SAS® Activity-Based Management 7.2
Data Administration Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc 2012. *SAS® Activity-Based Management 7.2: Data Administration Guide*. Cary, NC: SAS Institute Inc.

SAS® Activity-Based Management 7.2: Data Administration Guide

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

Electronic book 2, May 2012

Contents

PART 8 Working with Other SAS Programs 395

PART 9 The ABC Procedure 473

Part 1

Managing Permissions

Chapter 1

User Capabilites and Groups

Overview

As an administrator, you create users, roles, and groups. In general, the sequence you will follow is the following:

1. Create roles See "Creating Roles" on page 4.

   A role is a set of capabilities. When you make a group be a member of a role, users who are members of that group inherit the capabilities of the role.

2. Create groups See "Creating Groups" on page 12.

   A group is a set of users who share the same capabilities. It is convenient to create groups before creating users because if groups exist, then when you create a user you can assign the user to one or more groups and thereby determine the user's capabilities.

3. Create users and assign them to groups. See "Creating Users" on page 17.

What features of SAS Activity-Based Management are available to a user is determined by the combination of the following two factors:

- the capabilities that the user inherits from the groups to which the user belongs
- the permissions that are granted to the groups to which the user belongs

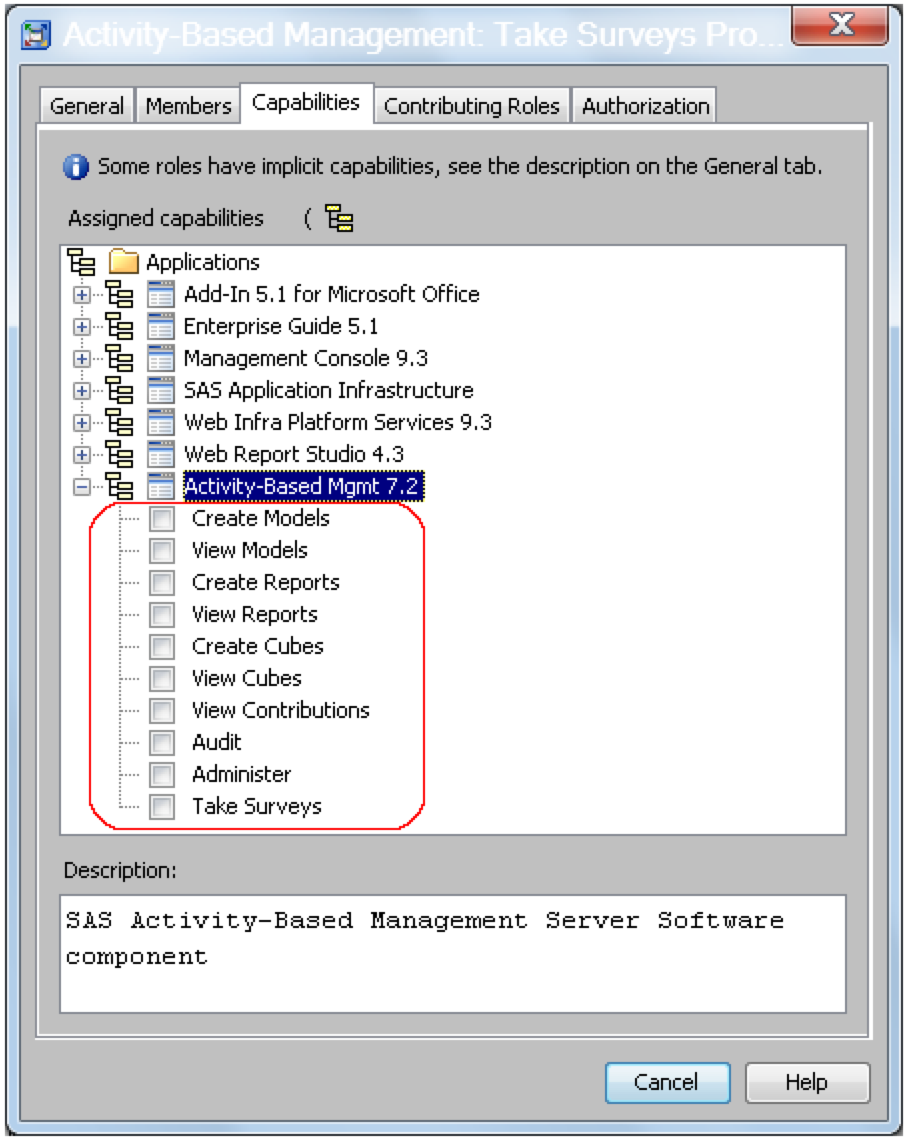
For a table that lists all the features of SAS Activity-Based Management and specifies what features are available to a user based on those two factors, see "Sample Allocation of Capabilities" on page 20.

For examples of allocating capabilities using roles and groups, see "Table of Capabilities and Model Access" on page 25.

Creating Roles

Overview

A role is a set of capabilities. The following picture shows the SAS Activity-Based Management capabilities that can be assigned to a role.



The following table shows the roles that are automatically created for you on installation of SAS Activity-Based Management. The table also shows the capability that each role possesses.

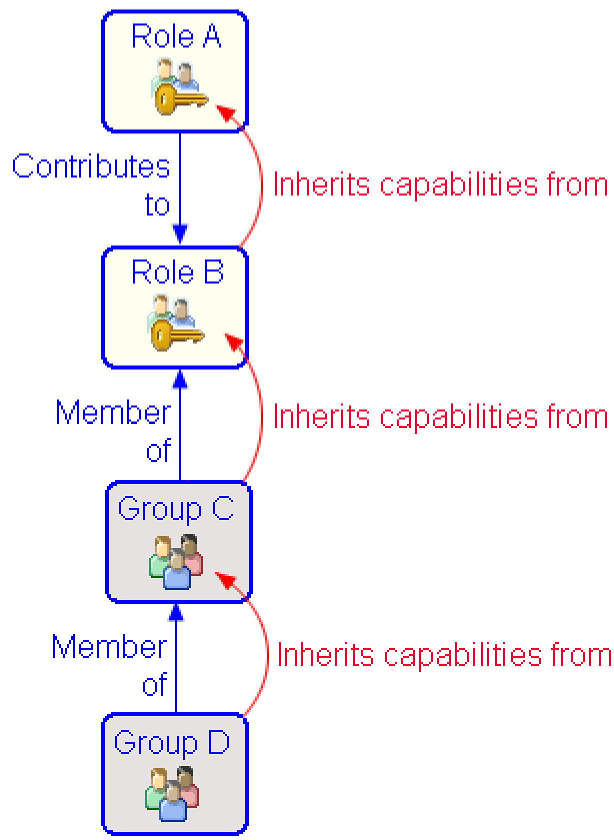
Role	Capabilities
Activity-Based Management: Administration	Administer
Activity-Based Management: Create Model	Create Models *Note:* The capability to create a model includes the capability to create a survey. See Chapter 7, “Creating Surveys,” on page 97.
Activity-Based Management: Take Surveys	Take Surveys See Chapter 11, “Taking a Survey,” on page 139.
Activity-Based Management: View Models	View Models

As an administrator you create roles with one or more capabilities. Then when you create a group, you make the group be a member of a role so that the group inherits capabilities from the role.

