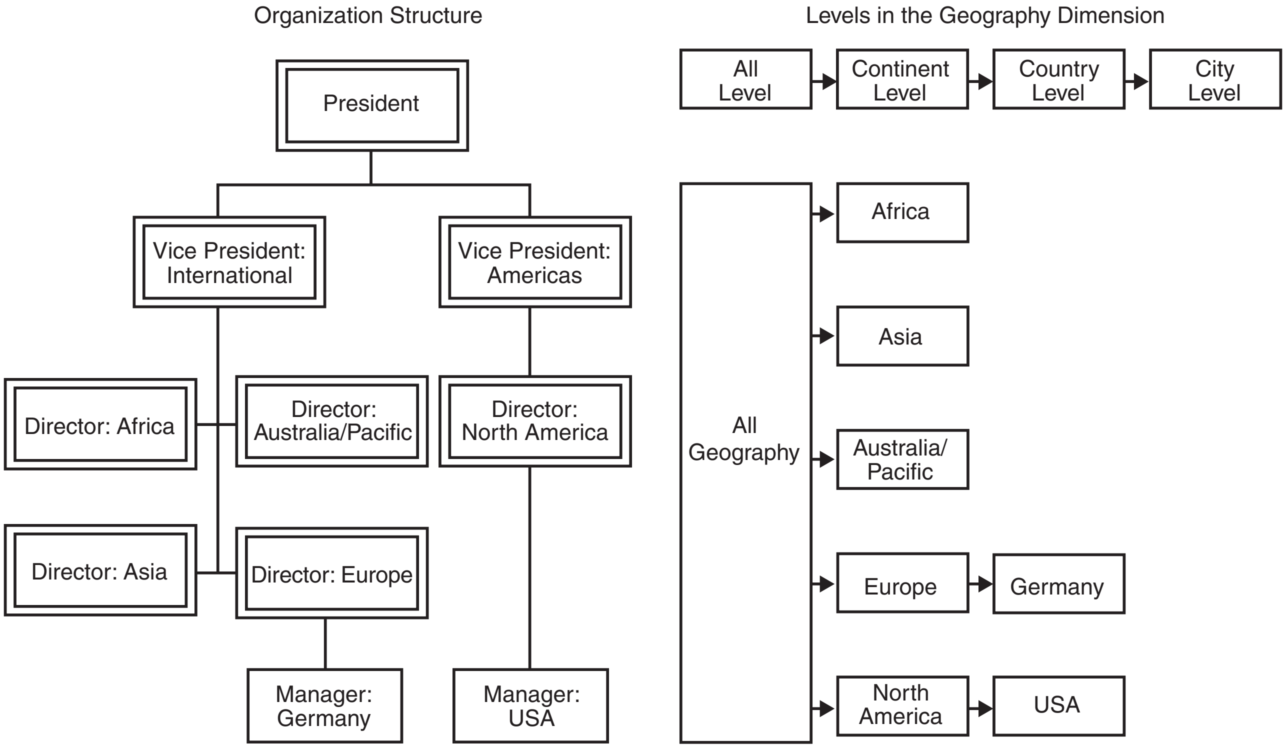
Example: Using Member-Level Permissions

Introduction, Assumptions, and Data Model

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 shows 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

To meet the security goals in this example, complete these steps:

1 In SAS Management Console, navigate to the cube under **Environment Management ▶ Authorization Manager ▶ Resource Management ▶ By Location ▶ SASMain ▶ SASMain-OLAP Schema**

2 Set these permissions for the entire cube:*
   - □ Give the PUBLIC group direct 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 direct grants of the Read and ReadMetadata permissions for the entire cube. This user sees all of the data.
   - □ Give the SAS System Services Group a direct 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 direct grants of the Read and ReadMetadata permissions.

3 Figure out the expressions that you will use to limit read access within the cube as appropriate for each executive. This table contains MDX expressions that you could use to subset the data based on each executive's area of responsibility.

Table 11.1 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.
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}	

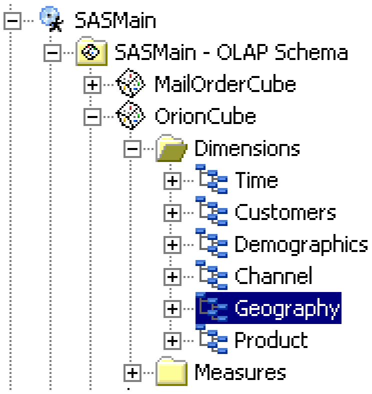
* For instructions, see "Example: Denying Most Users Read Permission to a Particular OLAP Cube" on page 119.

User	MDX Expression	Notes
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.

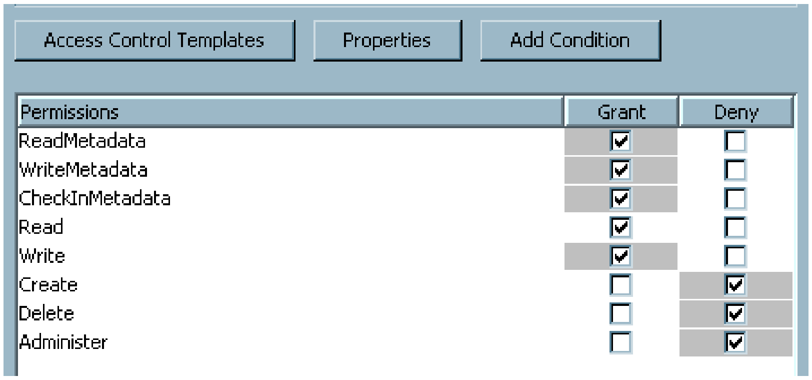
4 In SAS Management Console, define metadata that assigns the correct condition to each executive who should have limited access to the data. Complete these steps:

a Within the cube, navigate to and double-click the **Geography** dimension.



b On the **Authorization** tab of the Properties dialog box, select (or add) the Vice President Americas user.

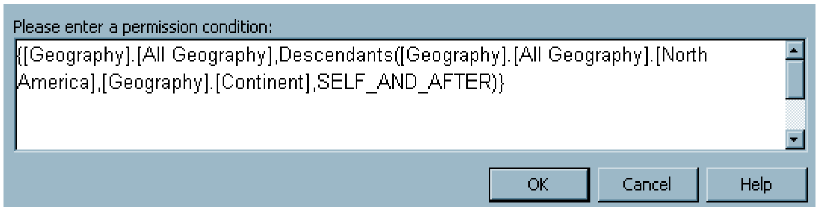
c In the permissions list, add a direct grant of the Read permission for the Vice President Americas user. The **Add Condition** button is now enabled.



Permissions	Grant	Deny
ReadMetadata	☑	□
WriteMetadata	☑	□
CheckInMetadata	☑	□
Read	☑	□
Write	☑	□
Create	□	☑
Delete	□	☑
Administer	□	☑

Notice that the check box for the Read permission has no added background color; this indicates that the permission is directly granted.

d Click **Add Condition** to open the Permission Condition dialog box. In the Permission Condition dialog box, enter the MDX expression for the Vice President Americas user. This condition limits the direct grant of the Read permission that you just gave to this user.



Please enter a permission condition:

{[Geography].[All Geography],Descendants([Geography].[All Geography].[North America],[Geography].[Continent],SELF_AND_AFTER)}

e Click **OK** to save this permission condition and then return to step 4b to repeat the process for the next executive.

PART 5

Appendixes

APPENDIX 1

Who's Who in the SAS Intelligence Platform

Standard User Metadata Identities

The following table lists standard user identities that you will see in the User Manager node in SAS Management Console. Most of these identities are created during installation. Not all deployments require all identities.

Table A1.1 Standard User Metadata Identities

User Name (account)	Functions and Capabilities
SAS Administrator (sasadm)	An unrestricted account that has full access to all metadata (other than passwords, which this account can overwrite but cannot read) and can administer all SAS servers. Use this account only to perform tasks that require unrestricted privileges (such as resetting a password for another user). Use this account only with SAS Management Console. Do not use this account to define a database library or to perform other tasks that involve reading passwords from the metadata.
SAS Trusted User (sastrust)	A trusted account that can impersonate other users on connections to the metadata server. Impersonation functionality is used by the SAS OLAP Server, report scheduling, and Web applications that are using Web authentication. In the default configuration, this account is used by the object spawner to read server definitions. In the documented workspace pooling configuration, this account functions as the pool administrator.
SAS Demo User (sasdemo)	A regular account that can serve as a generic end-user account to test any of the SAS client applications. This account is not required for configuration and can be removed safely.

User Name (account)	Functions and Capabilities
SAS Web Administrator (saswbadm)	A service account that SAS Information Delivery Portal and SAS Web Report Studio use to perform administrative functions.
SAS Guest User (sasguest)	A surrogate account that some Web applications use to define access for users who do not have individual metadata identities. SAS Information Delivery Portal uses this account to define the content of the Public Kiosk. In the recommended configuration, SAS Web Report Studio uses this account as a surrogate user for users who do not have their own metadata identities.

Note: The SAS General Server User is not listed in this table because there is no user definition or individual metadata identity for the SAS General Server User. This account (sassrv) is included as a login for the group definition for the SAS General Server Users group. By default, this service account is the process owner for stored process servers. Stored process servers and SAS/SHARE servers use this account to communicate with the metadata server. △

Related Topics:

"Minimizing the Availability of Accounts" on page 76

"How to Designate an Unrestricted, Administrative, or Trusted User" on page 93

Standard Group Metadata Identities

The following table lists the standard group identities that you will see in the User Manager node in SAS Management Console. Except where otherwise indicated, these identities are created during installation and membership is defined by you. Not all deployments require all identities.

Table A1.2 Standard Group Metadata Identities

Group Name	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. In most cases, users who do not have a individual metadata identities have the access that has been defined for the PUBLIC group.[1]
SASUSERS	A standard group with implicit membership. This group includes those members of the PUBLIC group who have a metadata identity. All members of the SASUSERS group are also members of the PUBLIC group.
Administrators	An optional group that you can create for convenience in managing access controls for your metadata administrators.
SAS System Services	A standard group whose membership includes the SAS Trusted User and the SAS Web Administrator.
SAS General Servers	A standard group whose membership includes the SAS Trusted User.
Portal Admins	A standard group for the SAS Information Delivery Portal. Membership includes the SAS Web Administrator.

Group Name	Description
Portal Demos	A standard group for the SAS Information Delivery Portal. Membership includes the SAS Demo User and any other demonstration accounts that you create.
WRS Report Consumer[2]	A standard role for SAS Web Report Studio. Membership in this group enables users to copy, move, save, rename, and delete reports.
WRS Report Author[2]	A standard role for SAS Web Report Studio. Membership in this group provides all of the capabilities of the report consumer role and also enables users to create and schedule reports.
WRS Advanced User[2]	A standard role for SAS Web Report Studio. Membership in this group provides all of the capabilities of the report author role and also enables users to distribute and archive reports.
WRS Administrator	A standard role for SAS Web Report Studio. Membership in this group enables users to perform all SAS Web Report Studio tasks, including creating and deleting recipient lists for report distribution. Unlike the SAS Administrator, members of this group must also have appropriate metadata permissions in order to perform any task.
WRS Prohibited	A standard role for SAS Web Report Studio. Membership in this group prevents users from logging on to SAS Web Report Studio, regardless of any other role assignments.

