Chapter 1
Concepts

About This Document

This document helps you administer users and permissions in SAS Management Console. It explains key concepts and provides step-by-step instructions for selected tasks. For more information about security, see the *SAS Intelligence Platform: Security Administration Guide*.

Introduction to User Administration

About User Administration

In order to make access distinctions and track user activity, security systems must know who is making each request. The primary purpose of user administration is to provide information that helps systems make this determination. The central piece of user information that the SAS environment requires is one external account ID for each user. The SAS environment uses its copy of these IDs to establish a unique SAS identity for each connecting user. All of a user's group memberships, role memberships, and permission assignments are ultimately tied to their SAS identity.

Note: For identification purposes, only the account IDs are needed. SAS doesn't maintain copies of external passwords for identification purposes.

To access user administration features in SAS Management Console, select the **User Manager** node on the **Plug-ins** tab. Your roles and permissions determine which user administration tasks you can perform.

Note: Don't confuse the **Users** folder (on the **Folders** tab) with the **User Manager** node (on the **Plug-ins** tab). The **User Manager** node is the only location from which you can manage identities. The **Users** folder provides containers in which users can store their content.

TIP As an alternative to interactively creating and maintaining identity information, you can write a program that performs these tasks as batch processes. See the user import macros documentation in the *SAS Intelligence Platform: Security Administration Guide*.

About Users

A user is an individual person or service identity.

We recommend that you create an individual SAS identity for each person who uses the SAS environment. This enables you to make access distinctions in the metadata layer and establishes a personal folder for each user. If generic access is sufficient for some of your users, those users can instead share the generic PUBLIC group identity.

An individual SAS identity is established by coordination between two sets of identity information:

- in an external system, a user account
- in the metadata, a user definition that includes a copy of the external account ID

To give someone an individual SAS identity, you create a metadata user definition that includes a copy of their external account ID. This list provides details for several configurations:

- In the simplest configuration, each user needs an account that is known to the metadata server's host.
  - If the metadata server is on Windows, users typically have Active Directory accounts.
  - If the metadata server is on UNIX, users might have UNIX accounts. Sometimes a UNIX host recognizes LDAP, Active Directory, or other types of accounts.
- In a common alternate configuration, the metadata server trusts authentication that is performed at the Web perimeter. In this configuration, anyone who uses a Web application needs a Web realm account.
- In a less common alternate configuration, the metadata server directly uses an LDAP provider such as Active Directory. This is appropriate only if you have accounts that aren't already accepted by the metadata server's host. For example, if the metadata server is on Windows, it isn't necessary (or appropriate) to configure direct use of Active Directory.

Note: A PUBLIC-only user doesn't need a metadata user definition (but does need an account). For metadata administrators and some service identities, it is appropriate to use a SAS internal account.

About Groups

A group is a set of users.

We recommend that you create groups to simplify security management as follows:

- It is more efficient to assign permissions to groups than to individual users.
- If you need to store passwords in the metadata, you can reduce the amount of required maintenance by using a group to make one shared account available to multiple users.
- It is sometimes more efficient to manage role membership by assigning groups to roles instead of assigning users directly to roles.

This table introduces three predefined groups:

Table 1.1 *PUBLIC, SASUSERS, and SAS Administrators*

Group	Description
PUBLIC	Includes everyone who can access the metadata server (directly or through a trust relationship).
SASUSERS	Includes those members of the PUBLIC group who have a well-formed user definition.
SAS Administrators	Should include only users who perform metadata administrative tasks. In a standard configuration, members are granted broad access but aren't unrestricted.

TIP A group's membership can include other groups as well as individual users. This enables you to create a nested group structure.

About Roles

A role manages the availability of application features such as menu items.

An application feature that is under role-based management is called a capability. Anyone who is a member of a role has all of that role's capabilities. This list highlights key points:

- Roles determine which user interface elements (such as menu items or plug-ins) you see when you use an application. Roles don't protect data or metadata (other than a few system items).
- Having a certain capability is not an alternative to meeting permission requirements. Permission requirements and capability requirements are cumulative.
- Roles and groups serve distinct purposes. You can't assign permissions to a role or capabilities to a group.
- Capabilities are always additive. Assigning someone to a role never reduces what that person can do.

Each application that supports roles offers a fixed set of capabilities. You can't convert an application feature that is not a capability into a capability. However, if you add custom plug-ins (in SAS Management Console) or custom tasks (in SAS Enterprise Guide or the SAS Add-In for Microsoft Office) you can register those features as capabilities.