For examples of allocating capabilities using roles and groups, see “Table of Capabilities and Model Access” on page 25.

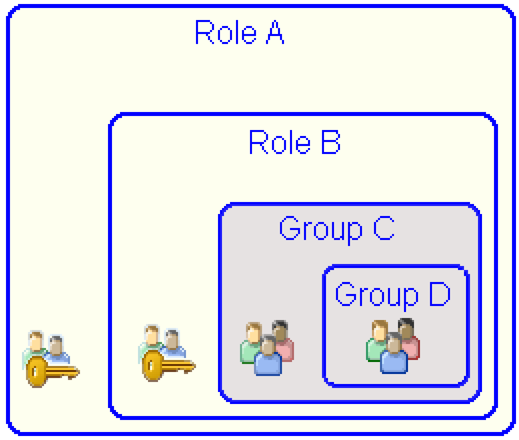
The following picture summarizes the inheritance of capabilities from roles and groups:

- Role B inherits from role A because role A is a contributing role to role B.
- Group C inherits from role B because group C is a member of role B.
- Group D inherits from group C because group D is a member of group C.



Note: The relationship inherits from is transitive. That is if C inherits from B, and B inherits from A, then C inherits from A.

Another way to look at the inheritance of capabilities is as one of inclusion as shown in the following picture which (assumes the Contributes to and Member of relationships in the picture above):

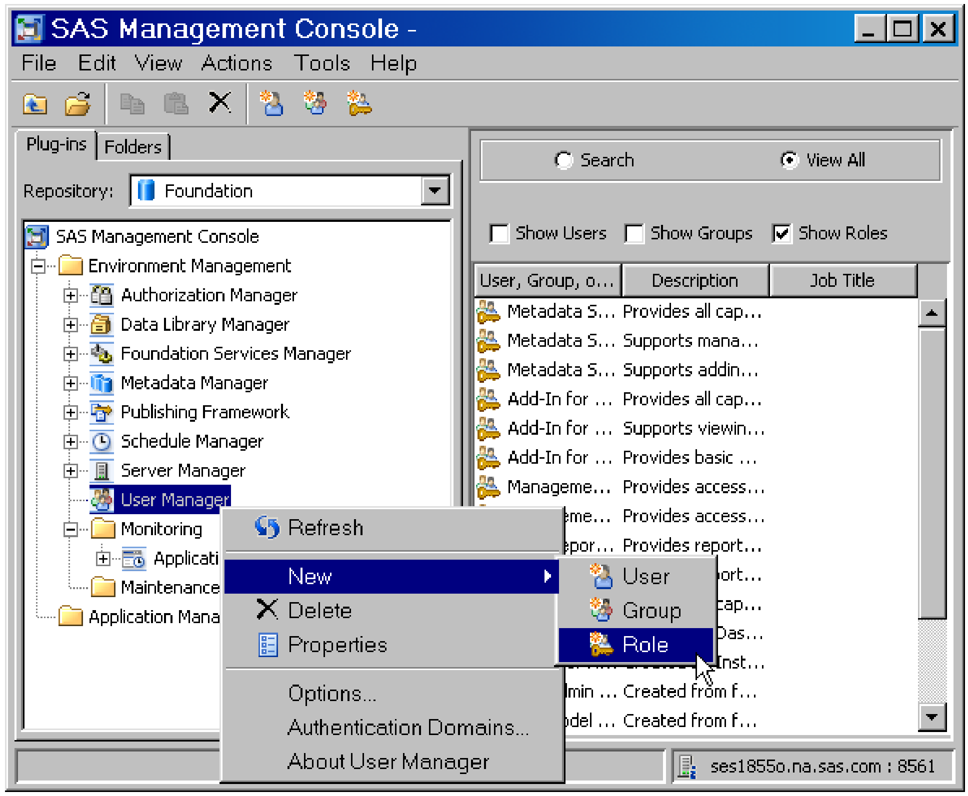


In this picture:

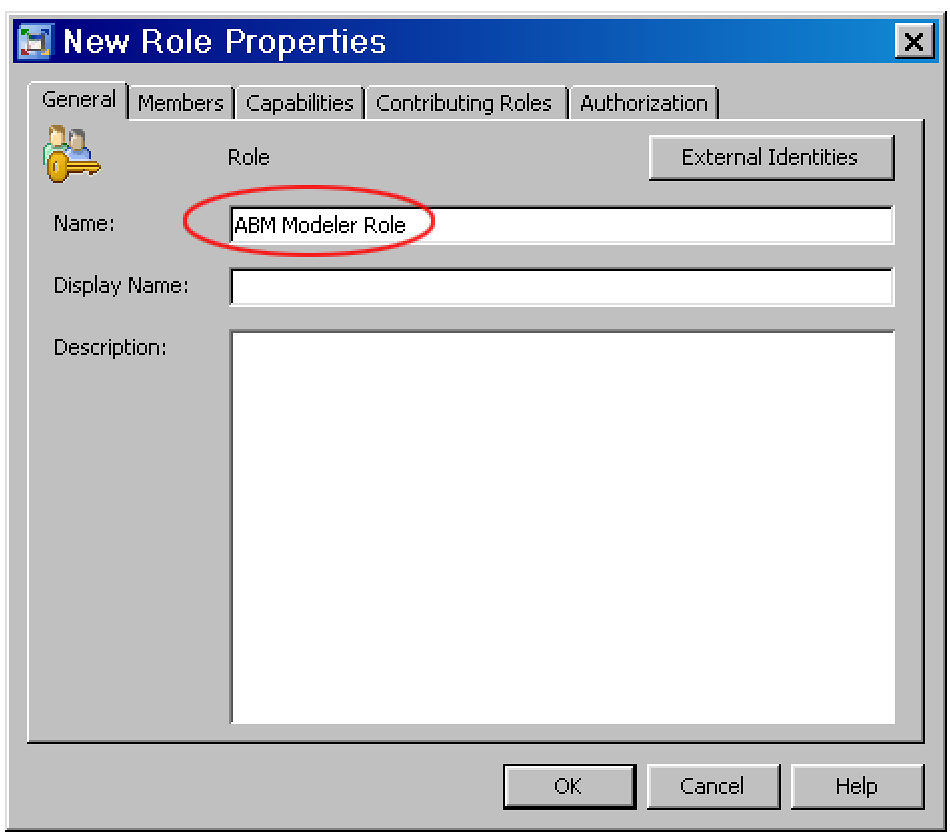
- Users in Role A have only the capabilities of Role A.
- Users in Role B have the capabilities of Role A and Role B.
- Users in Group C have the capabilities of Role A and Role B (and they could have whatever capabilities not pictured that Group C inherits from other roles).
- Users in Group D have the capabilities of Role A and Role B and Group C (and they could have whatever capabilities not pictured that Group D inherits from other roles or groups).

