THE POWER TO KNOW.

SAS® 9.3 Guide to BI Row-Level Permissions

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Guide to BI Row-Level Permissions*. Cary, NC: SAS Institute Inc.

SAS® 9.3 Guide to BI Row-Level Permissions

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

1st electronic book, July 2011

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

Contents

About This Book

Audience

This book explains how to implement row-level permissions through SAS Information Maps. Implementing BI row-level permissions is a specialized, advanced task. The primary audience for this book is experienced information map creators.

The following documents provide background information and instructions for supporting tasks:

- *SAS Information Map Studio: Getting Started with SAS Information Maps* (available from **support.sas.com/documentation/onlinedoc/ims**)
- *SAS Management Console: Guide to Users and Permissions* (available from **support.sas.com/documentation/onlinedoc/sasmc**)

Chapter 1

Overview of BI Row-Level Permissions

Introduction to BI 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.

BI row-level permissions filter SAS data sets and third-party relational tables when those data sources are accessed through a SAS Information Map. The initials BI indicate that this is a business intelligence feature. BI row-level permissions are defined in information maps, mediated and enforced by SAS Intelligent Query Services, and surfaced when reports are viewed in applications such as SAS Web Report Studio. BI row-level permissions are based on filters and rely on target data that is modeled to work with those filters.

BI row-level permissions offer the following 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 authorization facility of the SAS Metadata Server.

- 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 the following points:

- Although 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. See Chapter 6, "Secure Environment for BI Row-Level Permissions," on page 33.

  CAUTION:
  BI row-level permissions are defined within information maps, so these constraints do not provide comprehensive security for the underlying data sources. A user who accesses the data directly is not subject to the filters that are defined within information maps.

- Although 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.

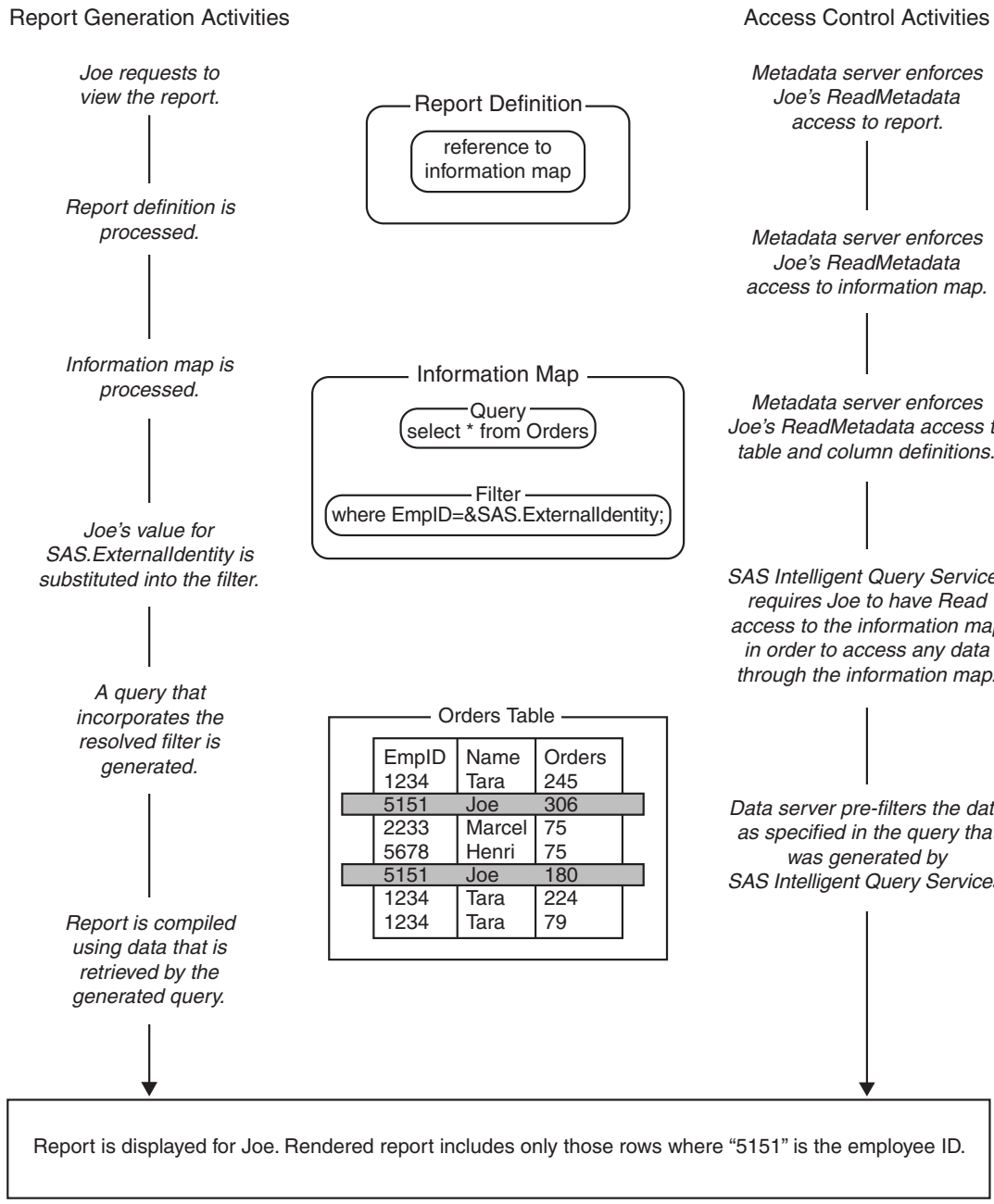
  TIP For information about other implementations of fine-grained controls, see the "Authorization Model" chapter in the *SAS Intelligence Platform: Security Administration Guide*.

- 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. See "Batch Reporting Considerations" on page 10.

Report Generation Process

The following figure depicts how BI row-level permissions are incorporated when a report is generated. In the figure, a user requests access to a report that includes data for which row-level permissions have been defined dynamically. For each step of the report-generation process, the figure depicts the access control activities in the metadata layer.

Figure 1.1 *Report-Generation Process*



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

- The information map includes a filter that is assigned to a particular metadata identity. This example uses an identity-driven property in a filter that is based on each group member's employee ID. The filter is assigned to a group to which Joe belongs. At run time, SAS Intelligent Query Services uses information from the metadata repository to substitute Joe's employee ID into the filter. The resolved, user-specific form of the filter is incorporated into the generated query. The filter is used to screen the target table before the rest of the generated query runs.
- The target data includes information that corresponds to the filter. In this example, the corresponding information consists of user-specific employee ID values in the EmpID column within the Orders table. The data server uses these values to filter the data as specified in the query that was generated by SAS Intelligent Query Services.

Filtering Techniques

Pre-Screening Methods

You define BI row-level permissions in filters that you assign to tables within an information map. To ensure that target data is pre-screened by a filter before any other criteria are applied, incorporate that filter into an information map as a prefilter. Use either of the following methods:

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

  Conditional Grant
  : The requesting user can access only those rows that meet specified SQL filtering conditions.

Filter Types

BI row-level permissions can use the following types of filtering:

static filtering
: compares values in the target data to a specified value. This enables you to implement a specific, fixed rule such as "Joe can see his salary information."

dynamic filtering
: 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." See "Identity-Driven Properties" on page 5.