[1] In some circumstances, users who do not have metadata identities have the access that has been defined for a surrogate user (rather than having the access that has been defined for the PUBLIC group). In the public kiosk of the SAS Information Delivery Portal, all users have the access that has been defined for the SAS Guest User. In the recommended configuration for SAS Web Report Studio, a user who does not have a metadata identity has the access that has been defined for a surrogate user.

[2] By default, all SAS Web Report Studio users implicitly have this role. However, if you explicitly assign any members to the role, then only the explicitly assigned members will have the role.

Note: A role is a special type of group that is used to control the availability of actions within an application. △

Related Topics:

"Creating a Group of Metadata Administrators (Optional)" on page 69

"Using SAS Web Report Studio Roles" in the chapter "Managing SAS Web Report Studio Content and Users" in the *SAS Intelligence Platform: Web Application Administration Guide*

APPENDIX 2

Bulk-Load Processes for Identity Management

Overview of Identity Bulk-Load Processes

Introduction to Identity Bulk-Load Processes

As an alternative to using the User Manager in SAS Management Console, you can use bulk-load processes for these identity management tasks:

- □ Perform an initial import of identity information from an external enterprise source into the metadata server.
- □ Periodically update the identity information on the metadata server with current information from the external source.

SAS provides the following items that you can use to define identity bulk-load processes for your environment:

canonical tables
: specially formatted tables in which identity information is stored during bulk processing. For reference information, see "Canonical Tables" on page 201.

macros
: SAS autocall macros that you can use to perform specific tasks, such as creating canonical tables or loading user and group definitions that originate from an enterprise identity source. For reference information, see "Bulk-Load Macros" on page 207.

sample applications
: SAS code that uses the bulk-load macros to perform bulk processing of identity information from enterprise information sources. Two sample applications (**importad.sas** for Active Directory and **importpw.sas** for UNIX /etc/passwd files) are described in this chapter. The code for these applications is located in the SAS Sample Library.

  Note: On Windows, the sample library is in *SAS-install-dir*`\SAS 9.1\core\sample`. On UNIX, the library is located in *SAS-install-dir*`/SAS_9.1/samples/base`. △

Supported Associations

The metadata server supports the following relationships between identity data:

- □ A person can have multiple locations, e-mail addresses, and telephone numbers. However, each person can have only one item of a given type. For example, a person can have one home e-mail address and one work e-mail address, but not two work e-mail addresses.
- □ A person can be a member of multiple groups.
- □ A person or group can have multiple logins (but no more than one login for each authentication domain).
- □ A group can be a member of other groups.

Integrity Constraints

As explained in "Uniqueness Requirements for Names and User IDs" on page 10, user and group definitions must meet certain criteria before they can be added to a SAS

Metadata Server. The following list summarizes the constraints on identity metadata that are most relevant to the identity bulk-load processes:

- □ Person names and group names must be unique in the metadata server. That is, if a person or group name is already defined on the metadata server, another with the same name cannot be added.
- □ A person or group can have multiple logins; however, a particular user ID must be associated with only one person or group.
- □ The user ID in a login must be unique within the authentication domain in which it is used. For example, the person "Joe Programmer" cannot have two logins with a user ID of "joe" that are both associated to the Oracle authentication domain.
- □ The user IDs for Windows logins must be qualified with the Windows domain that owns the account.
  - □ If the login is for a network domain, then the user ID should take the form of *domain-name\userid*.
  - □ If the account is local to a specific machine, then the user ID should be in the form of *machine-name\userid*.

Note: When updating identities, you cannot add and remove a definition for a person or a group that has the same name at the same time.

If you follow the processes defined in this documentation, then errors that violate these constraints will be detected by the %MDUCHGV macro before an attempt is made to load the changes into the metadata server. You can then check the error data set created by %MDUCHGV and take corrective action before attempting to make the changes on the server. △

Understanding External Identities

In order to synchronize identity information between two disparate sources, your program must be able to map an entry in one source to its corresponding entry in the other source. This list explains how this mapping is accomplished:

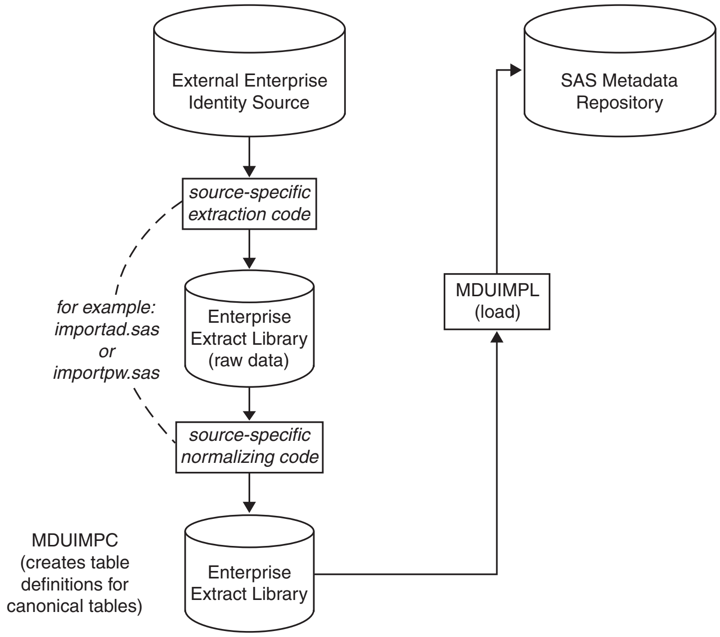
1. In your external source, you select a field to use for the mapping. This should be a field that contains a unique and unchanging value for each identity that you want to manage with batch processes. Typically, this will be an identifier such as employee number.
2. When you perform an initial import from your external source into the metadata, the external identity value for each user or group is stored in the metadata as part of an ExternalIdentity object. Each imported identity has at least one external identity value.
3. During the comparison phase of the synchronization process, external identity values serve as the key between the tables that are extracted from the external source and the tables that are extracted from the metadata.

How to Perform an Initial Import of Identity Information

Overview of the Initial Import Process

The following figure provides a generalized depiction of the process for importing identity information and indicates how the %MDUIMPC and %MDUIMPL macros can be used in this process.

Figure A2.1 Initial Import Process for Identity Information


External Enterprise Identity Source
source-specific extraction code
Enterprise Extract Library (raw data)
source-specific normalizing code
Enterprise Extract Library
for example: importad.sas or importpw.sas
MDUIMPC (creates table definitions for canonical tables)
MDUIMPL (load)
SAS Metadata Repository

The remainder of this section describes the initial process in detail for Active Directory and UNIX /etc/passwd files, and provides some guidance for importing identity information from other sources.

Initial Import from Active Directory

Importing Identities from Active Directory

SAS provides a sample application (**importad.sas**) that you can use to bulk load identity information from an Active Directory server. To adapt and run the sample application, complete these steps:

1. Ensure that any metadata identities that have already been created on the metadata server are not included in the enterprise data source from which you will import identities. If you attempt to import a user or group that has the same name as an existing metadata identity, then the import will fail.
2. Review "Canonical Tables" on page 201 to understand the information that is stored and how the associations between objects are formed.
3. Read the information in "Using importad.sas" on page 189. This will enable you to determine how to map the user information fields in the sample application to the user information fields in your enterprise system. Comments in the application source code describe how to modify the applications to use different fields.
4. In the SAS Program Editor, open **importad.sas**. (This file is in the SAS Sample Library. On Windows systems, you can find the file in *SAS-install-dir*\SAS 9.1\core\sample, and on UNIX systems, the file is in *SAS-install-dir*/**SAS_9.1/samples/base**.)
5. Modify section 1 of the program to include site-specific metadata server and source server connection values. Connect as an *unrestricted user* (such as sasadm).

6 In sections 3 and 4 of the program, substitute site-specific attributes or modify other aspects of the application as needed.

Note: The %MDUIMPC macro uses a keyid variable to uniquely identify Person, Group, and AuthenticationDomain objects. For this reason, a Person, Group, or AuthenticationDomain cannot have the same keyid value. △

7 Submit the application.

8 Rename and save the application for use during synchronization.

Using importad.sas

The **importad.sas** program should be executed in a SAS system installation that includes SAS Integration Technologies software. SAS Integration Technologies software is available in installations that are used to run workspace servers.

The **importad.sas** program references standard Active Directory schemas to identify the user and group attributes that will be extracted, and it uses the LDAP CALL Routine interface to query the Active Directory server and perform the extraction. If your site has extended the standard schema with site-specific attributes, you might want to modify section 3 to reference additional or alternate attributes. Also, all users should examine sections 3 and 4 carefully to understand the LDAP filters that are used to retrieve persons and groups.

Active Directory will not return the full set of records if the number of records exceeds the maximum query limit. You can use filters to subset the data into sets that do not exceed the maximum query limit. When you combine these smaller sets, you will have all the data that you want to extract. You also can modify the filters to customize which entries are imported. For example, you could specify to import entries only for people who are members of a particular group.

The program requires you to specify a Distinguished Name in the Active Directory hierarchy for Person and Group searches to begin. You must also specify attributes that will serve as the keyid variables for Person and Group information. The default keyid variable for a Person is the employee ID. For a Group, the Distinguished Name is used. If you have groups stored in two different areas of the Active Directory server, then you need to duplicate the extraction code to pull from the different base DNs for the groups.

Other important variables that you need to define include the following:

ADExtIDTag
specifies the source of the identity metadata to be loaded onto the metadata server. The default value is **Active Directory Import**. This value should not be quoted. This value is stored in an ExternalIdentity metadata object that is created and associated with each Person and Group metadata object created by the import.

MetadataAuthDomain
specifies the name of a metadata authentication domain to which logins created by the process should be associated. This name can be, but is not required to be, the same name as the Windows domain. Logins from multiple Windows domains can participate in the same metadata authentication domain if the Windows domains trust each other. For example, if you created a default authentication domain named DefaultAuth at installation, and there are logins that will be accessing these same servers and using the same authentication mechanism, then specify **DefaultAuth**.

WindowsDomain
specifies the name of the Windows domain to prepend to the user ID in Login objects created by the extraction. The metadata server requires Windows user IDs to be domain-qualified.

filename keepxml 'filename.xml'
specifies a location to store the XML created by the %MDUIMPL macro. Specify a complete filename, including the path and .xml extension.