Creating a Role

1. Open SAS Management Console, connecting to your metadata server.
2. Select **User Manager**, and then select **New** ⇨ **Role**.

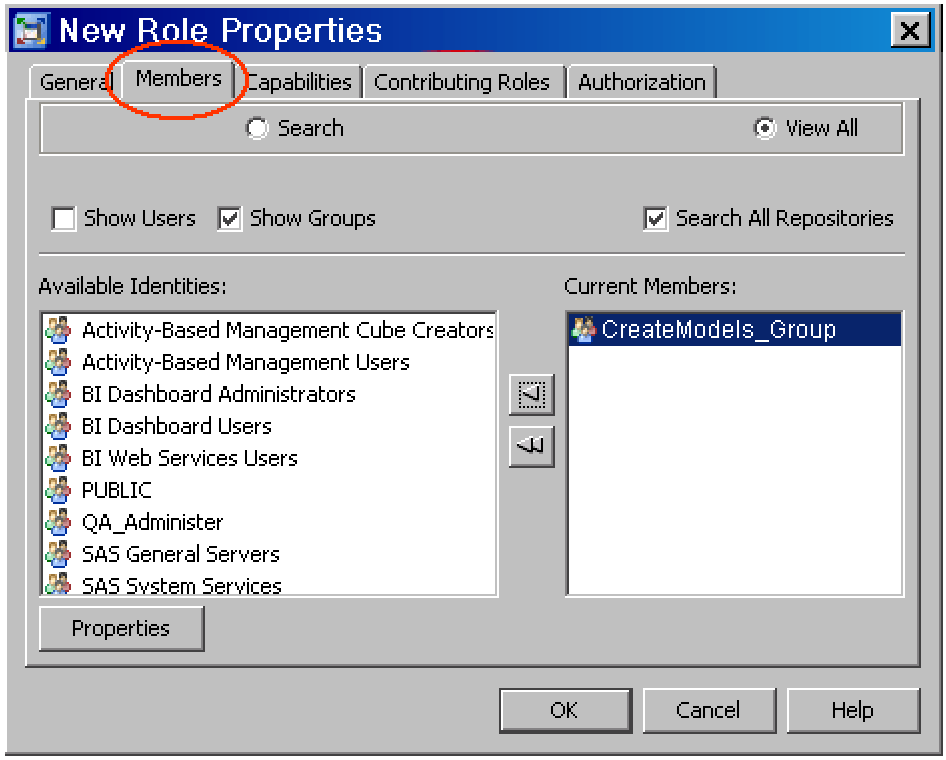


3. Name the role (for example ABM Modeler Role).

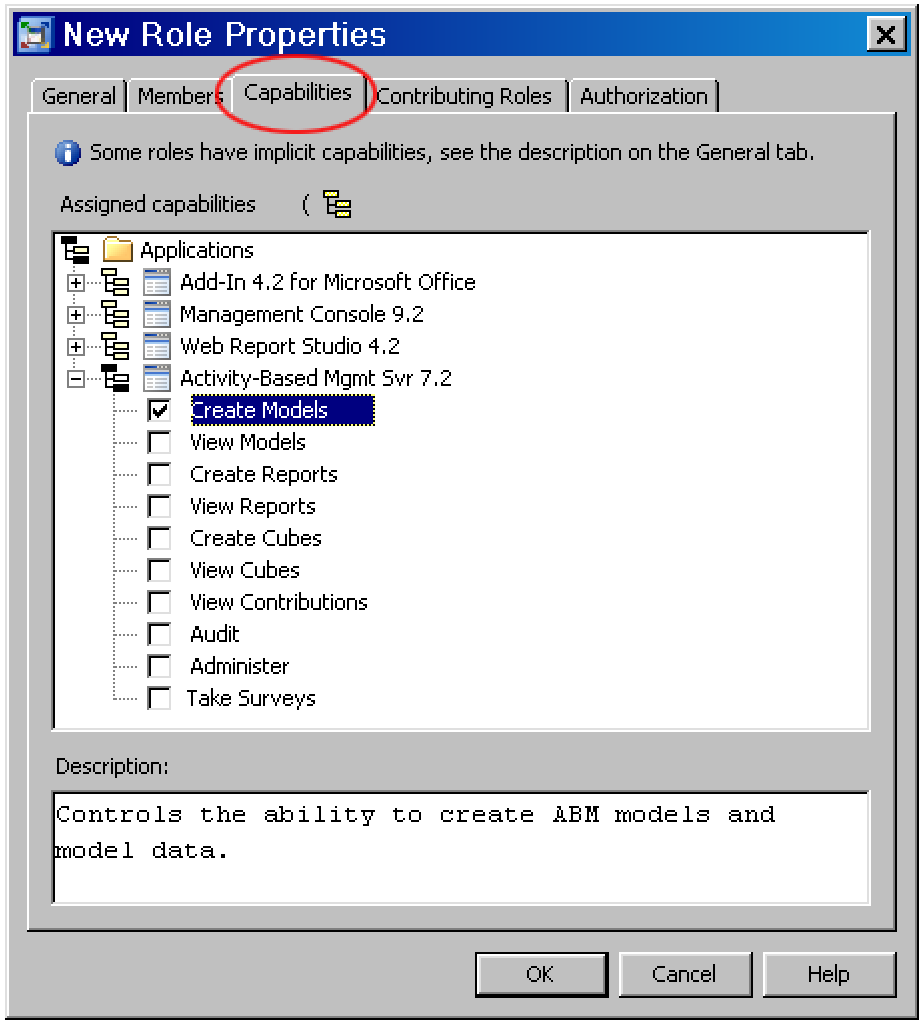


4. Click the **Members** tab and add, as members of this role, the groups that you want to inherit the capabilities of the role.