Summary of Filtering Techniques

Table 1.1 *Summary of BI Row-Level Filtering Techniques*

	Pre-screening Method	
Filter Type	**General Prefilter**	**Authorization-Based Prefilter**
Dynamic (identity-driven)	• Per-person access distinctions for everyone. • One filter, attached directly to an information map. Example: "everyone can see his or her own salary".	• Per-person access distinctions for every member of a particular group. • One filter, assigned to a group in an access control for the information map. Example: "everyone in GroupA can see his or her own salary".
Static (fixed value)	• One fixed subset for everyone. • One filter, attached directly to an information map. Example: "everyone can see global totals".	• A few distinct subsets, each for a particular group. • Several filters, each assigned to a group in an access control for the information map. Example: "everyone in GroupA can see totals for the West region; everyone in GroupB can see totals for the East region".

Note: A static general prefilter is not a true row-level technique because it does not yield different results for different users.

Identity-Driven Properties

Overview

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

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

1. Each identity-driven property corresponds to a characteristic (such as name, user ID, or external identity).

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

These are the most useful identity-driven properties:

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

SAS.ExternalIdentity
: returns a site-specific value (for example, employee ID). This property is often useful because its values are likely to match user information in your data. An identity can have more than one external identity value. However, only the first value is returned. Unlike the values for other identity-driven properties, values for this property are not always populated in the metadata. An external identity value functions as a synchronization key in the user bulk load and synchronization macros. For more information, see the *SAS Intelligence Platform: Security Administration Guide*.

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

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

These identity-driven properties are also supported:

SAS.IdentityGroupName
: returns a group name, as displayed in the **Name** field on the group's **General** tab. If a user logs on with an ID that is stored in a login on a group definition, then the name of the group that owns that login is returned. If a user logs on with a user ID that is not stored in the metadata, then the PUBLIC group is returned.

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

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

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

Examples of Identity-Driven Substitutions

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

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

Table 1.2 *Examples of Filters That Use Identity-Driven Properties*

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

* Because the user does not log on with a user ID that is stored as part of a group definition, the user has no value for this property. This returns an empty string.

Missing Values in Identity-Driven Properties

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

Here are some alternatives for addressing missing values:

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

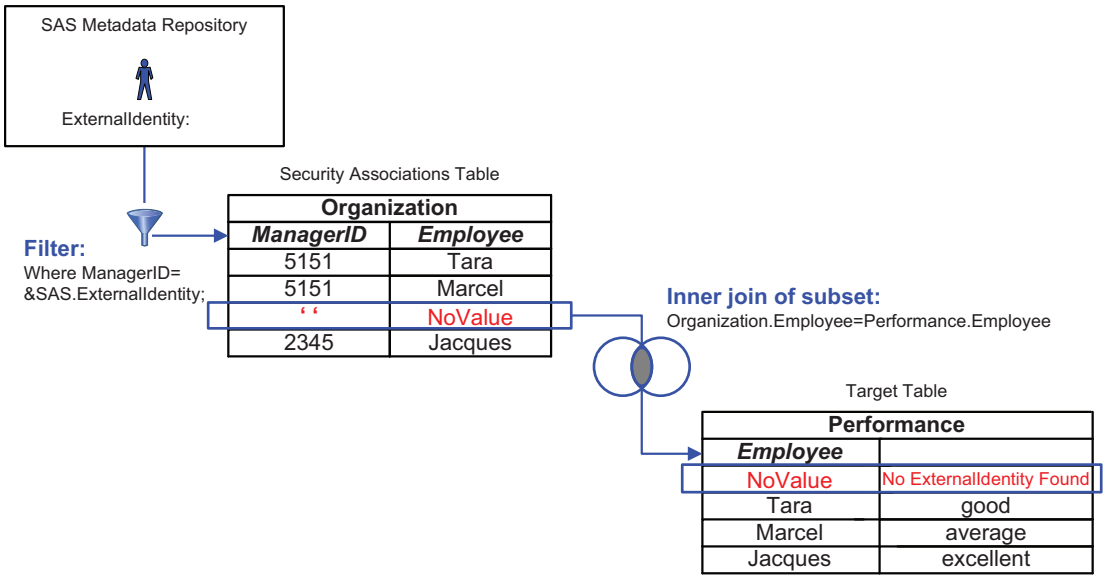
  Note: If the target table is in a DBMS, the extra row must contain an empty (blank) character string (not a NULL value).

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

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

  The following figure depicts an example:

Figure 1.2 *Example: Error Handling for a Missing External Identity Value*



Precedence Considerations

When an authorization decision is made, a permission condition is applied only if it is on the explicit control that is closest to the requesting user. Other conditions that are relevant because of further-removed group memberships don't provide additional, cumulative access.

If there is an identity precedence tie between multiple groups at the highest level of identity precedence, those tied conditions are combined in a Boolean OR expression (any row that meets either condition is returned). If the identity precedence tie includes an unconditional grant, access is not limited by any conditions.

The following table provides examples:

Table 1.3 *Precedence for Permission Conditions*

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

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

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

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

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

Batch Reporting Considerations

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

- Manually refreshed reports contain cached data (which can be updated by a user action in the report viewer).
- Pre-generated reports reflect the access of the user ID that was used to generate the report. Identity-specific access distinctions are preserved for pre-generated reports only if you define a separate report job for each user ID.

Chapter 2

Implementation Process for BI Row-Level Permissions

Preliminary Tasks

1. If you need comprehensive security, set up the secure configuration. See Chapter 6, "Secure Environment for BI Row-Level Permissions," on page 33.
2. Make sure that appropriate table-level and information map-level controls are in place in the metadata layer.

   Table 2.1 *Coarse-Grained Controls*

Access	Target Table	Information Map
All rows	Grant Read, ReadMetadata	Grant Read, ReadMetadata
No rows	Deny Read, ReadMetadata	Grant Read, ReadMetadata
Some rows	Grant Read, ReadMetadata	Grant Read, ReadMetadata

   Note: The grants for the "No rows" users are necessary only if those users access other tables through the information map.
3. Plan how you will create the data subsets that will narrow access for each user. Your choice will be affected by the number and type of access distinctions that you are making, the information that your data already contains, and your plans for enhancing your existing data to support row-level filtering.

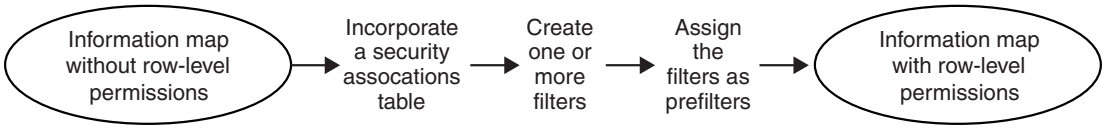
   Note: In a situation in which multiple filters (multiple distinct permission conditions) apply to a particular user as a result of the user's group memberships, the subset of data that is available to that user is determined by identity precedence. See "Precedence Considerations" on page 8.

Information Map Tasks

Summary

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

Figure 2.1 *Information Map Design for Row-Level Permissions*



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

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. One way to do this is to add a table that expresses the relationship between users and data.

Note: This type of table is called a security associations table. For more information, see Chapter 5, "Data Modeling for BI Row-Level Permissions," on page 27.

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

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

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

Create an Identity-Driven Filter

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

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

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

   The examples column in the New Filter dialog box shows what your values are for each property.

   - The SAS.PersonName value corresponds to the **Name** field on the **General** tab of your user definition. This is not always the same as the value of the **Display Name** field.
   - The SAS.IdentityGroupName value is usually blank and isn't often useful.
   - Although the SAS.IdentityGroups property displays only one value, this property actually returns a list of the groups and roles that you belong to.
   - The SAS.ExternalIdentity value is populated only in certain circumstances.
4. Select the row for the identity-driven property that you want to use in the filter.