Each application that supports roles provides one or more predefined roles. Each predefined role has a unique initial set of capabilities. The capabilities that a role provides should reflect the activities and responsibilities of that role's members. You can adjust the distribution of capabilities in these ways:

- Change role memberships. For example, to prevent regular users from seeing plug-ins in SAS Management Console, you might narrow the membership of the **Management Console: Content Management** role by making changes on that role's **Members** tab.
- Customize the initial roles-to-capabilities mapping by using any of these techniques:
  - Incrementally select or clear explicit capabilities for a role. You can't deselect capabilities for the unrestricted role.
  - Aggregate existing roles so that one or more roles contributes all of their capabilities to another role.
  - Create new roles that provide unique combinations of capabilities.

This table introduces the main administrative roles:

Table 1.2 *Main Administrative Roles*

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

* Unrestricted users are subject to denials in other authorization layers, can use only those logins that are assigned to them (or to groups to which they belong), and don't have implicit capabilities that are provided by components other than the metadata server.

** Restricted user administrators can't update identities for which they have an explicit ☑ or ACT ✅ (green) denial of WriteMetadata.

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

About Logins

What is a Login?

A login is a SAS copy of information about an external account. Every login must include a user ID. In a login for a Windows account, the ID must be qualified (for example, *userID@company.com)*, *domain\userID*, or *machine\userID*.

TIP The requirement to provide a qualified ID for a Windows account applies to the SAS copy of the ID. It is usually not necessary to qualify the user ID that you provide when you launch a SAS application.

TIP If you do provide a qualified ID when you log on, you must use the same format that was used in your login. For example, Windows might accept both *WIN\me* and *Me.MyLastName@mycompany.com*, but SAS can understand only one of these qualified form (the form in which the SAS copy of the ID is stored).

Logins for Users

Each user should have a login that establishes their SAS identity. It is not necessary to include a password in this login. However, the password column always displays eight asterisks (regardless of whether a password is actually stored). For example, this is how Joe's login might look when a user administrator views Joe's **Accounts** tab:

```
DefaultAuth | WIN\Joe  | ********
```

A user might have additional logins that provide access to other systems. For example, if Joe has his own Oracle account, he might have these two logins:

```
DefaultAuth | joe      | ********

OracleAuth  | ORAjoe   | ********
```

Note: The Oracle login should include a copy of Joe's Oracle password.

If a site uses Web authentication, the requirements are different. For example, if Joe uses both Web and desktop applications at such a site, Joe might have these three logins:

```
DefaultAuth | WIN\Joe | ********

OracleAuth  | ORAjoe  | ********

web         | WEBjoe  | ********
```

Note: Like his DefaultAuth login, Joe's Web login is used only to launch clients, so there is no need to create a SAS copy of Joe's Web realm password.

Logins for Groups

Groups don't have to have logins. The main reason to give a login to a group is to make a shared account available to multiple users. A group login contains a SAS copy of the user ID and password for a shared account. For example, to provide shared access to DB2, a group might have a login that looks like this:

```
DB2Auth | sharedDB2id | ********
```

All members of the group can use this login. Since this login is for a third-party database, a copy of the DBMS account password should be stored in this login.

About Internal Accounts

What is an Internal Account?

An internal account is a SAS account that the metadata server authenticates independently, without relying on an external authentication provider such as the operating system. Use internal accounts only for administrators and some service identities. For these purposes, an internal account is an acceptable substitute for an external account with a corresponding login. For example, the SAS Administrator and the SAS Trusted User can be based on internal accounts.

Benefits of Internal Accounts

Internal accounts have these advantages:

- Internal accounts provide an alternative to creating external accounts for SAS internal purposes such as inter-process communication.
- Internal accounts can be maintenance free. You don't have to synchronize internal accounts with some other user registry. Internal accounts don't have to conform to the security policies of the rest of your computing environment. For example, even if your host security policy forces password changes every 30 days, you can retain the initial policy for internal account passwords (which is that these passwords never expire).
- Internal accounts are usable only in the SAS realm, so they reduce exposure to the rest of your security environment.

TIP You can also use an internal account to temporarily assume another user's identity for validation or troubleshooting purposes.

Limitations of Internal Accounts

Although the **Create Internal Account** button is available on all user definitions, internal accounts are not intended for regular users. Someone who has only an internal account can't do these things:

- launch a standard workspace server without interactively providing some external credentials
- participate in Integrated Windows authentication or Web authentication
- add, delete, initialize, or unregister a foundation repository

Policies for Internal Accounts

By initial policy, these server-level settings are in effect:

- Accounts don't expire and aren't suspended due to inactivity.
- Passwords must be at least six characters, don't have to include mixed case or numbers, and don't expire.
- The five most recent passwords for an account can't be reused for that account.
- There is no mandatory time delay between password changes.
- After three failed attempts to log on, an account is locked. If an account is locked because of log on failures, further log on attempts cannot be made for one hour.
- For an account that has a password expiration period, there is a forced password change on first use and after the password is reset by someone other than the account owner. By initial policy, passwords don't expire so there are no forced password changes.