For example, the following picture shows CreateModels_Group added as a member of the new role. This causes the CreateModels_Group to inherit the capabilities of the role being created.



5. Click the **Capabilities** tab and select the capabilities that you want to assign to this role—and indirectly to any group that is a member of this role.

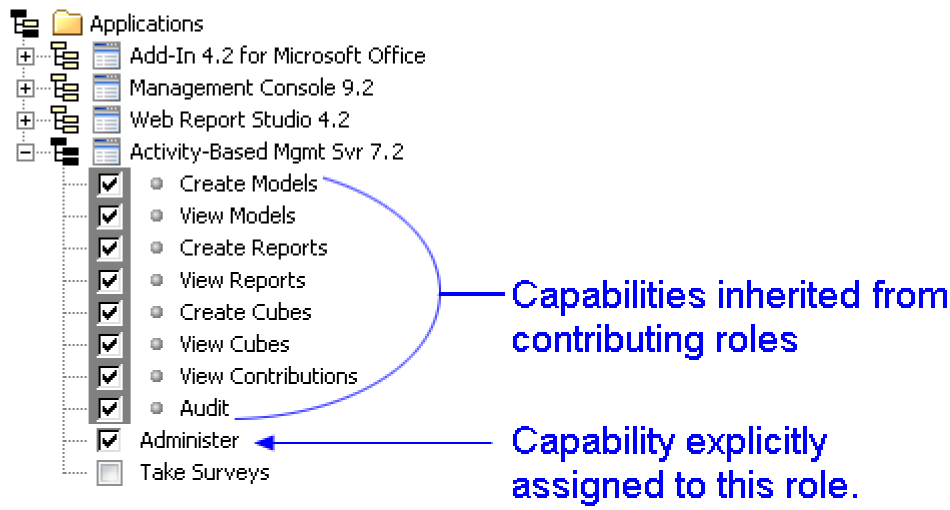


The **Create Models** capability gives full access to a model including model creation and deletion, cube creation and viewing, etc. Plus, it gives users some other abilities not related to a particular model such as creating column layouts and setting up exchange rates. The Create Models capability provides the abilities of a Modeler in previous releases of SAS Activity-Based Management.

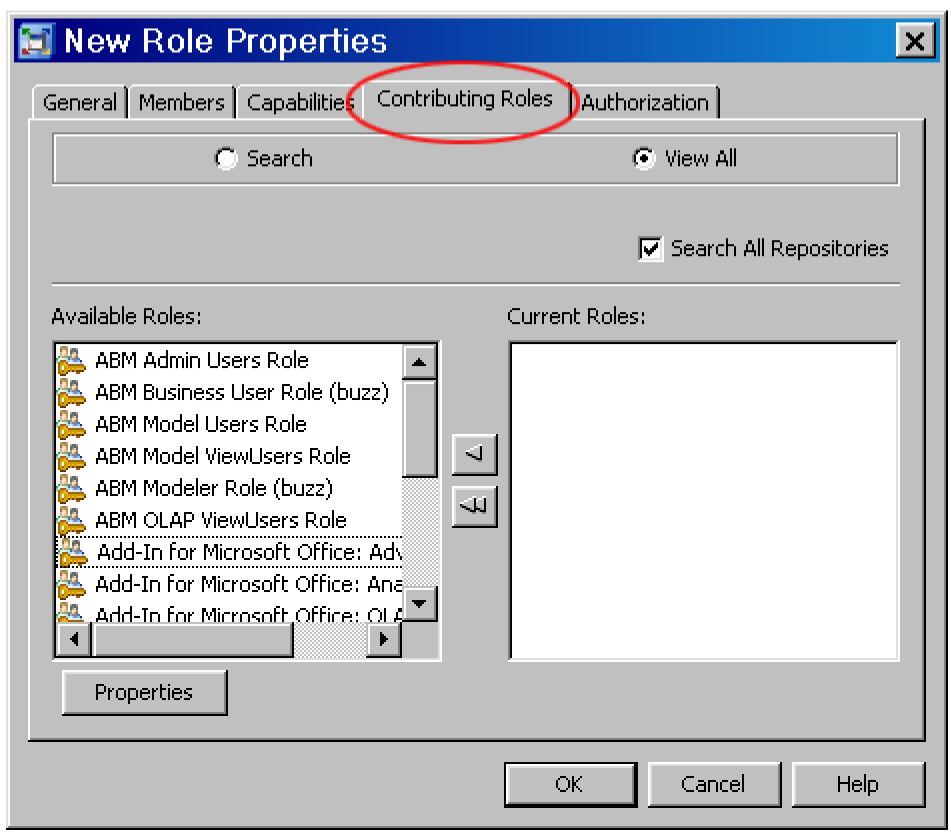
The **View Models** capability gives user the ability to create, view, and publish reports, plus some other abilities. The View Models capability provides the abilities of a Business User in previous releases of SAS Activity-Based Management.

For a table that lists all the features of SAS Activity-Based Management and specifies what features are available to a user based on those two factors, see “Sample Allocation of Capabilities” on page 20.