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

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

Assign the Filter as a Prefilter

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

To assign a filter as a general prefilter:

1. Open the information map and select **Edit** ⇨ **Properties** ⇨ **Information Map**.

   TIP If this action is not available, save the information map and try again.
2. In the Information Map Properties dialog box, select the **General Prefilters** tab.
3. In the **Selected filters** list (the right-hand panel), select your security associations table.
4. In the **Available filters** list, select the filter and then use the arrow button to move the filter to the **Selected filters** list. Click **OK**.

To assign a filter as an authorization-based prefilter:

1. Open the information map and select **Tools** ⇨ **Authorization** to open the Authorization dialog box.

TIP If this action is not available, save the information map and try again.

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

Verification

You can do some preliminary testing to check your filter logic from within SAS Information Map Studio. 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 static authorization-based prefilters, temporarily assign different filters to your identity.

TIP Before you test from within SAS Information Map Studio, save the information map (to ensure that all settings are applied).

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

Chapter 3

Simple Examples

About These Examples

This chapter provides basic demonstrations to illustrate how identity-driven properties work. For convenience, the examples in this chapter use tables in the SASHELP library, which is available in most installations. The examples are not intended to reflect complete security scenarios or best practices for efficient data manipulation.

Detailed instructions for using SAS Information Map Studio or for registering users, groups, and tables are beyond the scope of this document. If you need additional instructions or background information, you might find the following documents useful:

- *SAS Information Map Studio: Getting Started with SAS Information Maps* (available from **support.sas.com/documentation/onlinedoc/ims**)
- *SAS Management Console: Guide to Users and Permissions* (available from **support.sas.com/documentation/onlinedoc/sasmc**)

General Prefilter with SAS.PersonName

Introduction

This example subsets data in the SASHELP.CLASS table based on each requesting user's name. The goal is to enable one user (UserA) to view only rows that contain information about females and another user (UserB) to view only rows that contain information about males. The example assumes that UserA and UserB already exist in the SAS metadata.

Map the Users to the Data

One way to define the relationships between users and data is to create a security associations table. For this example, you could use code like this:

```
data sashelp.rlp_class;
   input PersonName $13. @14 Gender $;
   datalines;
UserA F
UserB M
;
run;
```

Note: You must register the table in the SAS metadata in order to make it available for use in the information map.

TIP As an alternative to creating a separate security associations table, you could add a PersonName column to the CLASS table. In this case, *UserA* would be the value in the female rows and *UserB* would be the value in the male rows.

Create the Information Map

1. In SAS Information Map Studio, create a new information map and insert the CLASS table and the RLP_CLASS table.
2. On the **Design** tab, move all of the columns in the CLASS table to the **Information Map Contents** panel. Do not move any columns from the RLP_CLASS table. Save the information map.
3. From the main menu, select **Edit** ⇨ **Properties** ⇨ **Information Map**. On the **Required Tables** tab, make RLP_CLASS a required table. Click **OK**.
4. On the **Relationships** tab, associate **Sex** in the CLASS table with **Gender** in the RLP_CLASS table.

Create and Assign the Filter

1. Select the **Design** tab. From the main menu, select **Insert** ⇨ **New Filter**.
2. In the New Filter dialog box, name the filter and then click **Edit Data Item**.
3. In the Edit Expression dialog box:

a. Select **Character** as the type of expression.

b. On the **Data Sources** tab (beneath the **Expression Text** box), under the **Physical Data** node, expand the RLP_CLASS table and select the **PersonName** column. Click **Add to Expression**.

c. Click **OK** to return to the New Filter dialog box.

4. From the **Condition** drop-down list, select **Is equal to**.
5. From the **Enter values** drop-down list, select **Derive identity values**.
6. In the table of properties, select **SAS.PersonName**.

   Note: The values in the **Examples** column are derived from the currently logged-on user. These examples show how the value should appear in the security associations table. If the security associations table included user IDs instead of user names (for example, the values **UserA@saspw** and **UserB@saspw**), then you would select the **SAS.Userid** property instead of the **SAS.PersonName** property.

7. (Optional) At the bottom of the dialog box, expand **Filter expression** to review the criteria that you have defined. Click **OK**.
8. From the main menu, select **Edit** ⇨ **Properties** ⇨ **Information Map**.
9. On the **General Prefilters** tab, assign the new filter to the CLASS table.

   Note: A prefilter is a mandatory filter that pre-screens and subsets the data in its associated table before any other part of a query is run.

Test the Filter

1. Log on to SAS Information Map Studio as UserA.
2. Open the information map. Select **Tools** ⇨ **Run a Test Query**.
3. Select all of the data items. You do not have to select the filter because you made it required.
4. Click **Run Test**. You will see only rows for females, because you are logged on as UserA.
5. Log on as UserB and verify that you see only rows for males.
6. (Optional) To test the filter in SAS Web Report Studio, open the information map as a report.

Note: If you log on as someone other than UserA or UserB, you will get no results, because you are not represented in the security associations table. The general prefilter affects access for everyone (even unrestricted users) if the data is accessed through the information map. However, if anyone opens the CLASS table directly, without going through the information map, the filter is not applied so all rows are returned.

General Prefilter with SAS.IdentityGroups

Introduction

This example subsets data in the SASHELP.PRDSALE table based on each requesting user's metadata group memberships. The goal is to let each sales person see only rows for his or her region, and to let the sales manager see rows for two regions. The example assumes that the following tasks are already completed:

- In SAS Information Map Studio, the PRDSALE table has been used to create an information map named **`Sales Map`**.
- The following users and memberships exist in the SAS metadata.

User	Job	Group Memberships
UserA	Sales person, East region	EAST
UserB	Sales person, West region	WEST
UserC	Sales manager, East and West regions	EAST, WEST

Map the Users to the Data

In this example, the filtering is based on each user's metadata group memberships. The example uses the SAS.IdentityGroups property, which exploits the metadata server's knowledge of user-group relationships (so it is not necessary to model those relationships in an external table). Further, the names of the involved metadata groups (EAST and WEST) exactly match values in an existing column in the PRDSALE table (the REGION column), so no external table is needed create that mapping. For these reasons, this example does not use a separate security associations table.

Create and Assign the Filter

1. In SAS Information Map Studio, open the information map and select **Insert** ⇨ **New Filter**.
2. In the New Filter dialog box:
   a. Name the filter *Region Memberships Filter*.
   b. From the **Data Items** drop-down list, select **Region**.
   c. From the **Condition** drop-down list, select **Is equal to**.
   d. From the **Enter values** drop-down list, select **Derive identity values**.
   e. In the table of properties, select **SAS.IdentityGroups**.
   f. Click **OK**.

   Note: The value of your data item must exactly match the name of the metadata group.

Note: The examples column shows values for the user that is currently logged on.

3. From the main menu, select **Edit** ⇨ **Properties** ⇨ **Information Map**.
4. On the **General Prefilters** tab, assign the new filter to the PRDSALES table.

   Note: A prefilter is a mandatory filter that pre-screens and subsets the data in its associated table before any other part of a query is run.

Test the Filter

1. Log on to SAS Information Map Studio as UserA.
2. Open the information map. Select **Tools** ⇨ **Run a Test Query**.
3. Move all of the data items to the **Selected items** list. You do not have to select the filter because you made it required.
4. Click **Run Test**. You will see only the EAST rows, because you are logged on as UserA.
5. Log on as UserB and then as UserC, to see whether the results are as expected.
6. (Optional) To test the filter in SAS Web Report Studio, log on and open the information map as a report.