Initial Import from UNIX /etc/passwd Files

Importing Identities from UNIX /etc/passwd Files

SAS provides a sample application (**importpw.sas**) that you can use to bulk load identity information from a UNIX /etc/passwd file. To adapt and run the sample application, complete these steps:

1. Ensure that any metadata identities that have already been created on the metadata server are not included in the enterprise data source from which you will import identities. If you attempt to import a user or group that has the same name as an existing metadata identity, then the import will fail.
2. Review "Canonical Tables" on page 201 to understand the information that is stored and how the associations between objects are formed.
3. Read the information in "Using importpw.sas" on page 190. This will enable you to determine how to map the user information fields in the sample application to the user information fields in your enterprise system. Comments in the application source code describe how to modify the applications to use different fields.
4. In the SAS Program Editor, open **importpw.sas**. (This file is in the SAS Sample Library. On Windows systems, you can find the file in *SAS-install-dir***\SAS 9.1\core\sample**, and on UNIX systems, the file is in *SAS-install-dir***/SAS_9.1/samples/base**.)
5. Modify section 1 of the program to include site-specific metadata server and source server connection values. Connect as an *unrestricted user* (such as sasadm).
6. In sections 3 and 4 of the program, substitute site-specific attributes or modify other aspects of the application as needed.

   Note: The %MDUIMPC macro uses a keyid variable to uniquely identify Person, Group, and AuthenticationDomain objects. For this reason, a Person, Group, or AuthenticationDomain cannot have the same keyid value. △
7. Submit the application.
8. Rename and save the application for use during synchronization.

Using importpw.sas

The **importpw.sas** program can import information from /etc/passwd files that include user information in a *gcos* field in the following form:

```
user name : password : uid (numeric) : primary group id : <continued>
gcos-field: home-directory : login-shell
```

where *gcos-field* consists of additional comma-delimited fields in the following form:

```
Person Name, Office, phone extension, misc, employeeid
```

The following is an example of what a password entry would look like in an /etc/passwd file that has a *gcos* field:

```
user name : password : uid (numeric) : primary group id : <continued>
Person Name, Office, phone extension, misc, employeeid:<continued>
```

```
home-directory : login-shell
```

The placement of colons and commas is important. Use colons to delimit standard fields in the entry. The commas delimit additional fields to include in the file. The commas are not required. You can use any delimiter that you want, except a colon, since it identifies the standard fields. If your /etc/passwd file does not match this layout, you will need to modify the sample program to either exclude information from the extraction or to extract it from the appropriate fields.

Note that the *employeeid* field of the *gcos* field is required. If the *employeeid* field for a particular /etc/passwd entry is empty, then that entry will be dropped from the import. The values in the *Person Name* field also 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 to extract group information from an /etc/group file. There are no special fields in this file. The format of a standard /etc/group file is as follows:

```
groupname : password : gid (numeric) : comma-delimited list of users
```

The users in the comma-delimited list of users are identified by user name rather than by a numeric user ID. Note that groups are not allowed to be members of other groups.

To disable a user or group entry, enter an asterisk (*) in the *password* field. The program drops those entries that contain an asterisk in the *password* field.

Other important variables that you need to define include the following:

PWExtIDTag
specifies the source of the identity metadata to be loaded onto the metadata server. The default value is **`Passwd File Import`**. This value should not be quoted. This value is stored in an ExternalIdentity metadata object that is created and associated with each Person and Group metadata object created by the import.

filename grpfile 'filename'
specifies pathname of the file from which to import group information. The default value is /etc/group.

filename pwfile 'filename'
specifies the pathname of the file from which to import person information. The default value is /etc/passwd.

MetadataAuthDomain
specifies the name of a metadata authentication domain to which logins created by the process should be associated. This name can be, but is not required to be, an actual network domain. This is a construct for associating logins to servers and processes for which the logins are valid. Depending on network configuration and trust relationships, multiple network domains can be associated with a single metadata authentication domain. For example, if you created a default authentication domain named DefaultAuth at installation, and there are logins that will be accessing these same servers and using the same authentication mechanism, then specify **`DefaultAuth`**.

UNIXEMAILDOMAIN
specifies a network domain. This domain will be appended to the user IDs extracted from the passwd file to form an e-mail address of the form *userid@value_of_variable*.

DUPLICATEPERSONS
specifies what to do when multiple accounts are encountered for the same person. The default behavior is to create a new Person object using the user ID as the person name.

filename keepxml 'filename.xml'
specifies a location to store the XML created by the %MDUIMPL macro. Specify a complete filename, including the path and .xml extension.

Initial Import from Other Sources

If your enterprise identity information is not in a format that is covered in the two previous topics, then your first step in the bulk-import process is to figure out how to extract the data from your enterprise source and make it available for use with the bulk-load macros. If you have an LDAP enterprise source, you might be able to modify the Active Directory sample application for your purposes. Otherwise, you can use the %MDUIMPC macro to define the canonical tables and then use DATA steps to extract the information from the external source and insert it into the tables.

After you have created and populated the canonical tables, use the %MDUIMPL macro to load the identity information into the metadata server. This macro uses the tables that were created with the %MDUIMPC macro to build the XML that is necessary to add the user information to a SAS Metadata Repository. The macro then invokes PROC METADATA to submit the XML to the server. Use the options that are described in "Connection Options for the SAS Metadata Server" on page 214 to identify the target metadata server and repository.

If you want to load group definitions that have logins, use the synchronization process to load the group definitions. The %MDUIMPL macro does not allow group definitions to have logins associated with them, but the corresponding synchronization macro, the %MDUCHGL macro, does allow logins on group definitions. The %MDUCHGL macro works similarly to the %MDUIMPL macro, except that it uses the change tables created by the %MDUCMP macro as its source of input; therefore, you must follow the steps in "How to Synchronize Imported Identity Information" on page 195 before you can execute the macro.

If you do not want to load or update the user and group definitions on the metadata server immediately, specify the options **submit=0** and **outrequest=*fileref*** to save the XML that they generate to a file. To execute the XML later, use PROC METADATA. For more information about PROC METADATA, see the METADATA procedure in the *SAS Open Metadata Interface: Reference* (available on SAS OnlineDoc 9.1.3).

If you want to specify information about an imported identity's origin, use the EXTIDTAG option. You can enter a descriptive string of up to 32 characters in this parameter. This string is stored as the Context attribute of an ExternalIdentity object that is created along with each Person and IdentityGroup object that is created in the target repository. If you do not specify a value for this option, the default value **IdentityImport** will be used. Any value you specify should not be quoted.

Sample Code for Generic Identity Import

The following program includes some sample identity data and demonstrates one way that you can import generic identity information. The purpose of the example is 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 executes the %MDUIMPC macro without options and uses macro variables to define the tables and columns. Input data is supplied in DATALINES statements. After the tables are created, the program executes the %MDUIMPL macro to create XML from the tables and to submit the XML to the metadata server. The OUTREQUEST option specifies to save the XML to the fileref TESTXML.

```
/*-------------------------------------------------------
 * Example use of %MDUIMPC and %MDUIMPL macros for importing
 * user information into the SAS Metadata Server.
 *------------------------------------------------------- */
/*---------------------------------------------------------------------------
 * Use the META* options to specify the metadata server connection options
 * where the user information will be loaded.
 *-------------------------------------------------------------------------*/
options metaserver=myserver
        metaport=8561
        metauser="mydomain\userid"
        metapass="mypassword"
        metaprotocol=bridge
        metarepository=Foundation;

/* Initialize the macro variable with default values.  Do not create empty 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 &logincol1;  */
         input keyid userid password authdomkeyid;
    datalines;
P001,WinNet\Michelle, ,A001
P001,Michelle, ,A002
P002,WinNet\Fred, ,
P003,WinNet\Brian, ,
P003,Brian, ,A002

;
/* Now, load the information contained in the data sets above into the */
/* metadata server.  Defaults will read the data sets from the Work    */
/* library.                                                           */
filename testxml 'filename.xml' lrecl=1024;

%mduimpl(outrequest=testxml);
```

How to Synchronize Imported Identity Information

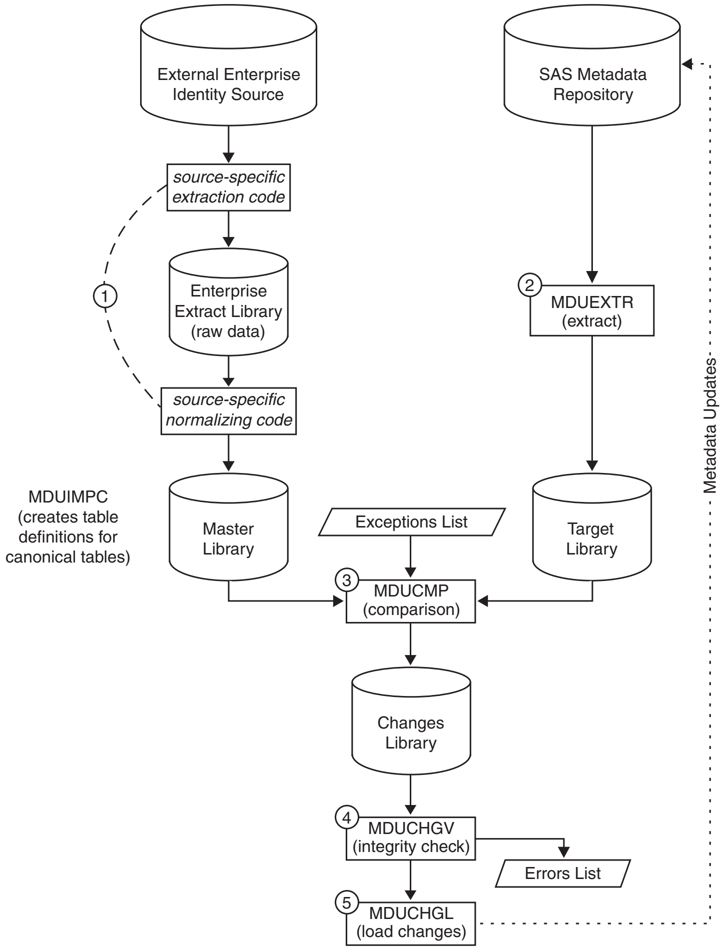
Overview of the Identity Synchronization Process

Metadata identities that are imported into a SAS Metadata Server using SAS bulk-load processes can be periodically compared to current information from the external source to ensure that the metadata definitions are current. Use the following general process to compare and update identity metadata with current information from an external source:

1. Rebuild the master canonical tables with current information from the external source.
2. Create canonical tables from the identities on the metadata server.
3. Compare the master tables to the server tables.
4. Validate the changes before updating the metadata server (optional).
5. Update the metadata server with any changes that are found.

The following figure depicts the process and indicates how the macros that SAS provides can be used in this process. The remaining topics in this section describe each step in detail.