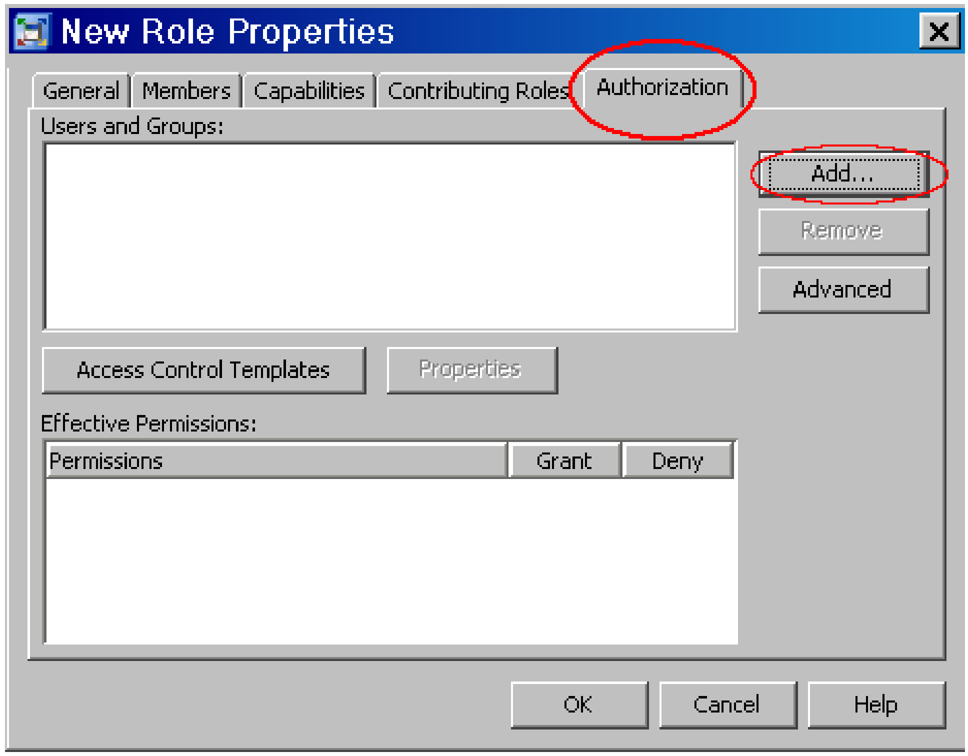
Note: In the **Capabilities** tab of SAS Management Console, the following icon indicates a capability that has been explicitly assigned to a role, whereas the following icon indicates a capability that has been inherited from a contributing role. For example, in the following picture you can see that the first eight capabilities have been inherited from one or more contributing roles, whereas only the last capability (Administer) has been explicitly assigned to the role.



6. Click the **Contributing Roles** tab and add any roles that you want to contribute capabilities to the role being created. This step is optional.



7. Click the **Authorization** tab and add any groups or roles that you want to have authorization to access the role being created. This step is optional.



8. Click **OK** to finish creatiing the role.

Creating Groups

Overview

A group is a set of users who share the same capabilities. It is convenient to create groups before creating users because if groups exist, then when you create a user you can immediately assign the user to one or more groups and thereby determine the user's capabilities. The following table shows the groups that are automatically created on installation of SAS Activity-Based Management. The table also shows the capabilities that each group inherits by being a member of a group or a role..

Group	is a member of:	inherits these ca
Activity-Based Management Administrators	Activity-Based Management Users (group) Activity-Based Management: Administration (role)	Administer
Activity-Based Management Cube Creators	Activity-Based Management Users (group)	none
Activity-Based Management Modelers	Activity-Based Management Users (group) Activity-Based Management: Create Models (role)	Create Models

Group	is a member of:	inherits these ca
Activity-Based Management Survey Takers	Activity-Based Management Users (group) Activity-Based Management: Take Survey (role)	Take Surveys
Activity-Based Management Users	none	none
Activity-Based Management Viewers	Activity-Based Management Users (group) Activity-Based Management: View Models (role)	View Models

Note: In order to access SAS Activity-Based Management, a user must be a member of either:

- the group Activity-Based Management Users
- a group that is a member of the group Activity-Based Management Users. Notice that all the groups that are automatically created on installation (listed in the table above) are members of the Activity-Based Management Users group.

Note: In the case of each group listed above, the capability possessed by the group is inherited from a role, and not from another group.

Note: The Activity-Based Management Cube Creators group does not inherit any capability. For information on giving capabilities to this group to work with SAS OLAP cubes, see "SAS OLAP Cube Creators" on page 23. The Create Cubes capability is not necessary for creating Microsoft Analysis Services cubes.

Note: The "Create Models" capability includes the capability to create a survey—there is no "Create Survey" capability per se.

Creating a group is the second part of a two-step process:

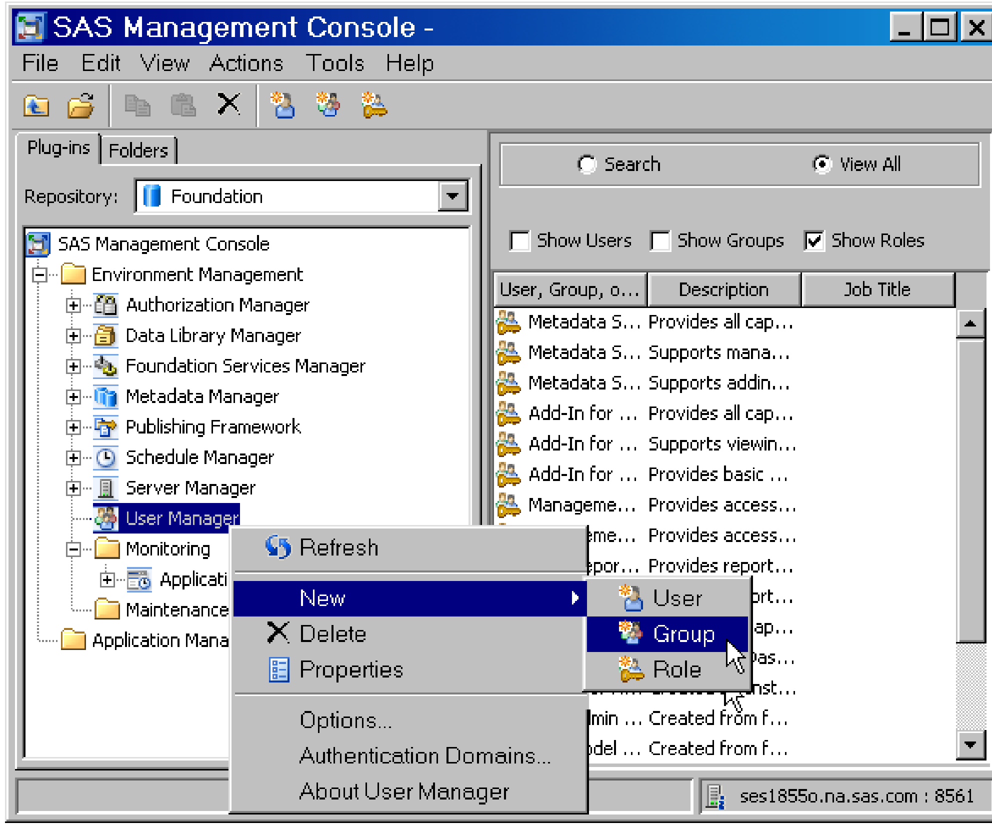
1. Create roles with the capabilities that you want to assign to the group.
2. Create a group and make it a member of the roles that you created. This gives the group the capabilities of the roles.