Note: If you log on as someone who is not a member of any of the region groups, you will get no results, because none of your group memberships will match a value in the REGION column of the PRDSALE table. The general prefilter affects access for everyone (even unrestricted users), if the data is accessed through the information map. However, if anyone opens the PRDSALE table directly, without going through the information map, the filter is not applied so all rows are returned.

Authorization-Based Prefilter with SAS.IdentityGroups

Introduction

This example is a variation on the preceding example. Instead of using a general prefilter, which applies universally to all requesting users, this example implements the SAS.IdentityGroups filter as an authorization-based prefilter. With this approach, you can establish different logic for different groups of users. In this example, we introduce a group of executives who should not be subject to the SAS.IdentityGroups filtering. Instead, members of the Executives group should see all of the data.

The following table summarizes the strategy:

Table 3.1 *Example: Two Classes of Access*

Access Class (User Group)	Direct Access Controls
Full Access (Executives)	Grant Read [Explicit]
Filtered Access (PUBLIC)	Grant Read [Conditional]

The first steps, mapping users to data and creating the information map and filter are identical to the preceding example. However, instead of editing the information map's properties to assign the filter as a general prefilter, complete the steps in the following topics.

Assign the Filter to the PUBLIC Group as a Permission Condition

1. In SAS Information Map Studio, open the information map and select **Tools** ⇨ **Authorization** to open the Authorization dialog box.
2. In the **Users and Groups** list, select **PUBLIC**. In the **Effective Permissions** list, add an explicit ☑ grant for the Read permission.
3. To limit the PUBLIC grant of the Read permission, assign the IdentityGroups filter to that group as a permission condition.
   a. Click **Add Condition** to open the Row-Level Permission Condition dialog box.

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

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

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

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

Give Executives an Explicit Grant of the Read Permission

1. In the Authorization dialog box, click **Add**. In the Add Users and Groups dialog box, select the Executives group and then click **OK**.
2. In the **Effective Permissions** list, give the Executive group an explicit grant of the Read permission. Because you want this group to be able to view all data, do not constrain Read access by applying a permission condition.
3. Click **Close**. To make your changes take effect, save the information map.

Test the Filter

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

- Users who aren't in the EAST, WEST, or Executive groups get no rows.
- Users who are in the EAST or WEST groups get filtered access, as in the preceding example.
- Users who are in the Executives group get all rows (unless they are also members of the EAST or WEST groups at a higher precedence level than they are in the Executives group).

Note: If anyone opens the target table directly, without going through the information map, the filter is not applied so all rows are returned.

Chapter 4

Complete Example

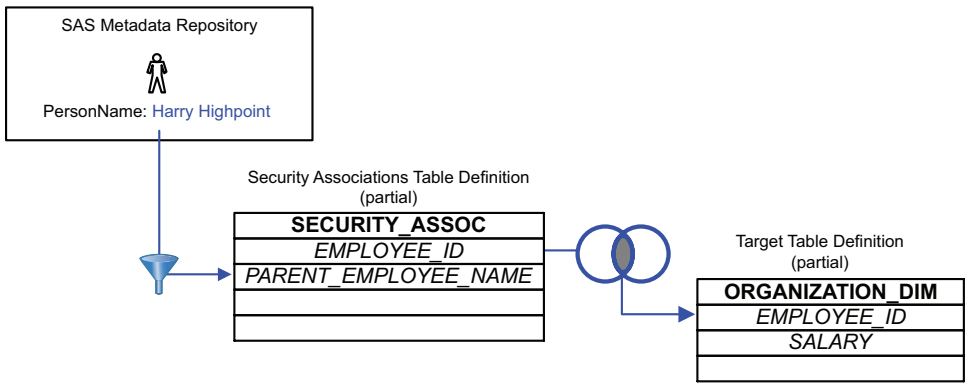
Assumptions and Data Model

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

- The target tables are registered in the metadata repository.
- Except where otherwise noted, users have Read permission for the information maps that they are using.
- The data model is a star schema that contains employee and customer data. The security associations table includes both direct and indirect reporting relationships.
- The company has set up the secure configuration. See Chapter 6, "Secure Environment for BI Row-Level Permissions," on page 33.

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

Figure 4.1 *Salary Example: Data Model*



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

Implementation and Testing

To implement the filtering for this example:

1. Create an information map that includes the necessary data and relationships.
   a. In SAS Information Map Studio, open a new information map.
   b. Insert the target table and the security associations table as data sources. In this example, the target table (ORGANIZATION_DIM) contains salary data, and the security associations table (SECURITY_ASSOC) contains a representation of the company's reporting relationships.
   c. On the **Design** tab, add the data items that you need from the ORGANIZATION_DIM table (insert the SALARY, EMPLOYEE_ID, and EMPLOYEE_NAME columns).

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

   d. On the **Relationships** tab, join the two tables on EMPLOYEE_ID.
   e. Save the new information map to an appropriate folder.
   f. To make SECURITY_ASSOC a required table, select **Edit** ⇨ **Properties** ⇨ **Information Map**. In the Information Map Properties dialog box, select the **Required Tables** tab. In the **Available tables** list, select the SECURITY_ASSOC table. Use the arrow button to move the table to the **Required tables** list, and then click **OK**.
2. Create a filter that subsets data by comparing each user's SAS.PersonName value to the PARENT_EMPLOYEE_NAME values in the security associations table.
   a. Select **Insert** ⇨ **New Filter** to open the New Filter dialog box.
   b. Enter a name such as **`byPersonName`** for the filter, and then click **Edit Data Item**.
   c. In the Edit Expression dialog box, select **`Character`** from the **Type** drop-down list. On the **Data Sources** tab, navigate to **Physical Data** ⇨ **SECURITY_ASSOC** ⇨ **PARENT_EMPLOYEE_NAME**, and then click **Add to Expression**.
   d. Click **Validate Expression**, and then click **OK** twice.
   e. In the New Filter dialog box, from the **Enter value(s)** drop-down list, select **Derive identity values (for row-level permissions)**. A table of identity-driven properties becomes available.

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

   f. In the table of properties, select the SAS.PersonName row.
   g. Click **OK**. The byPersonName filter is now available for use in the information map.
3. Assign the filter as a general prefilter:

a. Select **Edit** ⇨ **Properties** ⇨ **Information Map**.

b. In the Information Map Properties dialog box, select the **General Prefilters** tab.

c. In the **Selected filters** box, select the SECURITY_ASSOCIATIONS table.

d. In the **Available filters** box, select the byPersonName filter.

e. Click the right arrow button to assign the byPersonName filter to the SECURITY_ASSOC table, and then click **OK**.

4. Save the information map.

Administrators can test by logging on to SAS Information Map Studio and running test queries. To verify that the filter is working as expected, log on using different accounts. For example:

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

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

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

Note: In the secure configuration, final verification must be performed from within SAS Web Report Studio.

Variation 1: SAS.ExternalIdentity Property

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

- Verify that users have the external identity values that you expect. In SAS Management Console, an **External Identities** button on each user definition provides access to this information.
- Use the PARENT_EMPLOYEE_ID column instead of the PARENT_EMPLOYEE_NAME column.
- Use the SAS.ExternalIdentity property instead of the SAS.PersonName property.

Variation 2: Authorization-Based Prefilter

This variation addresses the following additional business requirements:

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

This table summarizes the strategy:

Table 4.1 *Information Map Controls*

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

Note: 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.