Figure A2.2 Synchronization Process for Identity Information


External Enterprise Identity Source
SAS Metadata Repository
source-specific extraction code
Enterprise Extract Library (raw data)
1
2
MDUEXTR (extract)
source-specific normalizing code
Metadata Updates
MDUIMPC (creates table definitions for canonical tables)
Master Library
Exceptions List
Target Library
3
MDUCMP (comparison)
Changes Library
4
MDUCHGV (integrity check)
Errors List
5
MDUCHGL (load changes)

1. Rebuilding the Master Canonical Tables

It is recommended that you use the same program that you used to create the master canonical tables to rebuild them. For example, if you used the **importad.sas** or **importpw.sas** sample applications to import identities, then you would use these applications to create new versions of the master tables. The applications would extract current information from the Active Directory server or /etc/passwd files, and overwrite the master canonical tables with new ones. You will need to suppress execution of the %MDUIMPL macro (which loads the identities on the metadata server) when you execute the applications. This can be done by specifying the macro variable **_EXTRACTONLY=1** before calling the applications.

2. Creating Canonical Tables from the SAS Metadata Server

In order to compare the content of the rebuilt master canonical tables to the metadata identities on the metadata server, you need to create canonical tables of the server identities. SAS provides the %MDUEXTR macro to extract identity metadata from the metadata server and write it into canonical tables. The %MDUEXTR macro extracts all Person and IdentityGroup metadata objects that are found on the server (not just imported ones) and their associated location, login, telephone, and e-mail information. It then creates tables similar to the ones described in "Canonical Tables" on page 201, except the tables have two additional columns:

objectid
: specifies the identity's unique metadata object identifier. This value is used to map updates to the extracted object back to the source object on the metadata server.

extid
: specifies an ExternalIdentity object identifier. The %MDUIMPL and %MDUCHGL macros create an ExternalIdentity object for every Person and IdentityGroup object that they create on the SAS Metadata Server. Identities that are defined in the User Manager in SAS Management Console do not have an ExternalIdentity object (or an extid value) associated with them.

3. Comparing the Master and SAS Metadata Server Canonical Tables

Overview of the Comparison Process

SAS provides the %MDUCMP autocall macro to enable you to compare the master and server canonical tables. This macro creates change tables that contain any changes that are found. The following list highlights important points to note about the comparison process:

- □ The comparison process ignores passwords, unless the password belongs to a new user or group definition. To change the password of an existing user or group definition, you must use the SAS Personal Login Manager or the User Manager in SAS Management Console. Note that only the password owner and the *unrestricted user* can change a password, and that the *unrestricted user* can overwrite an existing password with a new one, but this user cannot view an existing password.
- □ By default, identities that were created manually in SAS Management Console are excluded from the synchronization process. However, manual updates that have been made to login, location, telephone number, and e-mail information that was initially imported will be overwritten by the synchronization process, unless you define exceptions to that process. See "Defining an Exception to the Comparison Process" on page 198 for steps that you can take to preserve manual updates to imported information.
- □ By default, the comparison process will mark new authentication domains found in the master canonical tables for addition to the target, but it will not mark an authentication domain that is not represented in the master tables for deletion. This is done to protect authentication domains that might have been defined in SAS Management Console. If you want to be able to compare only imported authentication domains and to delete domains that do not have a matching domain in the master tables, set the AUTHDOMCOMPARE parameter to **`AUTHDOMCOMPARE=keyid`**.
- □ The %MDUCMP macro does not attempt to validate the information in the change tables. To ensure that the new information from the enterprise data source meets server-enforced integrity constraints, run the %MDUCHGV macro on the change data sets before loading the changes on the metadata server.

The %MDUCMP macro records differences in the master and target canonical tables in six change tables. These tables are described in "%MDUCMP Autocall Macro" on page 210.

Defining an Exception to the Comparison Process

An exception is something that you exclude from the comparison process. The %MDUCMP macro enables you to define exceptions in an exceptions data set that is submitted in the EXCEPTIONS parameter. This exceptions data set needs to have two 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. You can define an exception for any column in a canonical table. To view a list of the columns defined for each canonical table, see “Canonical Tables” on page 201. The exception value can be a real or assigned value. For example:

```
phone      PhoneType="SMC Phone"
email      EmailType="SMC Email"
logins     authDomKeyId="A002"
logins     userid="testid%"
```

In this example, the first two entries specify an assigned value. That is, it is a value that users can optionally enter in the **Type** field of the **Phone** and **Email** entry panels of the User Manager in SAS Management Console to indicate an intentional change to imported information. The **Type** field corresponds to the PhoneType column of the phone canonical table and to the EmailType column of the email canonical table. The entries specify to exclude Phone and Email objects that have the values "SMC Phone" and "SMC Email" from the comparison.

The last two entries specify real values. Of these, the first entry specifies to exclude from the comparison all Login objects that have an authentication domain key identifier of **A002** in the logins table. The authentication domain key identifier is assigned as a primary key in the authDomKeyId column of the authdomain table and is referenced in the logins table as a foreign key. The second entry specifies to exclude from comparison all Login objects which have a value that begins with **testid** in the userid column.

Note that tables and columns can be listed in the exceptions data set multiple times. All entries for a table will be applied.

4. Validating the Change Tables

A new or modified user and group definition will be added to the metadata server only if it meets server-enforced integrity constraints. These constraints are described in “Integrity Constraints” on page 186. SAS provides the %MDUCHGV autocall macro to enable you to validate that the changes identified in the change tables will not produce these errors when loaded by the %MDUCHGL macro. You should run the %MDUCHGV macro before executing the %MDUCHGL macro to update the metadata server.

The %MDUCHGV macro compares the contents of the change tables to the contents of the canonical tables created from the metadata server to ensure that no duplicate user identifiers, duplicate authentication domains, and other integrity violations are found. When violations are detected, the macro sets the variable MDUCHGV_ERRORS to a nonzero value and writes any violations to the data set specified in the ERRORSDS parameter. You can use the information in the ERRORSDS data set to address the problems in the change tables by hand in the User Manager in SAS Management Console. As an alternative, you can re-execute the %MDUCMP macro and submit the

ERRORSDS data set in the EXCEPTIONS= parameter to re-create the change tables without the offending changes. If an exceptions data set is already being used, you need to append the content of the ERRORSDS data set to that data set.

If you do not validate the change tables before updating the server, any changes that violate the server-enforced integrity constraints will cause an error when an attempt is made to update the metadata server. When this occurs, none of the changes are made to the server, and an error message is written to the SAS log.

5. Updating the SAS Metadata Server with Changed Information

SAS provides the %MDUCHGL autocall macro for updating metadata on the metadata server with information from the change tables. See the reference information for %MDUCHGL for more information.

Sample Code for Identity Synchronization

The following sample program contains code you might execute to update identities that were imported from a Microsoft Active Directory host server. The program assigns librefs to work directories, calls the **importad.sas** sample application to extract updated metadata, and sets the _EXTRACTONLY macro variable to cause **importad.sas** to simply extract updated information and create the canonical tables. The program then executes the %MDUEXTR macro to extract identity metadata from the metadata server, executes the %MDUCMP macro to compare the old and new information, executes the %MDUCHGV macro to validate the changes, and executes the %MDUCHGL macro to write the changes to the metadata server.

```
/*-------------------------------------------------------
* Sample program for synchronizing the user and group
* information in the metadata server with Active Directory.
* This sample assumes the metadata was originally populated
* from Active Directory using importad.sas or via this
* sample program.
*
* To use the program, modify the appropriate sections to set
* metadata system options, define libraries, and specify the
* location of the modified importad.sas sample program.
*-------------------------------------------------------*/

/*-------------------------------------------------------
* Define system options for the metadata server.
*-------------------------------------------------------*/
options metaserver=METADATASERVER.NETWORK.DOMAIN
        metaport=8561
        metauser="unrestricted-userid"
        metapass="password"
        metaprotocol=bridge
        metarepository="Foundation";

/* Define the library where the extracted metadata info will be stored. */
libname mdextr "drive:\path\mdextract lib";

/* Define the library where the extracted Active Directory information
/* will be stored. Note: This library must also be used in the
/* importad.sas file because this program will simply include the modified
/* sample program to perform the extraction. */
libname extextr "drive:\path\enterprise extract lib";

/* Define the library where the change data sets will be saved. */
libname extrnchg "drive:\path\enterprise change lib";


/*---------------------------------------------------------------------
* Step 1.
* When importad.sas is used for synchronization, you must turn %MDUIMPL
* off by defining the macro variable _EXTRACTONLY. Specifying a value
* for this variable has no effect; it is the existence of the _EXTRACTONLY
* macro variable that causes importad.sas to extract information
* and create the canonical tables without loading the information into
* the metadata repository.
*----------------------------------------------------------------------*/
%let _EXTRACTONLY = ;
%include "drive:\path\importad.sas";


/*----------------------------------------------------------------------
* Step 2.
*
* Execute %MDUEXTR to extract user information from the metadata
* server into a modified form of the canonical tables.
*----------------------------------------------------------------------*/
%mduextr(libref=mdextr);


/*----------------------------------------------------------------------
* Step 3.
*
* Execute the %MDUCMP comparison macro with the flag that specifies
* to compare only information that originated from enterprise sources.
* The information in the Enterprise extract lib is treated as the master,
* and the information in Mdextr lib is treated as the target.  Exception
* list processing is also possible at this step and may be needed to
* prevent the deletion of information added in SAS Management Console.
* Change tables from this step are stored in the extrnchg lib.
*-----------------------------------------------------------------------*/
%mducmp(master=extextr, target=mdextr, change=extrnchg, externonly=1);


/*----------------------------------------------------------------------
* Step 4.
*
* Validate the change tables using the %MDUCHGV macro.
* This macro will check the changes to see if they violate any server
* integrity constraints.  Information about violations will be written
* to a SAS data set. The information can be used by an administrator
* or other tool to correct problems before attempting to load the changes.
* NOTE: the violations data set may be fed into the %MDUCMP macro to
* regenerate the change tables without the offending changes.
*
```

```
* If problems are found, an error message will be written to the log.
* Code should be added below to inspect the 'returncode' macro and take
* steps to correct the problems.
*----------------------------------------------------------------------*/
%mduchgv(change=extrnchg, target=mdextr, temp=work, errorsds=work.mduchgverrors);


/*----------------------------------------------------------------------
* Step 5.
*
* Execute the %MDUCHGL macro to build an XML stream from the change
* tables and load the changes into the metadata server, bringing it
* into parallel with the enterprise source.  The change tables are read
* from the Enterprise change lib.
*
* If you place the call to %MDUCHGL in a macro, you will be able to
* check the results of %MDUCHGV and abort the load if errors were found.
*----------------------------------------------------------------------*/
%macro exec_mduchgl;
   %if (&MDUCHGV_ERRORS ^= 0) %then %do;
      %put ERROR: Validation errors detected by %nrstr(%mduchgv). Load not attempted.;
      %return;
      %end;
   %mduchgl(change=extrnchg, submit=1);
%mend;

%exec_mduchgl;
```