Note: These settings are defined in the metadata server's omaconfig.xml file. In User Manager, you can customize some of these settings on a per-account basis.

CAUTION:
Passwords for a few required accounts (such as the SAS Administrator and the SAS Trusted User) are included in configuration files. If you change these passwords, you must also update the configuration. See the *SAS Intelligence Platform: Security Administration Guide*.

About Authentication Domains

What is an Authentication Domain?

An authentication domain is a name that facilitates the matching of logins with the servers for which they are valid. This matching is not important when you launch a client, but it is important when you access certain secondary servers such as a third-party DBMS or, in some configurations, a standard workspace server.

When Do I Need to Add an Authentication Domain?

In the simplest case, all logins and SAS servers are associated with one authentication domain (DefaultAuth). This list describes the most common reasons for using more authentication domains:

- If you use Web authentication, you might need a second authentication domain for the logins that contain Web realm user IDs.
- If you have a third-party server (such as a DBMS server) that has its own user registry, you need a separate authentication domain for that server and its logins.
- If both of the following criteria are met, you need a separate authentication domain for the standard workspace server and its logins:
  - The standard workspace server doesn't share an authentication provider with the metadata server (and can't be configured to do so).
  - You want to provide seamless individualized access to the standard workspace server.

About Passwords

Passwords in Logins

It is usually not necessary to create a SAS copy of an external password. The main reason to include a password in a login is to provide seamless access to a server that requires credentials that are different from the credentials that users initially submit. These are the most common examples:

- A third-party DBMS server usually requires a different set of credentials.
- In a multi-platform environment, the standard workspace server might require a different set of credentials.

If credentials aren't otherwise available, most applications prompt users for an appropriate user ID and password.

Passwords in Internal Accounts

Internal accounts exist only in the metadata. Each internal account includes a password. By initial policy, internal passwords don't expire.

Passwords in Configuration Files

Passwords for a few required accounts (such as the SAS Administrator and the SAS Trusted User) are included in configuration files. If you need to change these passwords, see the *SAS Intelligence Platform: Security Administration Guide*.

About External Identities

What is an External Identity?

While logins and internal accounts are involved in the log on process, external identities are not. An external identity is an optional synchronization key for a user, group, or role. If you use batch processes to coordinate SAS identity information with your primary user registry, you need external identities (such as employee IDs) to facilitate matching. This list explains the circumstances in which a user, group, or role needs an external identity:

- For a user, group, or role that you maintain interactively in SAS Management Console, no external identity is needed.
- For a user, group, or role that you maintain using batch processes, one external identity is needed.

Where do External Identities Come From?

External identities can be added in these ways:

- For a user, group, or role that is created by an import process, an external identity is added as part of that process.
- For any user, group, or role, you can interactively add an external identity on the **General** tab of their definition.

Requirement: Unique Names and IDs

Within a metadata server, these uniqueness requirements apply:

- You can't create a user definition that has the same name as an existing user definition.
- You can't create a group or role definition that has the same name as an existing group or role definition.
- You can't assign the same user ID to different users or groups. All of the logins that include a particular user ID must be owned by the same identity. This enables the metadata server to resolve each user ID to a single identity.
  - This requirement is case-insensitive. For example, you can't 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 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.
  - This requirement can't 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* in DefaultAuth, you can't give any other user a login with the user ID *smith*, even if you put that login in another authentication domain.
- If you give a user two logins that contain the same user ID, the logins must be in different authentication domains. Within an authentication domain, each user ID must be unique. For example, if you give Tara O'Toole two logins that both have a user ID of *tara*, then you can't associate both of those logins with the OraAuth authentication domain. As with the previous requirement, this requirement is case-insensitive and is applied to the fully qualified form of the user ID.

Introduction to Access Management

About Access Management

Access management determines which items a user can interact with. The permissions that you set in SAS Management Console are part of a metadata-based access control system that SAS provides. These permission settings supplement protections in other layers (such as the operating system and the WebDAV). Across layers, protections are cumulative. You can't perform a task unless you have sufficient access in all layers.

CAUTION:

> **Do not rely exclusively on metadata layer permissions to protect data.** Manage physical access (operating system and DBMS permissions) in addition to metadata layer access.

You manage access to an item as part of the item's properties (on the item's **Authorization** tab). Your roles and permissions determine which access management tasks you can perform.

Granularity and Mechanics of Permissions

Repository-Level Controls

Repository-level controls function as a gateway. Participating users usually need ReadMetadata and WriteMetadata permissions for the foundation repository. Repository-level controls also serve as a parent-of-last-resort, defining access to resources that don't have more specific settings. Repository-level controls are defined on the **Permission Pattern** tab of the repository ACT.