For examples of allocating capabilities using roles and groups, see "Table of Capabilities and Model Access" on page 25.

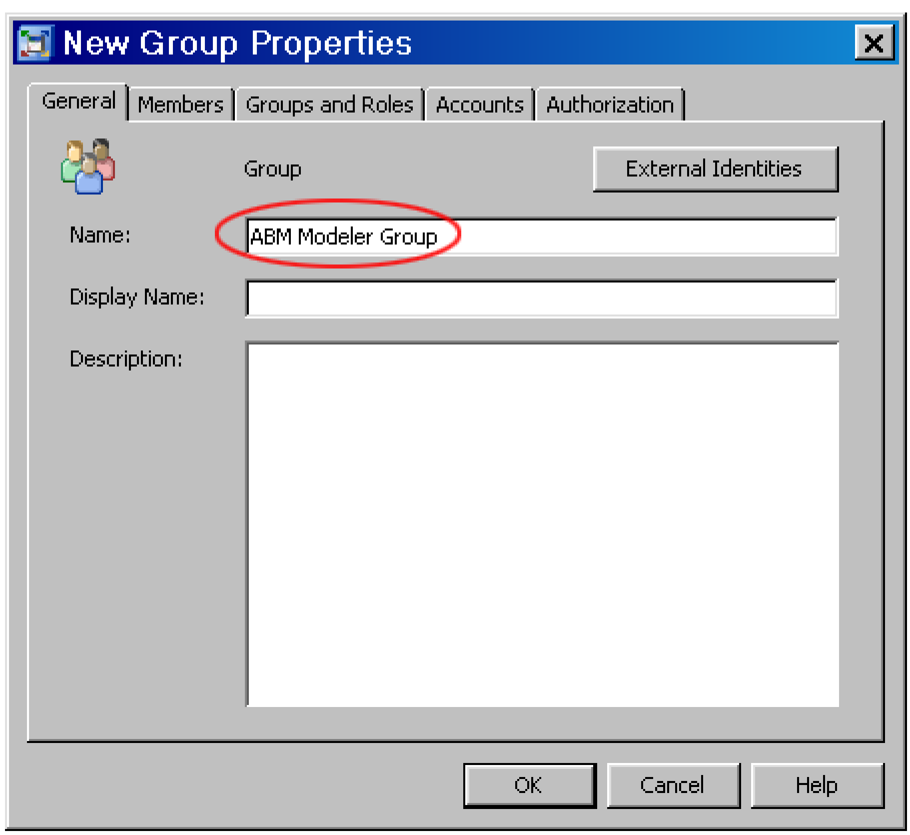
Create a Group

To create a group, do the following:

1. Open SAS Management Console, connecting to your metadata server.
2. Select User Manager, and then select **New** ⇨ **Group**.

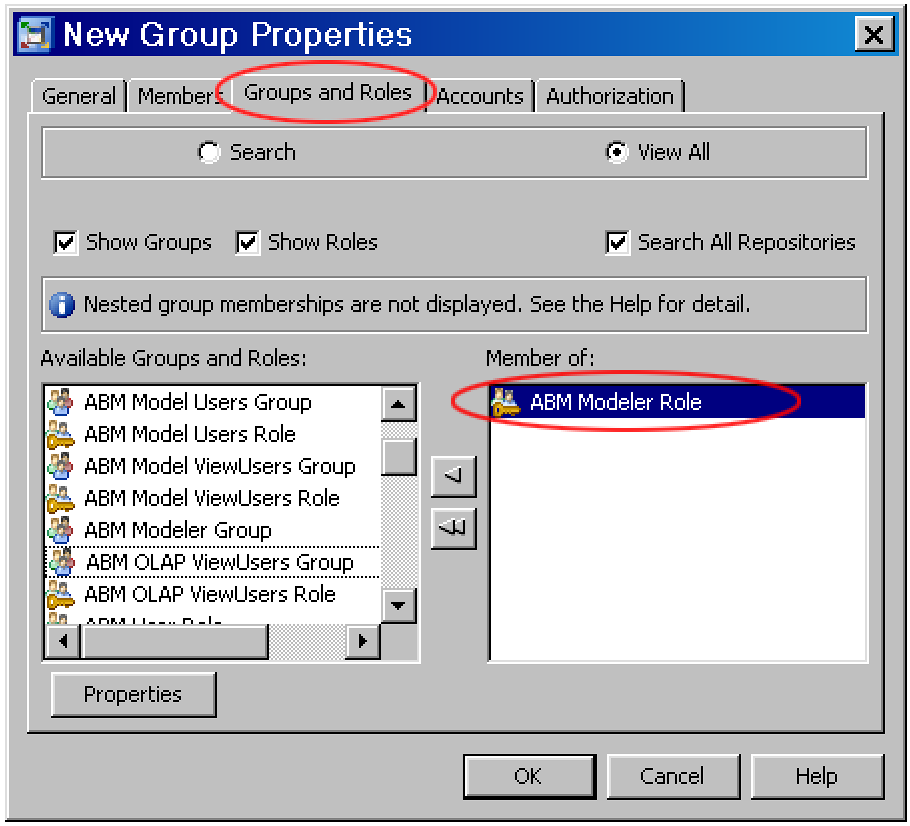


3. Name the group (for example ABM Modeler Group).



4. Click the **Groups and Roles** tab and add the roles whose capabilities you want the group to have.

We say that the group is a member of the role. Such roles contribute their capabilities to the group. Alternatively, we can say that the group inherits its capabilities from the roles.

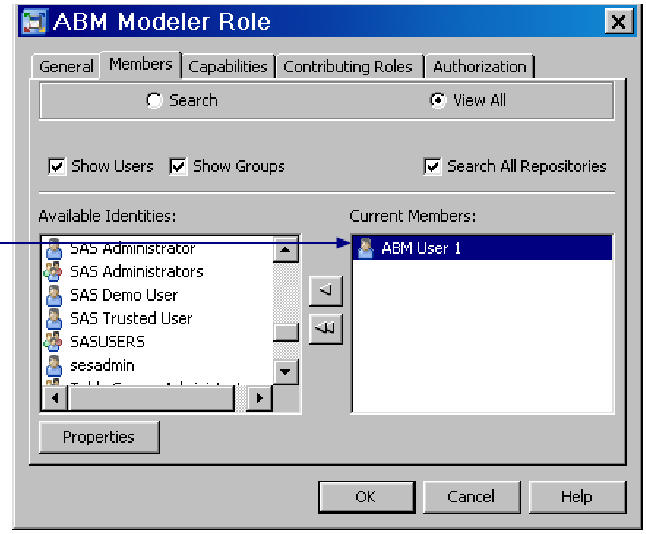


5. Click **OK** to finish creating the group.

Creating Groups versus Creating Roles

You might be aware of the fact that an administrator can give capabilities to a user by making the user a member of a role. The following picture shows ABM User 1 as a member of the ABM Modeler Role. By being a member of this role, ABM User 1 acquires the capabilities of the role in this caseCreate Model.