Identity Bulk-Load Processes: Reference

Canonical Tables

Overview of Canonical Tables

The bulk-load process uses a set of specially formatted tables (canonical tables) that you can populate with user and group information from an enterprise source. A *canonical table* is a table that defines the standard attributes and associations for an identity metadata object. Each metadata type in the SAS Metadata Model consists of a set of attributes and associations: the attributes describe the characteristics of the metadata type; the associations describe the metadata type's relationships with other types. SAS provides the %MDUIMPC autocall macro to enable you to create canonical tables for creating objects of the Person, IdentityGroup, AuthenticationDomain, Login, Location, Phone, and Email metadata types on the metadata server, and to create associations between them. The %MDUIMPC macro defines these canonical tables:

Table A2.1 Tables and Macro Variables Created by the %MDUIMPC Macro

Table Name	Purpose	Name/Keep List Global Variable Name	Column Attributes Macro Name
person	Generate Person objects	&persontbla;	%definepersoncols
location	Generate Location objects for associated Person	&locationtbla;	%definelocationcols
phone	Generate Phone objects for associated Person	&phonetbla;	%definephonecols
email	Generate E-mail objects for associated Person	&emailtbla;	%defineemailcols
idgrps	Generate IdentityGroup objects	&idgrptbla;	%defineidgrpcols
grpmems	Define IdentityGroup members	&idgrpmemstbla;	%defineidgrpmemscols
authdomain	Define AuthenticationDomain objects	&authdomtbla;	%defineauthdomcols
logins	Define Login objects with associated AuthDomain	&logintbla;	%definelogincols

Table Structure

The columns in each of the canonical tables are described in the following tables:

Table A2.2 Columns in the person Table

Column Name	Description
keyid	Primary key used to identify the person and to relate other information to this particular person
name	Name of the person
description	Description of the person
title	Person’s title

Table A2.3 Columns in the location Table

Column Name	Description
keyid	Foreign key identifying the person with whom the particular location is associated
locationName	Location name

Column Name	Description
locationType	Type of location
address	Street address
city	City name
postalcode	Postal code
area	Location area designation
country	Country name

Table A2.4 Columns in the phone Table

Column Name	Description
keyid	Foreign key identifying a person or group using a telephone number
phoneNumber	Telephone number
phoneType	Type of telephone number

Table A2.5 Columns in the email Table

Column Name	Description
keyid	Foreign key identifying a person or group using an e-mail address
emailAddr	E-mail address
emailType	Type of e-mail address

Table A2.6 Columns in the idgrps Table

Column Name	Description
keyid	Primary key identifying a group that is also used to relate other information to the group
name	Group name
description	Description of the group
grpType	Type of group

Table A2.7 Columns in the grpmems Table

Column Name	Description
grpkeyid	Foreign key to a group
memkeyid	Foreign key to a group or person that is a member of the group identified by *grpkeyid*

Table A2.8 Columns in the authdomain Table

Column Name	Description
keyid	Primary key identifying an authentication domain that is used to relate other information to the authentication domain
authDomName	Authentication domain name

Table A2.9 Columns in the logins Table

Column Name	Description
keyid	Foreign key identifying the person to whom the login belongs
userid	A user ID that is specified in the format of its authenticating domain
password	Password
authDomKeyId	Foreign key to an AuthenticationDomain

Table Requirements

One set of canonical tables should be used to create and maintain all imported metadata identities on the metadata server. Even if you are extracting user and group definitions from multiple external sources, you must use a single set of canonical tables to load and maintain information about the external sources on the metadata server.

A separate canonical table is required for each object type that will be imported. Any table that you create should also include all columns in the documented order.

Table Relationships

The %MDUIMPC macro uses a series of primary and foreign keys to create table relationships. Each canonical table contains a keyid column and columns representing the metadata type's attributes. The keyid value is a primary key that identifies a person, group, or authentication domain and relates it to other information. In the phone, location, email, and logins tables, the keyid is referenced as a foreign key in order to relate phone, location, e-mail and login information to a person. The grpmems table references the keyid defined in the idgrps table as a foreign key in the grpkeyid column in order to relate persons and groups identified in the memkeyid column as members of a particular group. The logins table references the keyid defined in the authdomain table as a foreign key in the authDomKeyId column in order to relate an authentication domain to a person.

Most enterprise identity systems assign a person an attribute for a person which can act as a natural keyid. For example, an employee identification number stored in Active Directory or the Distinguished Name attribute of the person as stored in an LDAP server make good keyid identifiers. An enterprise-wide employee identifier that does not change over time is the best choice for a keyid.

The figures that follow illustrate how data is stored in the tables and how the primary and foreign keys are used to relate data for a fictional user named "Michelle Harrell."

The first figure illustrates the relationship between the data defined for "Michelle Harrell" in the person, location, email, and phone tables. "Michelle Harrell" has separate home and office locations, e-mail addresses, and telephone numbers defined for her. This data is mapped directly to her key identifier in the location, email, and phone tables.

Figure A2.3 Relationships among the Data in the person, idlocation, location, email, and phone Tables

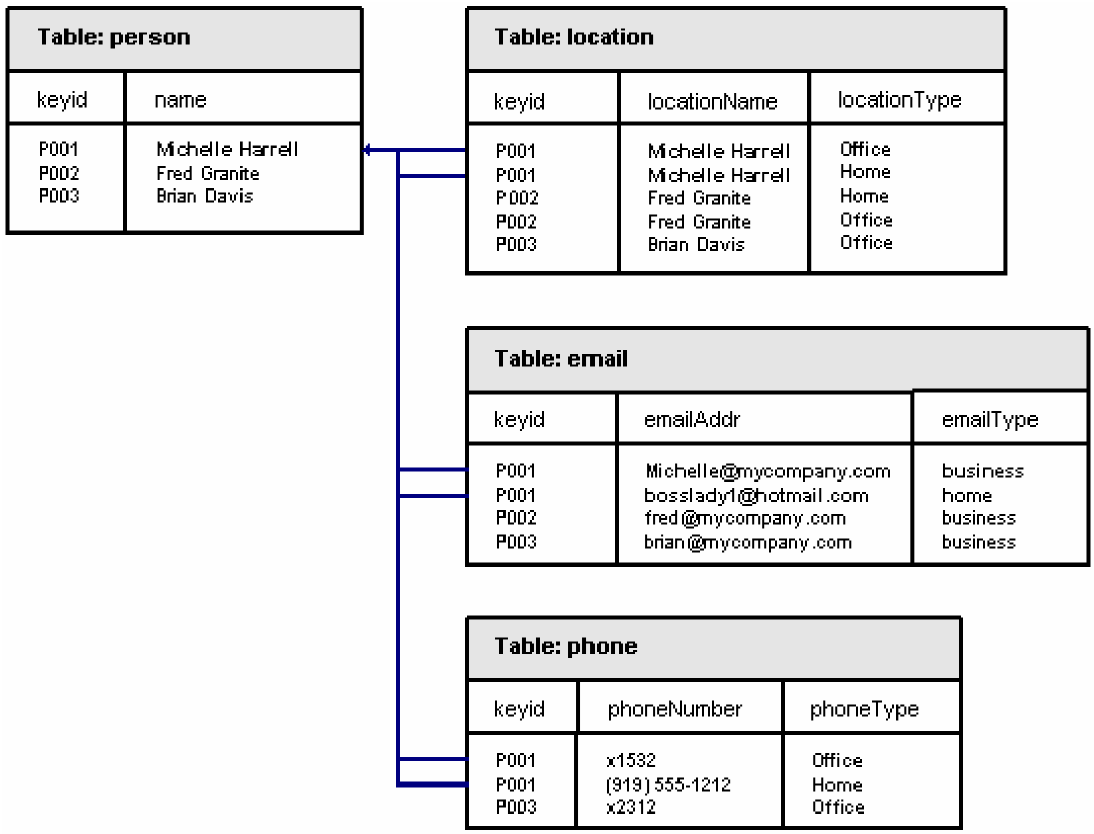


Table: person

keyid	name
P001	Michelle Harrell
P002	Fred Granite
P003	Brian Davis

Table: location

keyid	locationName	locationType
P001	Michelle Harrell	Office
P001	Michelle Harrell	Home
P002	Fred Granite	Home
P002	Fred Granite	Office
P003	Brian Davis	Office

Table: email

keyid	emailAddr	emailType
P001	Michelle@mycompany.com	business
P001	bosslady1@hotmail.com	home
P002	fred@mycompany.com	business
P003	brian@mycompany.com	business

Table: phone

keyid	phoneNumber	phoneType
P001	x1532	Office
P001	(919) 555-1212	Home
P003	x2312	Office

Note: Each person should only have one phone, e-mail, or address of a particular type. For example, if you have two "Office" e-mails, the second one will be overwritten by the update process. △

The following figure illustrates how information about Michelle Harrell's group memberships is stored. Data about group memberships is stored in the person, idgrps, and grpmems tables.

Figure A2.4 Relationships among the Data in the person, idgrps, and grpmems Tables

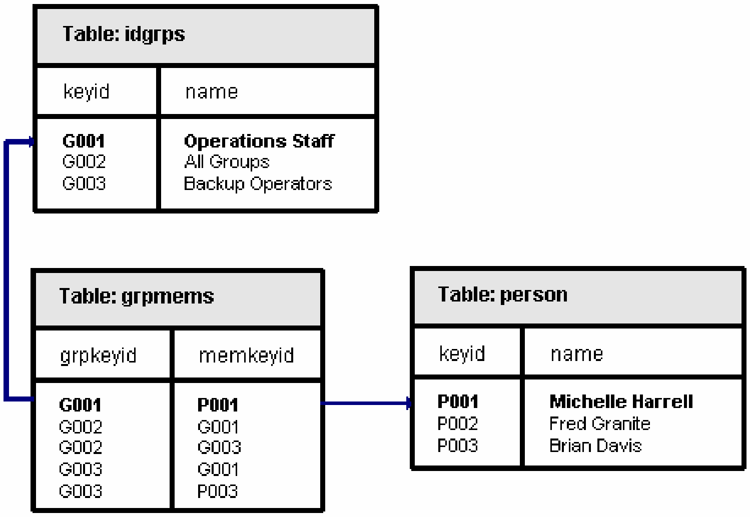


Table: idgrps

keyid	name
G001 G002 G003	**Operations Staff** All Groups Backup Operators

Table: grpmems

grpkeyid	memkeyid
G001 G002 G002 G003 G003	**P001** G001 G003 G001 P003

Table: person

keyid	name
P001 P002 P003	**Michelle Harrell** Fred Granite Brian Davis

The organization described by this data has the following three groups defined:

□ Operations Staff (G001)

□ All Groups (G002)

□ Backup Operators (G003)

"Michelle Harrell" is a member of the Operations Staff group. This relationship is represented in the first row of the grpmems table shown in Figure 2. The grpmems table maps the group's identifier (G001) to Michelle's key identifier (P001).