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

To set the permissions:

1. Prepare the information map by using either of these methods:
   - Create a new information map for this variation by completing steps 1 and 2 in the main example.
   - Reuse the information map from the main example by saving that map with a different name and deassigning the filter that was assigned on the **General Prefilters** tab.
2. Open the information map and select **Tools** ⇨ **Authorization** to open the Authorization dialog box.
3. In the **Users and Groups** list, select **PUBLIC**. In the **Effective Permissions** list, add explicit ☑ denials for the Read and ReadMetadata permissions.
4. Click **Add**. In the Add Users and Groups dialog box, select the HR and SASUSERS groups and then click **OK**.
5. In the Authorization dialog box, give SASUSERS explicit ☑ grants of the Read and ReadMetadata permissions.
6. To limit the SASUSERS grant of the Read permission, assign the byPersonName filter to that group.
   a. Click **Add Condition** to open the Row-Level Permission Condition dialog box.

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

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

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

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

7. In the Authorization dialog box, give the HR group explicit grants of the Read and ReadMetadata permissions. Because you want this group to be able to view all salaries, do not constrain Read access by adding a permission condition.
8. In the Authorization dialog box, click **Close**. To make your changes take effect, save the information map.

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

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

Chapter 5

Data Modeling for BI Row-Level Permissions

Overview and Examples

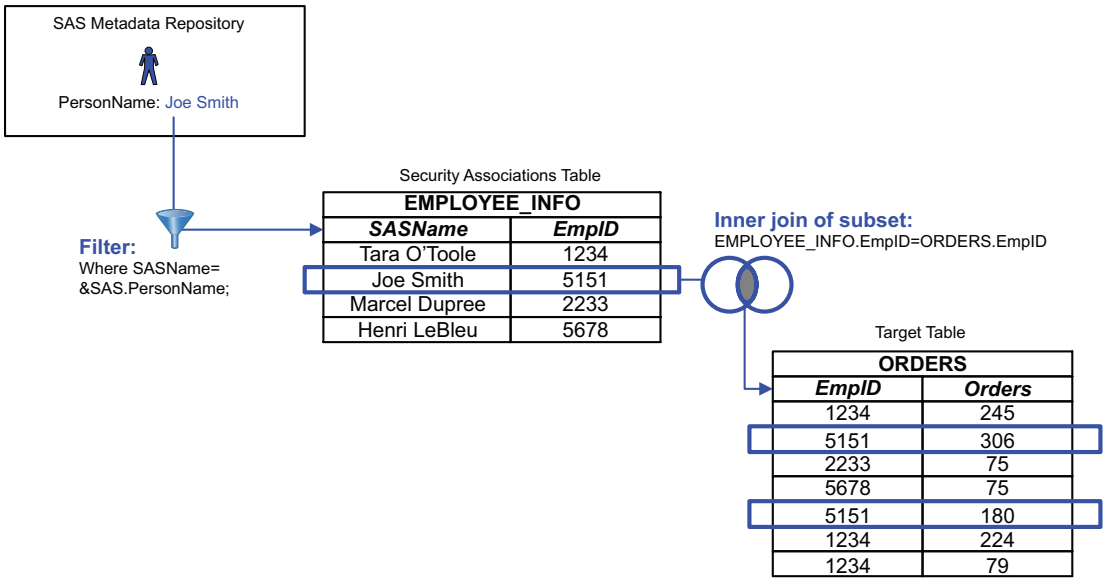
BI row-level permissions are based on filters and rely on target data that is modeled to work with those filters. It is usually necessary to enhance existing data to include information that works with the filters that you want to use. For example, consider a four-person company with a flat organizational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

Figure 5.1 *Orders Example: Target Table*

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

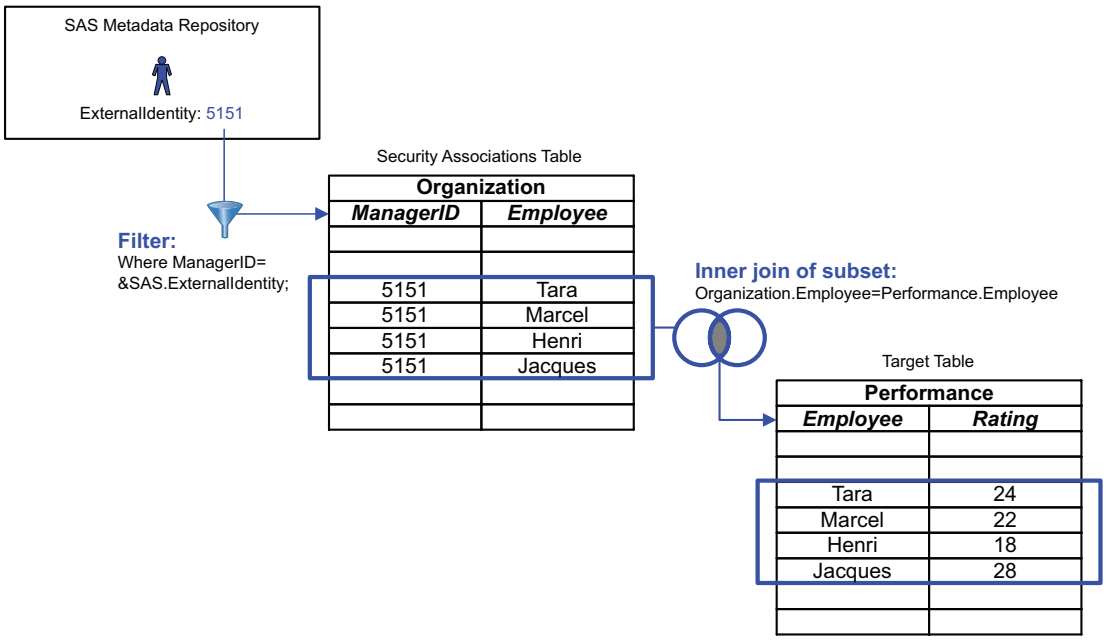
Assume that you didn't import users (so you don't have SAS.ExternalIdentity values in the metadata that correspond the EmpID values in the ORDERS table). To avoid setting up a different filter for each user, you decide to use the SAS.PersonName identity-driven property. Create a table that maps each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

Figure 5.2 *Orders Example: Data Model*



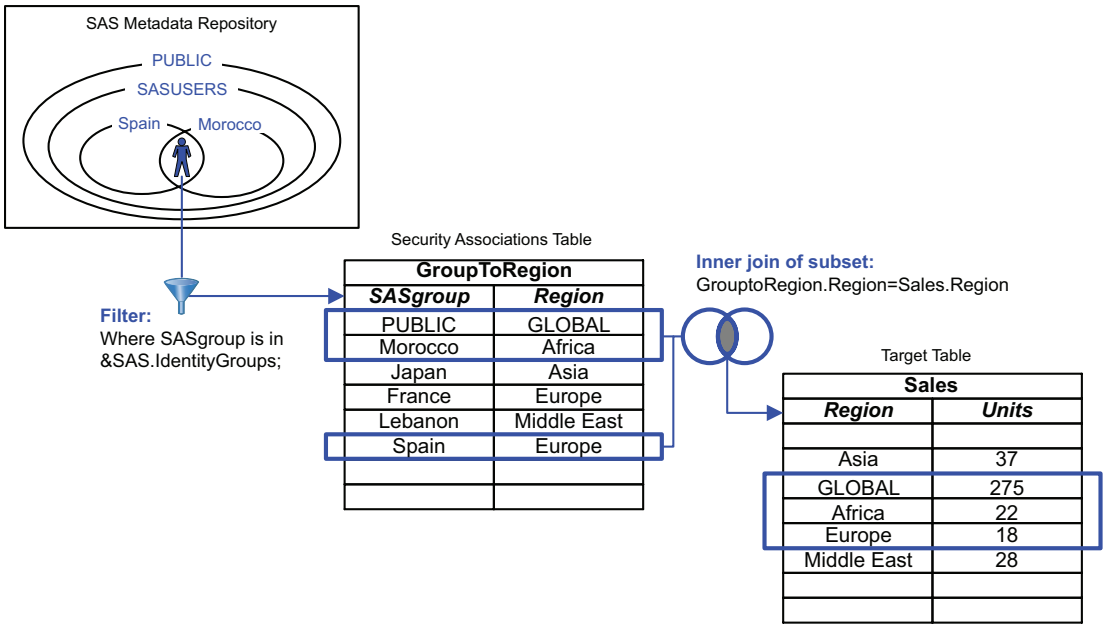
Another simple example is to subset employee performance information based on each manager's external identity value, as depicted in the following figure.