An administrator can make
a user a member of a role
as well as a member of a group



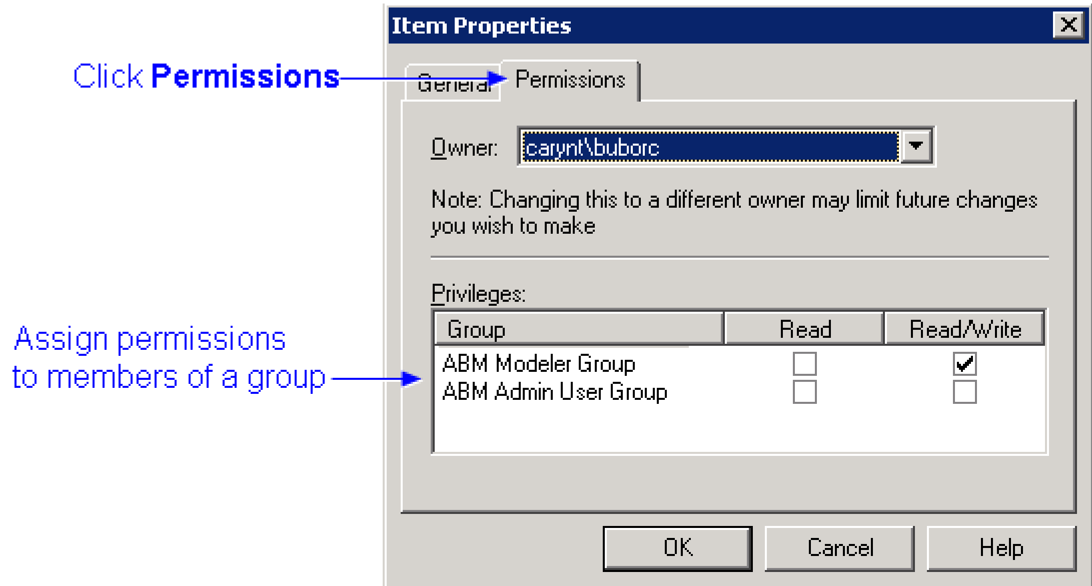
So, you might wonder why an administrator should bother to create a group, ABM Modeler Group, that inherits from the ABM Modeler Role. Why not simply make users a member of the ABM Modeler Role or some other role? The answer is that in order to access SAS Activity-Based Management, a user must be either:

- a member of the group Activity-Based Management Users
- a member of a group that is a member of the group Activity-Based Management Users.

Access to individual workspace items (models, cubes, and so forth) is assigned to groups and not to roles. So, if a user is not a member of a group, then that user cannot access Activity-Based Management objects (such as models) even if the user is a member of a role.

Additionally, the owner of a workspace item can assign read or read/write access to that item. To assign access to a workspace item, the owner does the following:

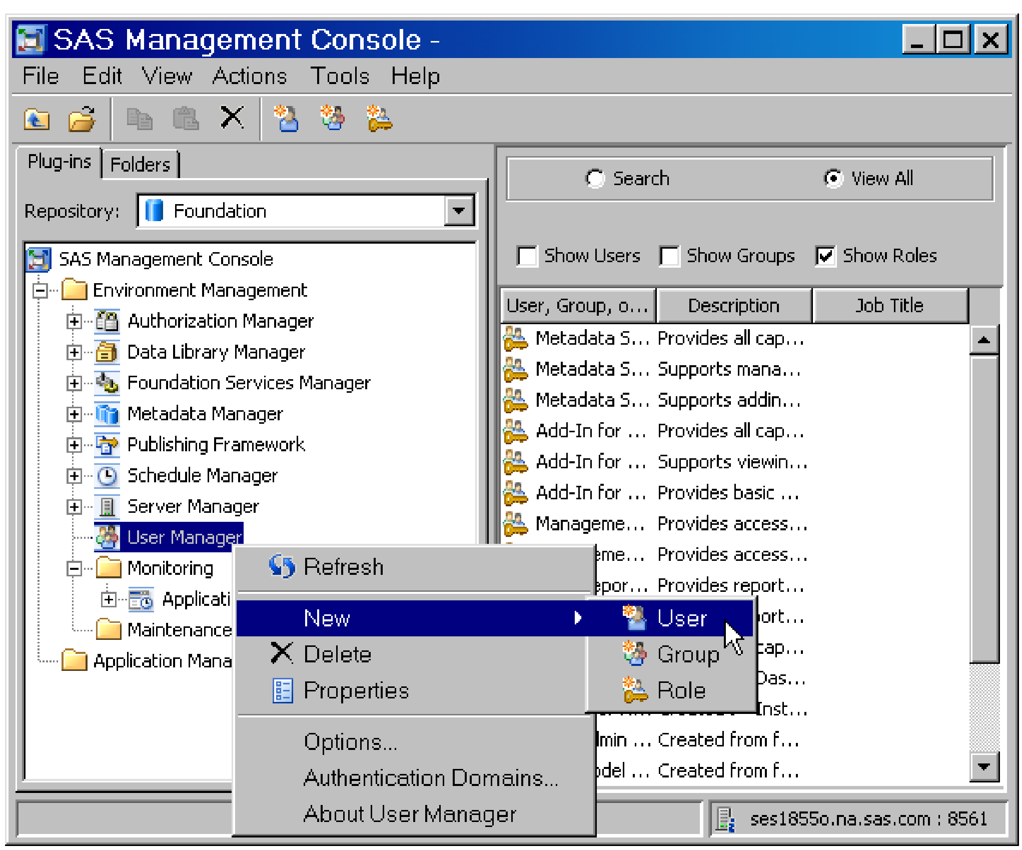
1. Select the workspace item.
2. Right-click and select **Item Properties** (or select **Edit** ⇨ **Item Properties**).
3. Click the **Permissions** tab.
4. Assign Read or Read/Write access to the desired groups. (Only groups are listed not roles.)