A group can also be a member of another group. In the second row of the grpmems table, we see that the Operations Staff group (G001) is a member of the All Groups group (G002). In addition, we see in the fourth row that the Operations Staff group (G001) is also a member of the Backup Operators group (G003). By virtue of the Operations Staff group's membership in the other groups, "Michelle Harrell" is an indirect member of the All Groups and Backup Operators groups.

Finally, the following figure illustrates the relationships between the data defined for "Michelle Harrell" in the person, logins, and authdomain tables. Person, Login, and AuthenticationDomain objects are the minimal set of objects required to establish a metadata identity on the SAS Metadata Server.

"Michelle Harrell" has two logins defined for her in the logins table. Both logins are associated with an authentication domain. Notice that the logins with user IDs "WinNet\Fred" and "WinNet\Brian" are not associated with an authentication domain. A login that is used by the metadata server to establish a user's identity is not required to have an authentication domain, because the user has already been authenticated on the metadata server. A login that will be read by an application to send to another system must have an authentication domain defined for it, because the information is needed to establish the connection. The first type of login is an inbound login; the second type of login is an outbound login. A single login can function as an inbound and an outbound login. In the figure, the login with user ID "WinNet\Michelle" is an example of a login that is both inbound and outbound.

Figure A2.5 Relationships among the Data in the person, logins, and authdomain Tables

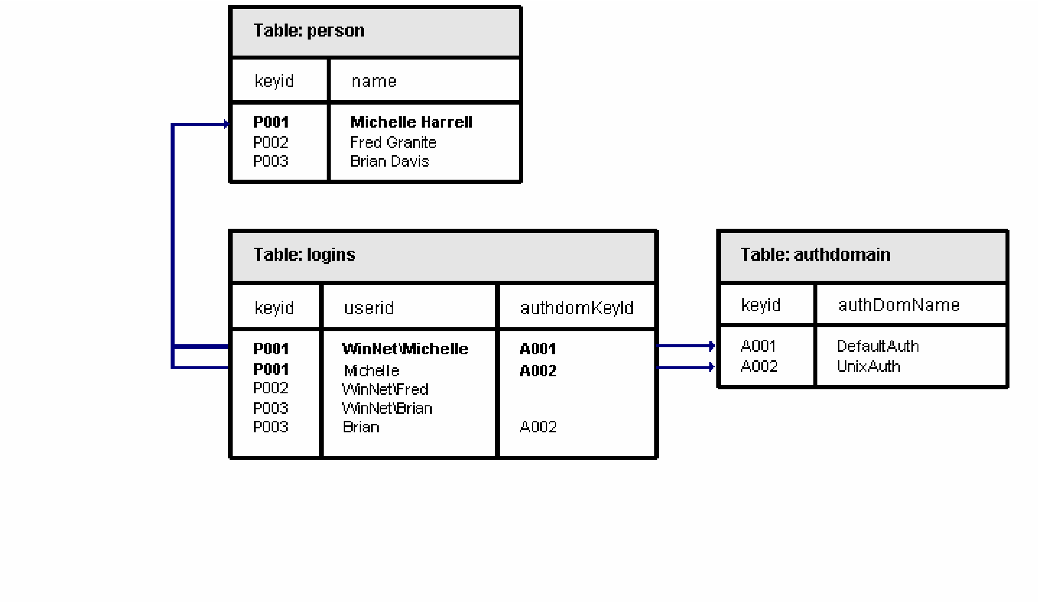


Table: person

keyid	name
P001	**Michelle Harrell**
P002	Fred Granite
P003	Brian Davis

Table: logins

keyid	userid	authdomKeyId
P001	**WinNet\Michelle**	**A001**
P001	Michelle	**A002**
P002	WinNet\Fred	
P003	WinNet\Brian	
P003	Brian	A002

Table: authdomain

keyid	authDomName
A001	DefaultAuth
A002	UnixAuth

When creating the person, logins, and authdomain tables, note the following:

- □ A person can have multiple logins defined in the logins table; however, a given login in the table can refer to only one person.
- □ A user ID used in one login can be used in other logins so long as all of the logins that use that user ID are owned by the same person.
- □ A user ID must be unique within an authentication domain. That is, there cannot be two logins with the same user ID in the same authentication domain.
- □ A person should have at least one login defined for him or her.

Bulk-Load Macros

Overview of Bulk-Load Macros

SAS provides a set of autocall macros for loading and maintaining user and group definitions that originate from an enterprise identity source. You can use these macros to create the tables and to import and periodically update the identity information on the metadata server. The macros create and maintain metadata objects of the following metadata types in a SAS Metadata Repository:

- □ Person
- □ IdentityGroup
- □ AuthenticationDomain
- □ Login
- □ Location
- □ Phone
- □ Email

For a description of each metadata type, see the alphabetical list of metadata types in the *SAS Open Metadata Interface: Reference* (available on SAS OnlineDoc 9.1.3).

Note: Only Person, Login, and AuthenticationDomain objects are required to establish a metadata identity on the metadata server. Group, location, e-mail, and telephone information is optional. △

%MDUIMPC Autocall Macro

The %MDUIMPC autocall macro creates canonical tables or views for importing user and group definitions to the metadata server from an external source. Here is the syntax for this macro:

```
%mduimpc ();
or
%mduimpc (libref=libref,
          maketable=0|1|2,
          infileref=fileref,
          fileheader=1|other_value);
```

libref

specifies the libref of the SAS library where the canonical tables or views should be created. If a libref is not specified, the default value is **Work**.

maketable

specifies whether to simply create macros and macro variables that enable you to create the canonical tables, or to create empty canonical tables or views.

0 (the default value) creates macro variables only.

1 creates empty canonical tables in addition to the macro variables.

2 creates canonical views of CSV files in the *infileref* directory in addition to the macro variables.

infileref

This parameter is used only when **maketable=2**. It specifies a SAS fileref that identifies CSV source files.

Note: The names of the CSV files in this location should match the table names listed in “Canonical Tables” on page 201. In UNIX host environments, the filenames must be lowercase. △

Note: The files should contain comma-separated values that are in the same order as the appropriate table in “Canonical Tables” on page 201. △

fileheaders

specifies whether the input files have a header line. This parameter is used only when **maketable=2**.

1 indicates that the input files have header information in the first row.

any other value indicates that the data begins in the first row.

The %MDUIMPC macro can be used in three ways:

- □ The default behavior of the macro is to define a set of macro variables that enable you to write a SAS program that creates the input tables necessary to create the objects.
- □ As an alternative, the macro supports options that enable you to create either of the following:

- □ empty tables of the correct format for which you can then write a separate program that appends user and group information
- □ DATA step views of CSV files that already contain information in the correct format

See "Table Relationships" on page 204 for information about how the macro joins the tables.

Submitting the %MDUIMPC macro without options is the same as specifying **maketable=0**.

When populating the input tables or defining the views, note that the specified user and group information should meet the integrity constraints imposed by the metadata server, as described in "Integrity Constraints" on page 186.

%MDUIMPL Autocall Macro

The %MDUIMPL autocall macro generates the XML necessary to load the tables created with the %MDUIMPC macro onto the metadata server and optionally executes it using PROC METADATA. Here is the syntax for this macro:

```
%mduimpl()
or
%mduimpl(libref=libref,
         temp=libref,
         outrequest=fileref,
         outresponse=fileref,
         submit=1|other_value,
         extidtag=context_value,
         mergeAuthDoms=1|other_value);
```

libref
specifies the libref of the SAS library that contains the input tables or views created with the %MDUIMPC macro. If this parameter is omitted, the default libref is **Work**.

temp
specifies the libref of a SAS library for storing temporary data sets created during %MDUIMPL processing. If this parameter is omitted, the default libref is **Work**.

outrequest
specifies a SAS fileref that identifies an optional file to save the generated XML. If you omit this parameter, the XML method calls are not saved to a file.

Note: The FILENAME statement assigning the fileref should specify a logical record length (LRECL) of 1024. △

outresponse
specifies a SAS fileref that identifies an optional file to store the output of the PROC METADATA request. If you omit this parameter, the PROC METADATA output is not saved to a file.

submit
specifies whether or not to submit the generated XML to the metadata server. The target metadata server and repository are determined from metadata system options.

1	(the default value) submits the generated XML to the metadata server.
any other value	does not submit the XML.

extidtag

specifies a value for the Context attribute of the ExternalIdentity object that is attached to all imported metadata to indicate its origin. The default value, if one is not specified, is **`IdentityImport`**. The value should not be quoted.

mergeAuthDoms

specifies whether or not login information should be associated with existing authentication domain information on the server or whether new AuthenticationDomain objects should be added.

`1` checks the SAS Metadata Server to see if the authentication domains in the input match any AuthenticationDomain objects that already exist on the server. If so, logins are associated to the existing AuthenticationDomain objects. If not, new AuthenticationDomain objects are created on the SAS Metadata Server.

any other value adds new AuthenticationDomain objects with unique names to the server.

Note: If **`mergeAuthDoms=0`**, and an authentication domain with the same name already exists, the load will fail. △

When executed using **`submit=1`** (the default value), the %MDUIMPL macro requires that system options be set to connect to the SAS Metadata Server. These system options are described in “Connection Options for the SAS Metadata Server” on page 214.

If you specify a fileref for OUTREQUEST and specify a value other than **`1`** for SUBMIT, then you can load the XML at a later date by using PROC METADATA directly. For more information, see the documentation for the METADATA procedure in the *SAS Open Metadata Interface: Reference*.

%MDUEXTR Autocall Macro

The %MDUEXTR autocall macro extracts identity metadata from the metadata server and writes it into canonical tables. Here is the syntax for this macro:

```
%mduextr (libref=SAS-library);
```

libref

specifies the libref of the SAS library where the canonical tables will be stored.

You need to set SAS system options for the %MDUEXTR macro in order to connect to the metadata server and to identify the repository from which to read identity metadata. These options are described in “Connection Options for the SAS Metadata Server” on page 214.

The macro uses PROC METADATA to submit extraction requests to the metadata server.

It writes all identity metadata found in the foundation repository into the canonical tables in the specified library.

%MDUCMP Autocall Macro

The %MDUCMP autocall macro compares the master canonical tables to the target canonical tables and creates change tables that list the changes that must be made to the target user information so that it will match the master.