Figure 5.3 *Performance Example: Data Model*



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

Figure 5.4 *Sales Example: Data Model*



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

Content of a Security Associations Table

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

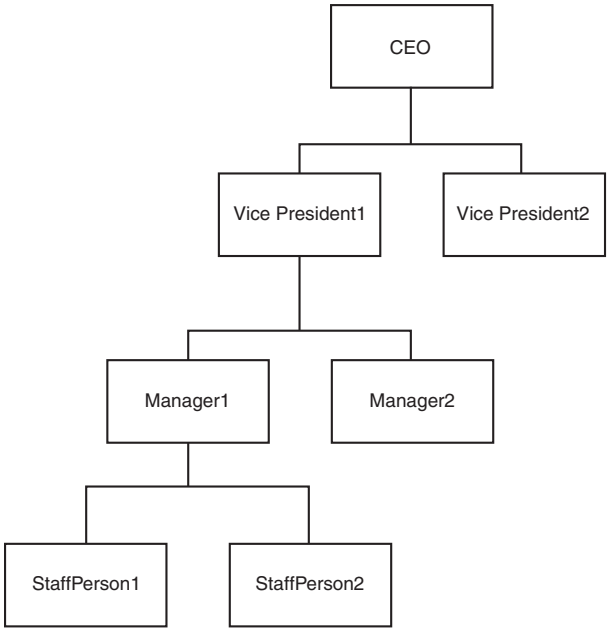
Format of a Security Associations Table

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

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

includes indirect reporting relationships along with direct reporting relationships; this is advantageous for query performance.

Figure 5.5 *Representations of an Organizational Hierarchy*



Security Associations Table (sparse)

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

Security Associations Table (fully articulated)

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

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

Creation and Maintenance of a Security Associations Table

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

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

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

In most cases it is helpful to have an index on the column in the security associations table that is used for filtering. In some cases, factors such as the size of the security

associations table or query optimization features in a particular data source might negate the need for this index.

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

Chapter 6

Secure Environment for BI Row-Level Permissions

Security and BI Row-Level Permissions

Introduction

Like any other security feature, a secure implementation of BI row-level permissions 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 filters that you define in your information maps. If this is an acceptable risk, then no special measures are needed. This can be an acceptable risk in environments such as the following:

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

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

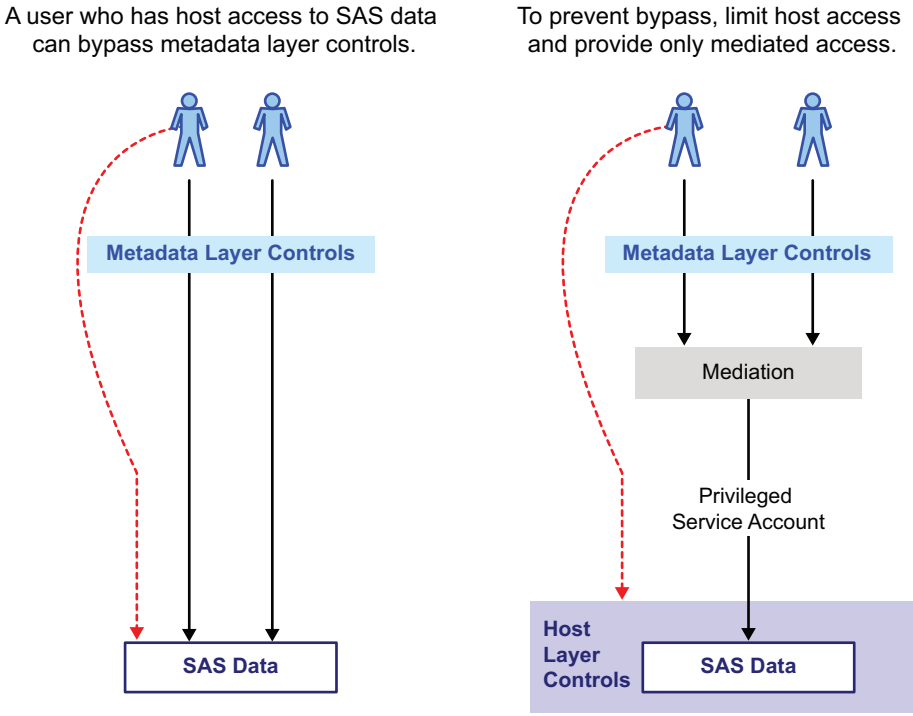
target tables to prevent direct access by regular users. The goal is to enable regular users to have only mediated access to the target tables. The strategy is as follows:

- Deny regular users physical access to the tables (using host or DBMS access controls).
- Require participating applications to use a dedicated, privileged account to fetch data for requesting users.
- Configure the retrieving server and environment in a way that minimizes the risk of a user exploiting the privileged account, or otherwise circumventing the row-level filters.

The Initial Configuration

The initial configuration in a new deployment uses mediated access for BI row-level permissions. The server-side pooled workspace server is used for queries against relational information maps. The following figure depicts this mediation of physical access.

Figure 6.1 *Example of Mediated Physical Access to Sensitive Data*



In the initial configuration, mediation is achieved through server-side workspace server pooling. The server's launch credential retrieves data for all requesting users. The server's launch credential must have physical access to the data. End-users don't need physical access to the data. This mediation, in combination with appropriate physical layer access controls, provides some separation, because it prevents users from directly accessing the data under their own credentials.

Incremental Measures for Increased Protection

There are several options for incrementally reducing the likelihood of inappropriate access. For example:

- You can give the server-side pooled workspace server a unique launch credential, instead of allowing it to continue to share the stored process server's launch credential.
- You can create an additional server-side pooled workspace server in its own SAS application server, with a dedicated launch credential, and assign sensitive libraries to only that server.

Neither of these options provides comprehensive security, because it is possible for a sophisticated end user to exploit a server-side pooled workspace server and obtain direct (unfiltered) access to the data. For this reason, the secure environment uses a client-side pooled workspace server, instead of a server-side pooled workspace server.

About the Secure Environment

When to Use the Secure Environment

If you require comprehensive security for your BI row-level permissions, you must set up the secure configuration as described in the following topics. This configuration prevents regular users from circumventing row-level filters by accessing the target tables directly.