Resource-Level Controls

Resource-level controls manage access to a specific item such as a report, an information map, a stored process, a table, a cube, or a folder. You can define resource-level controls individually (as explicit settings) or in patterns (by using access control templates).

Fine-Grained Controls

Fine-grained controls affect access to subsets of data within a resource. To establish fine-grained controls, you define permission conditions that constrain access to rows within a table or members within an OLAP dimension.

Feature-Level Controls

Some applications use roles to limit access to functionality. These applications check each user's roles in order to determine which menu items and features to display for that user. Roles are not an authorization feature; they are managed and documented as part of user administration.

Inheritance and Precedence of Permissions

Two Relationship Networks

Permission settings are conveyed across two distinct relationship networks, a resource network and an identity network. Permissions that are set directly on an item have priority over permissions that are set on the item's parent. For example, when access to a report is evaluated, a denial that is set on the report (and assigned to the PUBLIC group) overrides a grant that is set on the report's parent folder (even if the grant is assigned to you).

The Resource Relationships Network

Permissions that you set on one item can affect many other items. For example, a report inherits permissions from the folder in which the report is located. This relationship network consists primarily of a folder tree. This list highlights exceptions:

- The root folder isn't the ultimate parent. This folder inherits from the repository (through the permission pattern of the repository ACT).
- The root folder isn't a universal parent. Some system resources (such as application servers, identities, and ACTs) aren't in the folder tree so they have the repository as their immediate and only parent.
- Inheritance within a table or cube follows the data structure. For example, table columns and cube hierarchies don't have a folder as their immediate parent. Instead, a column inherits from its parent table and a hierarchy inherits from its parent cube.
- In unusual circumstances, it is possible for an item to have more than one immediate parent. If there is a tie in this network (for example, if there are no settings on an item, the item has two immediate parents, and one parent provides a grant while the other parent provides a denial), the outcome is a grant. In other words, a grant from any inheritance path is sufficient to provide access.

The Identity Relationships Network

Permissions that you assign to one group can affect many other identities. For example, if you grant a group access to an OLAP cube, that grant applies to all users who are members of the group. This relationship network is governed by a precedence order that starts with a primary identity, can incorporate multiple levels of group memberships, and ends with implicit memberships in SASUSERS and then PUBLIC. If there is a tie in this network (for example, if you directly assign a user to two groups and give one group a grant and another group a deny), the outcome is a deny.

Use and Enforcement of Each Permission

Table 1.3 *Use and Enforcement of Each Permission*

Permission (Abbreviation)	Actions Affected and Limitations on Enforcement
ReadMetadata (RM)	View an item or navigate past a folder. For example, to see an information map you need RM for that information map. To see or traverse a folder you need RM for that folder.
WriteMetadata (WM)	Edit, delete, change permissions for, or rename an item. For example, to edit a report you need WM for the report. To delete a report you need WM for the report (and WMM for the report's parent folder). WM affects the ability to create associations. For example, you need WM on an application server in order to associate a library to that server. WM affects the ability to create items in certain containers. For example, to add an item anywhere in a repository you need WM at the repository level. For folders, adding and deleting child items is controlled by WMM, not WM.
WriteMemberMetadata (WMM)	Add an item to a folder or delete an item from a folder. For example, to save a report to a folder you need WMM for the folder. To remove a report from a folder, you need WMM for the folder (and WM for the report). To enable someone to interact with a folder's contents but with not the folder itself, grant WMM and deny WM.*

Permission (Abbreviation)	**Actions Affected and Limitations on Enforcement**
CheckInMetadata (CM)	Check in and check out items in a change-managed area. Applicable only in SAS Data Integration Studio.**
Administer (A)	Operate (monitor, stop, pause, resume, refresh, or quiesce) servers and spawners. For the metadata server, the availability of similar tasks is managed by the Metadata Server: Operation role (not by this permission).
Read (R)	Read data. For example, while you need RM for a cube in order to see a cube, you need R for that cube in order to run a query against it. Enforced for OLAP data, information maps, data that is accessed through the metadata LIBNAME engine, and dashboard objects.
Create (C)	Add data. For example, on a table, C controls adding rows to the table. Enforced for data that is accessed through the metadata LIBNAME engine.
Write (W)	Update data. For example, on a table, W controls updating the rows in the table. Enforced for data that is accessed through the metadata LIBNAME engine, for publishing channels, and for dashboard objects.
Delete (D)	Delete data. For example, D on a library controls the deletion of tables from the library. Enforced for data that is accessed through the metadata LIBNAME engine and for dashboard objects.

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

** In any change-managed areas of a foundation repository, change-managed users should have CM (instead of WM and WMM).

Note: The table server supports additional permissions. See the documentation for that component.