Note: This macro does not attempt to validate the information in the change tables. To ensure that the new information from the enterprise data source meets server-enforced integrity constraints, run the %MDUCHGV macro on the change data sets before loading the changes on the metadata server. △

Note: By default, identities that were created manually in SAS Management Console are excluded from the synchronization process. However, manual updates that have been made to login, location, telephone number, and e-mail information that was initially imported will be overwritten by the synchronization process, unless you define exceptions to that process. See “Defining an Exception to the Comparison Process” on page 198 for steps you can take to preserve manual updates to imported information. △

Here is the syntax for this macro:

```
%mducmp (master=libref,
         target=libref,
         change=libref,
         exceptions=<libref.>dataset,
         externonly=0|1
         authdomcompare=name|keyid);
```

master

specifies the libref of the SAS library containing the master canonical tables. When you are updating imported identities on the metadata server with current information from the enterprise source, specify the library that contains information extracted from the enterprise identity source.

target

specifies the libref of the SAS library containing the target canonical tables. When you are updating imported identities on the metadata server with current information from the enterprise source, specify the library that contains information extracted from the SAS Metadata Server.

change

specifies the libref of the SAS library in which to create the change tables. Six tables are created, where *xxx* is the base name of the canonical table.

*xxx*_add contains user and group definitions that need to be added to the target tables to make them look like the master tables.

*xxx*_update contains user and group definitions that need to be modified in the target tables to make them look like the master tables.

*xxx*_delete contains user and group definitions that need to be deleted from the target tables to make them look like the master tables.

person_summary summarizes changes to Person objects.

idgrps_summary summarizes changes to IdentityGroup objects.

authdomain_summary summarizes changes to AuthenticationDomain objects.

exceptions

specifies a data set that contains exception values. Instructions for creating an exceptions data set are provided in “Defining an Exception to the Comparison Process” on page 198.

externonly

when the master data set includes an ObjectId column, this option determines whether identities that were manually created in SAS Management Console are included in the comparison.

Note: In most circumstances, the master data set does not include an ObjectId column, so this option has no effect. However, a master data set that is extracted from the SAS metadata repository (rather than from an external enterprise identity source) will include an ObjectId column. △

A value of **1** ensures that only identities that have an associated ExternalIdentity object (imported identities) are included in the comparison. This is the default value. A value of **0** causes both imported identities and identities that were defined in SAS Management Console to be included in the comparison.

authdomcompare

specifies how authentication domain information should be compared. The default value, **name**, specifies to compare all authentication domains in the master and target canonical tables by name. That is, if an authentication domain is found that has a matching name in both the master and target canonical tables, then that domain is left alone (no updates are made). If a domain name is found in the master tables that does not also exist in the target tables, then the domain is marked for addition to the target. If a domain name is found in the target tables that does not also exist in the master tables, the domain is left alone. It is assumed that the authentication domain was added to the target in SAS Management Console and is meant to be there.

Note: When **EXTERNONLY=1** and **AUTHDOMCOMPARE=name**, **AUTHDOMCOMPARE=name** overrides **EXTERNONLY=1** for authentication domains: the %MDUCMP macro compares all authentication domains whether or not the domains have an ExternalIdentity object. △

The value **keyid** specifies to compare authentication domains that have a matching ExternalIdentity identifier in the master and target tables. For each domain that has an ExternalIdentity object, the domain is compared and updated if a matching key identifier exists in the master and target canonical tables. If a key identifier exists in the master tables that is not in the target tables, the domain is marked for addition to the target source. If a key identifier exists in the target tables that is not the master tables, the domain is marked for deletion from the target source. When **AUTHDOMCOMPARE=keyid**, the %MDUCMP macro compares only authentication domains that have an ExternalIdentity, regardless of the EXTERNONLY setting. Use this option if you wish to be able to update the authentication domain name or delete an imported authentication domain.

%MDUCHGV Autocall Macro

The %MDUCHGV autocall macro checks the change tables created by the %MDUCMP macro against information extracted from the metadata server to see if any of the changes violate server-enforced integrity constraints. Here is the syntax for this macro:

```
%mducmp (target=libref,
         change=libref,
         temp=libref,
         errorsds=name);
```

target

specifies the libref of the SAS library containing the target canonical tables. When updating imported identities on the metadata server with current information from the enterprise source, this library contains information extracted from the SAS Metadata Server.

change

specifies the libref of the SAS library containing the change tables created by the %MDUCMP macro.

temp

specifies the libref of a SAS library where temporary tables can be written. If this option is omitted, the macro uses the libref **Work**.

errorsds
specifies the name of the data set where integrity errors are logged. ERRORSDS stands for "error data set." The data set has the following columns:

errcode specifies a numeric code which corresponds to a particular error.

errmsg specifies a text message that describes the error that was detected.

tablename specifies the name of the canonical table from which the 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 will be 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, the %MDUCHGV macro sets the DUCHGV_ERRORS column to a non-zero value and creates the ERRORSDS data set, which lists the errors that were found. See “4. Validating the Change Tables” on page 198 for information about how the information in the ERRORSDS data set can be used.

%MDUCHGL Autocall Macro

The %MDUCHGL autocall macro generates the XML necessary to load identity metadata updates onto the metadata server and optionally submits it using PROC METADATA. Here is the syntax for this macro:

```
%mduchgl (change=libref,
          temp=libref,
          outrequest=fileref,
          outresponse=fileref,
          submit=1|other_value,
          extidtag=context_value);
```

change
specifies the libref of the SAS library containing the change tables created by the %MDUCMP macro.

temp
specifies the libref of a SAS library where temporary tables can be written. If this option is omitted, the macro uses the libref **Work**.

outrequest
specifies a SAS fileref that identifies an optional file to save the generated XML. If you omit this parameter, the XML method calls are written to a temporary file.

Note: The FILENAME statement assigning the fileref should specify a logical record length (LRECL) of 1024. △

outresponse
specifies a SAS fileref that identifies an optional file to store the output of the PROC METADATA request. If you omit this parameter, the PROC METADATA output is not saved to a file.

submit
specifies whether or not to submit the generated XML to the metadata server. The target metadata server and repository are determined from metadata system options.

`1`	(the default value) submits the generated XML to the metadata server.
any other value	does not submit the XML.

extidtag
specifies a value for the Context attribute of the ExternalIdentity object that is associated with metadata added by this macro. The default value is **`IdentityImport`**. The value should not be quoted.

You need to set SAS system options for the %MDUCHGL macro in order to connect to the metadata server and to identify the repository in which to write changes. These options are described in “Connection Options for the SAS Metadata Server” on page 214.

Connection Options for the SAS Metadata Server

Macros that load, extract, and change metadata (%MDUIMPL, %MDUEXTR, and %MDUCHGL, respectively) on the metadata server must connect to the SAS Metadata Server in order to submit their requests. You can connect to a metadata server by setting the following SAS system options in your SAS program:

```
options metaserver=server_name
        metaport=port_number
        metauser="userid"
        metapass="password"
        metaprotocol=bridge
        metarepository=Foundation;
```

- □ For METASERVER, specify the host name or TCP address of the computer hosting a SAS Metadata Server.
- □ For METAPORT, specify the unique port number where the metadata server is started and listens for requests. The default port for a metadata server is 8561.
- □ For METAUSER, specify a valid user ID for the task that you want to perform. In order to create new user accounts on a metadata server, you must connect to the server using a user ID that has at least *administrative user* status on the server. In order to extract and update existing user accounts on the metadata server, connect to the metadata server using a user ID that has *unrestricted user* status.
- □ For METAPASS, specify the password associated with the specified user ID.
- □ For METAPROTOCOL, specify **`bridge`**. This is the connection protocol required by the metadata server.
- □ For METAREPOSITORY, specify **`Foundation`**. This is the name of the repository for storing global metadata and user definitions.

APPENDIX 3

Security Implementation Example

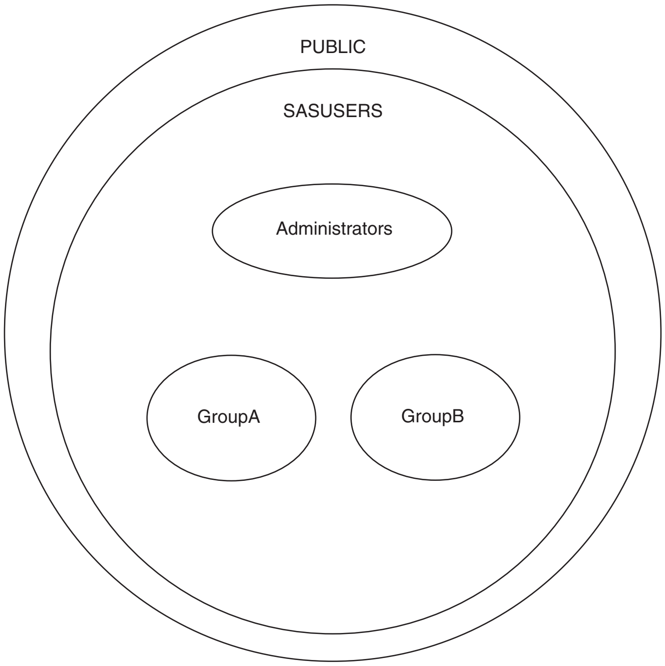
Goals and Configuration

This simplified case study demonstrates how you can implement mutually exclusive access control in an environment that has multiple authentication providers. These are the security goals for this example:

- Establish mutually exclusive access controls for the data in two SAS libraries so that they meet the following requirements:
    - One set of users has exclusive access to the SAS data in LibraryA.
    - Another set of users has exclusive access to the SAS data in LibraryB.
    - The Administrators group has ReadMetadata access to both libraries.
- Enable only the users who access data in LibraryA to access the third-party database server.
- Give only members of GroupA, GroupB, and Administrators default write access to the repository.