Overview of the Secure Environment

Here are the key points of the configuration:

- You create a restricted client-side pooled workspace server that uses dedicated accounts for the pool administrator and the puddle login. This isolates the puddle account from applications that do not fully enforce row-level security.
- You create an additional deployment instance of SAS Web Report Studio. The pool administrator account is known to only the new deployment instance of SAS Web Report Studio, so no other applications can use the pool (and, potentially, exploit the puddle login credential).

  Note: In the secure configuration, the only supported client for presenting reports to end users is SAS Web Report Studio. The sensitive data is not available from other applications, such as SAS Enterprise Guide, because physical layer protections grant access to only the puddle account, which is known only to SAS Web Report Studio.

With the secure configuration, sensitive data is available only to designated end users (through SAS Web Report Studio, with row-level filters applied), and to trusted IT staff (through SAS Information Map Studio and through direct access).

Pre-Requisites for Setting Up the Secure Environment

Setting up the secure environment is an advanced task that requires the following preparation:

- a separate, additional middle-tier machine to host the second deployment of SAS Web Report Studio (physical separation of the two deployments is necessary in order to avoid JVM port and context root conflicts)
- the ability to create two new service accounts in the operating system
- familiarity with SAS software installation and deployment, middle-tier administration, and metadata administration

To set up the secure environment, complete all of the tasks in the following topics.

Create Service Accounts and An Access Group

1. In the operating system, create two service accounts that can be authenticated by the workspace server's host:

   rpooladm is the pool administrator account, which handles requests for processes in the restricted workspace server's client-side pool.

   rpoolsrv is the puddle login account, which functions as the launch credential for the restricted workspace server's client-side pool.

   TIP On Windows, give these accounts the **`Log on as a batch job`** Windows privilege (for example, if you have a *SAS Server Users* host group, you can simply add these accounts to that group). You can use domain accounts or local accounts.

2. In SAS Management Console, on the **Plug-ins** tab, use the **User Manager** to create the following identities:

   - User: *Restricted Pool Administrator*

     On the **Accounts** tab, add a new login that contains the *rpooladm* user ID and no password. Assign this login to the authentication domain of your workspace server.

   - Group: *Restricted Puddle Login Group*

     - On the **Accounts** tab, add a new login that contains the *rpoolsrv* user ID and password. Assign this login to the authentication domain of your workspace server.
     - On the **Members** tab, make the *Restricted Pool Administrator* a member of the group.

   - Group: *Restricted Puddle Access Group*

     On the **Members** tab, make the users and groups that will use the restricted pool members of the group. This group does not need any logins.

   TIP On Windows, remember to qualify the user ID that you store in each login (for example, **`WIN\rpoolsrv`**).

Set Up a Second Deployment Instance of SAS Web Report Studio

Create the New Deployment Instance

During installation, the SAS Deployment Wizard creates one SAS Web Report Studio deployment instance. To create an additional deployment instance, you rerun the SAS Deployment Wizard.

Follow these guidelines:

- Create the new deployment instance on a second middle-tier machine (to avoid JVM port and context root conflicts between the two deployment instances).
- Use a deployment plan that contains the SAS middle tier.
- You must choose the **Custom** configuration prompting level.
- Provide connection information for the metadata server that you are currently using (the new deployment instance uses your existing SAS Metadata Server).
- Provide the correct machine name on which you are adding the SAS Web Report Studio deployment instance.
- Give the second SAS Web Report Studio deployment instance a name such as *RestrictedWRS*. Each deployment instance name must be unique. If you leave this field blank, or provide a name that is already in use, the configuration fails.

For more information about the SAS Deployment Wizard, see the *SAS Intelligence Platform: Installation and Configuration Guide*.

Configure Application Properties for the New Deployment Instance

1. Configure the *RestrictedWRS* deployment instance of SAS Web Report Studio to use client-side pooling.
   a. On the **Plug-ins** tab, expand **Application Management** ⇨ **Configuration Manager**. Right-click the *RestrictedWRS* deployment of SAS Web Report Studio and select **Properties**.
   b. Select the **Settings** tab. On the left side, select **Application** ⇨ **Pooling**. On the right side, set the client-side pooling option to **Yes**.

      Note: If you are not able to change this setting, click **Cancel**. Under **Configuration Manager**, right-click **SAS Application Infrastructure**, navigate to its client-side pooling setting, and click the Lock icon (to unlock it). Then return to the properties of the *RestrictedDataReporting* deployment instance of SAS Web Report Studio, and turn on client-side pooling.

   c. On the **Advanced** tab, as the **Property Value** for the **App.ClientSidePoolingAdminID** property, enter the user ID for the pool administrator (*rpooladm*).

      TIP On Windows, qualify the user ID (for example, *WIN\rpooladm*).
2. Constrain the *RestrictedWRS* deployment instance of SAS Web Report Studio so that it accepts relational information maps from only a designated, protected location in the SAS folders tree.
   a. On the **Advanced** tab, click **Add** and use the Define New Property dialog box to add he following properties and values:

Property Name	Property Value
`wrs.map.accessibility.check.enabled`	`true`
`wrs.map.accessibility.check.rootlocations`	*`path`**

* For example, **SBIP://METASERVER/rlp/RestrictedMaps**. To designate multiple locations, enter a comma-separated list. This restriction does not affect OLAP information maps.

b. In SAS Management Console, create the designated folder (if it does not already exist). On the folder's **Authorization** tab, perform the following tasks:

- Explicitly deny the WriteMetadata and WriteMemberMetadata permissions to PUBLIC.
- Explicitly grant the WriteMemberMetadata permission to only those users and groups who should be able to add and update the sensitive relational information maps.
- Make sure that end users of reports based on the secure information maps have ReadMetadata access.

3. To make the application properties changes take effect, restart the Web application server.

Create a Restricted Workspace Server

1. In the host layer, create directories and a configuration file:

   a. In your equivalent of ***SAS-configuration-directory*\SASApp**, create a directory called **RestrictedPool** and a subdirectory (below **RestrictedPool**) called **logs**.

   b. In the **RestrictedPool** directory, create a configuration file to be used when the restricted workspace server is started.

   - On Windows, create a file named sasv9.cfg with the following content:

     ```
     -config "SAS-configuration-directory\SASApp\sasv9.cfg"
     ```

   - On UNIX, create a file named workspaceServer.cfg with the following content:

     ```
     -config !SASROOT/sasv9.cfg
     -config sasv9.cfg
     ```

2. Decide how the restricted workspace server will connect to the metadata server. Choose one of the following approaches:

   - Use trusted peer connections, which the metadata server accepts without requiring credentials. In the initial configuration, the metadata server accepts trusted peer connections from all user IDs and machines, so no special configuration is required. See the "Trusted Peer Connections" in Chapter 10 of *SAS Intelligence Platform: Security Administration Guide*.

     Note: In this approach, the restricted server's processes that are initiated from SAS Web Report Studio run under the *rpoolsrv* identity, and the restricted server's processes that are initiated from a desktop application run under the requesting users's identity. The *Restricted Puddle Login Group* and any allowed individual desktop users must have access to any external DBMS credentials.

   - Use credential-based connections, where the workspace server provides a user ID and password that are stored in its configuration file. In this approach, you add the METAUSER and METAPASS options to the configuration file that you created in step 1b. For example:

     ```
     -metauser "rpoolsrv"
     -metapass "encrypted-rpoolsrv-password"
     ```

CAUTION:
With this approach, it is essential to provide host protection of the configuration file for the restricted workspace server (because it contains privileged credentials).

TIP On Windows, qualify the user ID (for example, *WIN\rpoolsrv*).

TIP Encrypt the password using the PWENCODE procedure. See Chapter 3, "The PWENCODE Procedure," in *Encryption in SAS*.

TIP If you change the *rpoolsrv* account password, you must also manually update the password in this configuration file.

Note: In this approach, all of the restricted server's processes are launched under the *rpoolsrv* identity. Only the *Restricted Puddle Login Group* needs access to any DBMS credentials.

3. In the metadata, define the restricted server.
   a. On the **Plug-ins** tab of SAS Management Console, right-click **`Server Manager`** and select **New Server**.
   b. In the New Server wizard, select **Resource Templates** ⇨ **Servers** ⇨ **SAS Application Server**.

      Note: The restricted workspace server must be in its own dedicated SAS Application Server.

      Click **Next**.
   c. Enter the name **`RestrictedPool`**.

      Click **Next**.
   d. Accept the default version and vendor information.

      Click **Next**.
   e. Select **Workspace Server**.

      Click **Next**.
   f. Select the **Custom** radio button.

      Click **Next**.
   g. Enter a value in the **Command** box as follows:

      For a workspace server on Windows:
      ```
      sas -config "SAS-configuration-directory\SASApp\RestrictedPool\sasv9.cfg"
      ```

      For a workspace server on UNIX:
      ```
      SAS-configuration-directory/SASApp/sas.sh
      -config RestrictedPool/workspaceServer.cfg
      ```

      Click **Next**.
   h. Specify the following values:

      Authentication domain
      Select the authentication domain of your existing, general-purpose workspace server. Usually, this is **`DefaultAuth`**.

      Bridge port
      Change the default value (8591) to an unassigned port value (such as 9591).

      Click **Next**.
   i. Click **Finish**.

4. Tell the object spawner about the restricted server.
   a. Under **Server Manager**, right-click the object spawner, and select **Properties**.
   b. On the **Servers** tab, move **RestrictedPool - Workspace Server** to the **Selected servers** list. Click **OK**.
   c. Restart the object spawner.
5. Test the connection to the restricted server.
   a. Under **Server Manager**, expand the **RestrictedPool** application server and the **RestrictedPool - Logical Workspace Server**. Select the **RestrictedPool - Workspace Server**.
   b. In the right pane, right-click the connection icon and select **Test Connection**.

      Note: If you are logged on with an internal account (an account that has the `@saspw` suffix), you are prompted for credentials. Enter the credentials for a user that has an external account, an individual metadata identity, and (on Windows) the `Log on as a batch job` Windows privilege.

      TIP If the connection fails, select **File ⇨ Clear Credentials Cache** from the main menu and try again. You can also check the log files for the object spawner and the workspace server and make sure the contents of the configuration file in the `RestrictedPool` directory are correct.
6. Configure the restricted server to support client-side pooling.
   a. Right-click the **RestrictedPool - Logical Workspace Server** and select **Convert To ⇨ Pooling**. In the message box, click **Yes**.
   b. In the Pooling Options dialog box, click **New**.
   c. In the New Puddle dialog box, provide values as follows:

Field	Value
Name	`restrictedPoolPuddle`
Minimum available servers	`0`
Minimum number of servers	`0`
Login	`rpoolsrv`
Grant access to group	`Restricted Puddle Access Group`

      Click **OK**.
   d. Click **OK** in the Pooling Options dialog box.

Note: For more information about client-side pooling, see the *SAS Intelligence Platform: Application Server Administration Guide*.

Assign Libraries to the Restricted Server

All libraries that are accessed from the *RestrictedDataReporting* deployment instance of SAS Web Report Studio must be assigned to the restricted server.

Note: Sensitive libraries should be assigned to only the restricted server; general purpose libraries can be assigned to other servers also.

For each library, complete these steps in SAS Management Console:

1. Right-click the library and select **Edit Assignments**.
2. In the Edit Assignments dialog box, hold down the CTRL key and click **RestrictedPool**. (This action selects **RestrictedPool** and leaves items that are already selected in a selected state.) Click **OK**.
3. Right-click the library and select **Properties**.
4. On the **Options** tab, click **Advanced Options**.
5. In the Advanced Options dialog box, select the **Library is pre-assigned** option. Click **OK**.
6. In the Properties dialog box, click **OK**.

All relational information maps that are accessed from the *RestrictedDataReporting* deployment instance of SAS Web Report Studio must meet the following requirements:

- The libraries must be assigned to the restricted server.
- The information maps must be stored in (or below) the metadata folder that you specified in the **`wrs.map.accessibility.check.rootlocations`** property. See step 2 in "Configure Application Properties for the New Deployment Instance" on page 37.

Review and Manage Physical Access to Sensitive Data

Ensure that physical layer protections make your sensitive data resources readable only by *rpoolsrv* and the IT staff. In particular, make sure that the launch credential for your general purpose workspace server (for example, *sassrv*) does not have physical access to the data.

For third-party DBMS data, 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*. For example, to provide access to a DB2 server, give that group a login that includes a DB2 user ID and password and that is associated with the DB2 server's authentication domain.

Note: Some members of your IT staff will also need to be able to authenticate to the database server.

Glossary

authentication
the process of verifying the identity of a person or process within the guidelines of a specific authorization policy.

authentication domain
a SAS internal category that pairs logins with the servers for which they are valid. For example, an Oracle server and the SAS copies of Oracle credentials might all be classified as belonging to an OracleAuth authentication domain.

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.

client-side pooling
a configuration in which the client application maintains a collection of reusable workspace server processes. See also puddle.

connection profile
a client-side definition of where a metadata server is located. The definition includes a computer name and a port number. In addition, the connection profile can also contain user connection information.

credentials
the user ID and password for an account that exists in some authentication provider.

data mart
a collection of data that is optimized for a specialized set of users who have a finite set of questions and reports.

external identity
a synchronization key for a user, group, or role. For example, employee IDs are often used as external identities for users. This is an optional attribute that is needed only for identities that you batch update using the user import macros.

information map
a collection of data items and filters that provides a user-friendly view of a data source. When you use an information map to query data for business needs, you do not have to understand the structure of the underlying data source or know how to program in a query language.

login
a SAS copy of information about an external account. Each login includes a user ID and belongs to one SAS user or group. Most logins do not include a password.

missing value
a type of value for a variable that contains no data for a particular row or column. By default, SAS writes a missing numeric value as a single period and a missing character value as a blank space.

permission condition
a control that defines access to data at a low level, specifying who can access particular rows within a table or particular members within an OLAP cube. Such controls are typically used to subset data by a user characteristic such as employee ID or organizational unit. For example, an OLAP cube that contains employee information might have member-level controls that enable each manager to see the salary history of only that manager's employees. Similarly, a table that contains patient medical information might have row-level controls that enable each doctor to see only those rows that contain data about that doctor's patients.

prefilter
in an information map, a mandatory filter that pre-screens and subsets the data in its associated table before any other part of a query is run. The two types of prefilters are authorization-based prefilters and general prefilters. An authorization-based prefilter applies to a specific user or group, and a general prefilter applies to all users.

puddle
a group of servers that are started and run using the same login credentials. Each puddle can also allow a group of clients to access the servers. See also client-side pooling.

server-side pooling
a configuration in which a SAS object spawner maintains a collection of reusable workspace server processes that are available for clients. The usage of servers in this pool is governed by the authorization rules that are set on the servers in the SAS metadata.

service identity
an identity or account that exists only for the purpose of supporting certain system activities and does not correspond to a real person. For example, the SAS Trusted User is a service identity.

Index

T

U

V