The following figure depicts the group structure for this example:

Figure A3.1 Group Structure



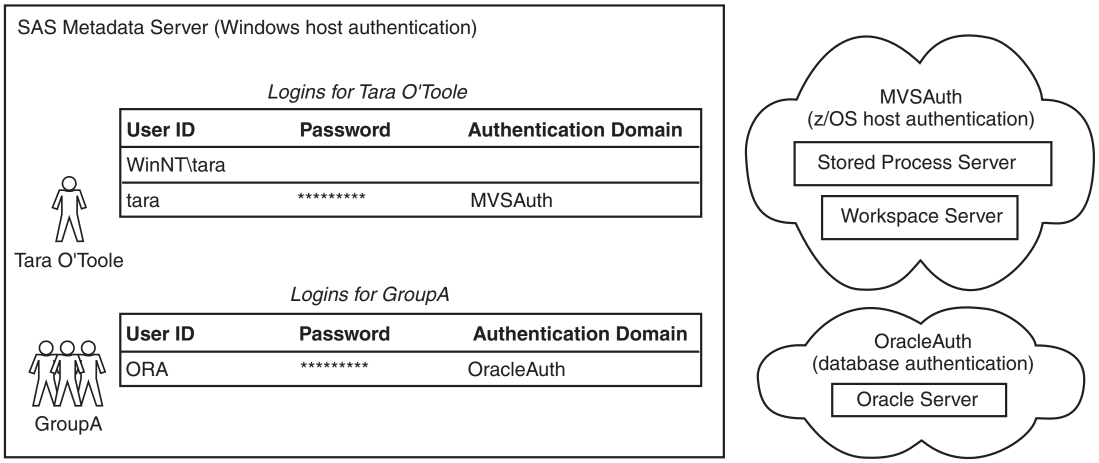
In this example, you have a diverse environment in which more than one authentication process is being used. The following figure depicts these details of the environment:

□ the servers and authentication domains for this example

□ the logins for Tara O'Toole, which provide an example of the metadata identities that you will create in this example

□ the shared login that you will define to enable GroupA to access the Oracle Server

Figure A3.2 Servers, Authentication Domains, and Logins


SAS Metadata Server (Windows host authentication)
Tara O'Toole
GroupA
MVSAuth
(z/OS host authentication)
Stored Process Server
Workspace Server
OracleAuth
(database authentication)
Oracle Server

Logins for Tara O'Toole

User ID	Password	Authentication Domain
WinNT\tara		
tara	*********	MVSAuth

Logins for GroupA

User ID	Password	Authentication Domain
ORA	*********	OracleAuth

Example: Implementing Security

To set up security for regular users in this example, complete these steps:

1 Log on to SAS Management Console by opening a metadata profile with your *administrative user* account.

2 Use User Manager to create a group definition for GroupA.

   a On the **General** tab, enter **GroupA** as the group name.
   b On the **Logins** tab, add a login for the database server. This login should specify the user ID (ORA) and the password for a shared account that you have created on the database server. From the **Authentication Domain** drop-down list, select **OracleAuth** to associate the login with the authentication domain in which the Oracle server is defined.
   c Click **OK** to save and close the group definition.

   Note: No other groups are members of GroupA, so you do not need to add any members on the **Members** tab. △

3 Use User Manager to create a group definition for GroupB.

   a On the **General** tab, enter **GroupB** as the group name.
   b Do not add any logins on the **Logins** tab. Members of GroupB do not access the database server.
   c Click **OK** to save and close the group definition.

   Note: No other groups are members of GroupB, so you do not need to add any members on the **Members** tab. △

4 On the **Authorization** tab for each group, secure the group definition by directly assigning access controls that deny WriteMetadata permission to PUBLIC and grant WriteMetadata permission to the Administrators group. For detailed instructions, see "Setting Explicit Protections for Security-Related Resources" on page 71.

   Note: Directly assigned permissions do not have a gray background color. If the background color for a permission on the **Authorization** tab for a group definition is gray, that permission comes from the repository ACT. A permission from the repository ACT is not sufficient to protect a group definition, because you will expand WriteMetadata access to the repository as your security implementation progresses. △

5 Define each group's default access to the entire repository. On the **Users and Permissions** tab of the Default ACT, set these access controls:

   □ Leave the permissions for PUBLIC as they are (all permissions are denied).

   □ Grant Read and ReadMetadata permissions to the SASUSERS group. By default all users who have a metadata identity will have read access to resources.

   □ Leave the permissions for the Administrators group as they are (ReadMetadata, WriteMetadata, and Administer permissions are granted).

   □ Grant Write, Create, Delete, and WriteMetadata permissions to GroupA and GroupB. By default, members of these groups will be able to make changes to most metadata objects and to the data that those objects represent.

   Note: It is not necessary to give GroupA and GroupB the Read and ReadMetadata permissions. Members of GroupA and GroupB are automatically members of SASUSERS, and SASUSERS has Read and ReadMetadata permissions. △

6 Modify the default access controls for selected resources in accordance with the security goals. The permissions that you set in the previous step are the default permissions for all resources. In this example, you will set additional controls for LibraryA and LibraryB so that only GroupA can access LibraryA and only GroupB can access LibraryB.

- □ On the **`Authorization`** tab for LibraryA, set these directly assigned permissions:
  - □ Deny Read, ReadMetadata, Create, Write, Delete, and WriteMetadata permissions to PUBLIC.

    Note: The directly assigned (no background color) denial to PUBLIC will prevent the Administrators group and GroupB from accessing the data in LibraryA. However, this denial is not displayed in the permissions lists for the Administrators group and GroupB. △
  - □ Grant Read, ReadMetadata, Write, Create, Delete, and WriteMetadata permissions to GroupA.

    Note: For members of GroupA, these directly assigned grants to the user-defined group (GroupA) will override the directly assigned denials to the implicit group (PUBLIC). △
  - □ Grant ReadMetadata and WriteMetadata permissions to Administrators.

    Note: For members of Administrators, these directly assigned grants to the user-defined group (Administrators) will override the directly assigned denials to the implicit group (PUBLIC). △
- □ On the **`Authorization`** tab for LibraryB, set these permissions:
  - □ Deny Read, ReadMetadata, Write, Create, Delete, and WriteMetadata permissions to PUBLIC.
  - □ Grant Read, ReadMetadata, Write, Create, Delete, and WriteMetadata permissions to GroupB.
  - □ Grant ReadMetadata and WriteMetadata permissions to Administrators.

Note: The permissions that you set on LibraryA and LibraryB will be inherited by the tables within each library. △

7 Create the necessary user accounts. Each user will need these accounts:

- □ a Windows account that enables the user to get to the metadata server as described in initial authentication
- □ a z/OS account that enables the user to get to the stored process and workspace servers in the MVSAuth authentication domain

8 Create a user definition for every user whose access needs cannot be met through membership in the PUBLIC group. In this example, you did not give the PUBLIC group any access to the repository, so all users must have a metadata identity in order to access any resources. For each user definition, complete these tasks:

a On the **`General`** tab, enter the user's name in the **`Name`** field. The other fields on the **`General`** tab are optional.

b On the **`Groups`** tab, define the user's group memberships. In this example, all users are automatically members of both PUBLIC and SASUSERS. Add selected users to either GroupA or GroupB.

c On the **`Logins`** tab for each user definition, complete these tasks:

- □ Add a login to enable the metadata server to determine the user's metadata identity. If this login is used only for the purpose of determining the user's metadata identity, then this login does not have to include a password or specify an authentication domain. In the figure, the login that is used to determine Tara's metadata identity consists of only her user ID (WinNT\tara).

- □ Add another login so the user can access the workspace server and stored process server in the MVSAuth authentication domain. As depicted in the previous figure, this login should include a password and be associated with the MVSAuth authentication domain. This login functions only as an outbound login. For example, the login that enables Tara to access the workspace server and the stored process server consists of her z/OS user ID (tara) and a password. This login is associated with the MVSAuth authentication domain.

Note: Do not give any individual users who are members of GroupA a login for the OracleAuth authentication domain. Members of GroupA will access the third-party database server using the login that you added on the **`Logins`** tab for GroupA. △

d Click **`OK`** to save and close the user definition.

Note: It is not necessary to set any permissions on the **`Authorization`** tab of the user definition. By default, only *administrative users*, *unrestricted users*, and the user who is represented by a particular user definition can make changes to that user definition. △

If you want to protect multiple resources (rather than protecting just one library for each user group), you should use an access control template. This enables you to define each identity/permission pattern only once and then apply each pattern to multiple resources. To establish mutually exclusive security, you would create two ACTs and apply one of the ACTs to each resource that is accessed exclusively by either GroupA or GroupB.

APPENDIX 4

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- □ *SAS Intelligence Platform: Desktop Application Administration Guide*
- □ *SAS Intelligence Platform: Overview*
- □ *SAS Intelligence Platform: System Administration Guide*
- □ *SAS Intelligence Platform: Web Application Administration Guide*

For a complete list of administration documentation for the SAS Intelligence Platform, see **`http://support.sas.com/913administration.`**

For a list of SAS documentation, see
`http://support.sas.com/documentation/onlinedoc/sas9doc.html.`

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **`sasbook@sas.com`**
Web address: **`support.sas.com/pubs`**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

access control entry (ACE)
a set of identities and permissions that are directly associated with a particular resource. Each access control entry is directly associated with only one resource. More than one ACE can be associated with each resource.

access control template (ACT)
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.

administrative user
a special user of a metadata server who can create and delete user definitions and logins. An administrative user can also perform tasks such as starting, stopping, pausing, and refreshing the metadata server. You are an administrative user if your user ID is listed in the adminUsers.txt file or if you connect to the metadata server using the same user ID that was used to start the metadata server.

authentication
the process of verifying the identity of a person or process within the guidelines of a specific authorization policy.

authentication domain
a set of computing resources that use the same authentication process. An individual uses the same user ID and password for all of the resources in a particular authentication domain. Authentication domains provide logical groupings for resources and logins in a metadata repository.

authentication provider
a software component that is used for identifying and authenticating users.

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.

credentials
the user ID and password for a particular user account that has been established either in the operating system or with an alternative authentication provider.

encryption
the act or process of converting data to a form that only the intended recipient can read or use.

inbound login
a login that is used to determine your metadata identity. The login is inbound to a SAS Metadata Server. A login that functions only as an inbound login does not need to include a password or to specify an authentication domain.

login
a combination of a user ID, a password, and an authentication domain. Each login provides access to a particular set of computing resources. See also inbound login, outbound login.

metadata identity
a metadata object that represents an individual user or a group of users in a SAS metadata environment.

member-level permission condition
a control that defines access to data at a low level, specifying who can access particular members within an OLAP cube. Member-level 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.

outbound login
a login that applications can retrieve from a SAS Metadata Server and send to other systems that need to verify a user's identity. The login is outbound from the SAS Metadata Server to the other systems. An outbound login must specify an authentication domain and must include credentials that are appropriate for the systems to which the login provides access.

permission condition
a constraint on the explicitly granted permissions for a particular resource. You can use a permission condition to grant access to a specific portion, or slice, of data within a resource. For example, if an OLAP cube has an EmployeeInfo dimension that includes a Salary level, you could give a particular user access to data for only those employees who have salaries that are less than $50,000 per year.

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.

row-level permission condition
a control that defines access to data at a low level, specifying who can access particular rows within a table. Row-level controls are typically used to subset data by a user characteristic such as employee ID or organizational unit. For example, 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.

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.

trust relationship

a logical association through which one component of an application accepts verification that has already been performed by another component. For example, the establishment of a trust relationship enables users to log on to the application once, and then to access all associated resources without the need for re-authentication. See also trusted user.

trusted user

a special user of a metadata server who can acquire credentials on behalf of other users in a multi-tier server environment.

unrestricted user

a special user of a metadata server who can access all metadata on the server (except for passwords, which an unrestricted user can overwrite but cannot read). An unrestricted user can also perform tasks such as starting, stopping, pausing, and refreshing the metadata server. You are an unrestricted user if your user ID is listed in the adminUsers.txt file and is preceded by an asterisk.

Index

Your Turn

We want 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**.