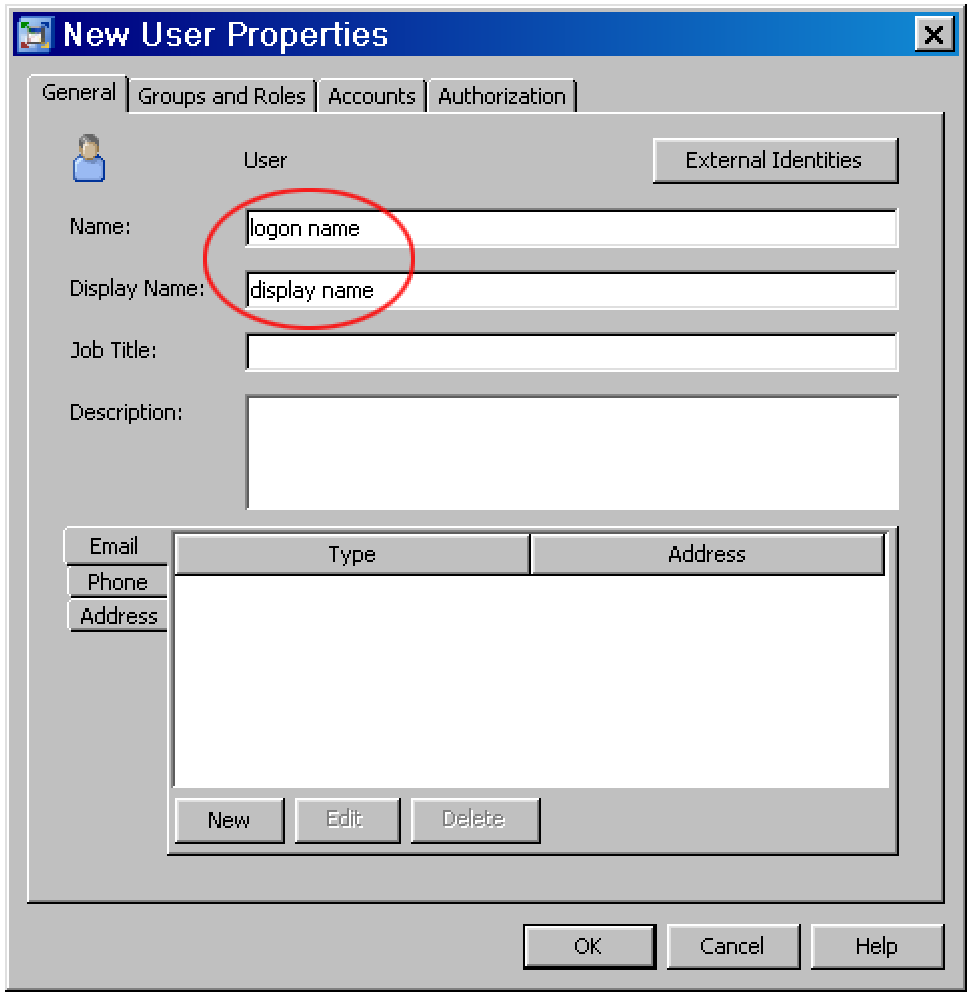
Creating Users

To create a user, do the following:

1. Open SAS Management Console, connecting to your metadata server.
2. Select **User Manager**, and then select **New** ⇨ **User**.

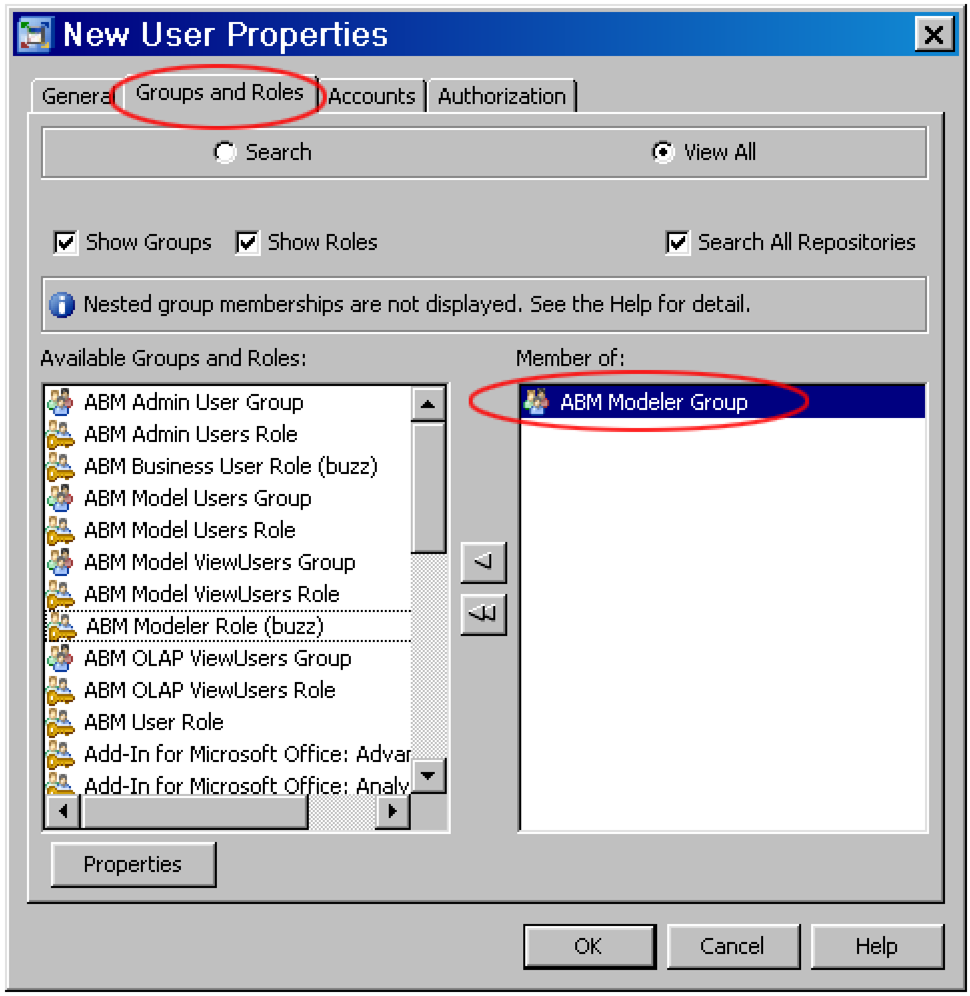


3. Enter the user's Name (which is the user's ABM logon name) and the user Display Name.



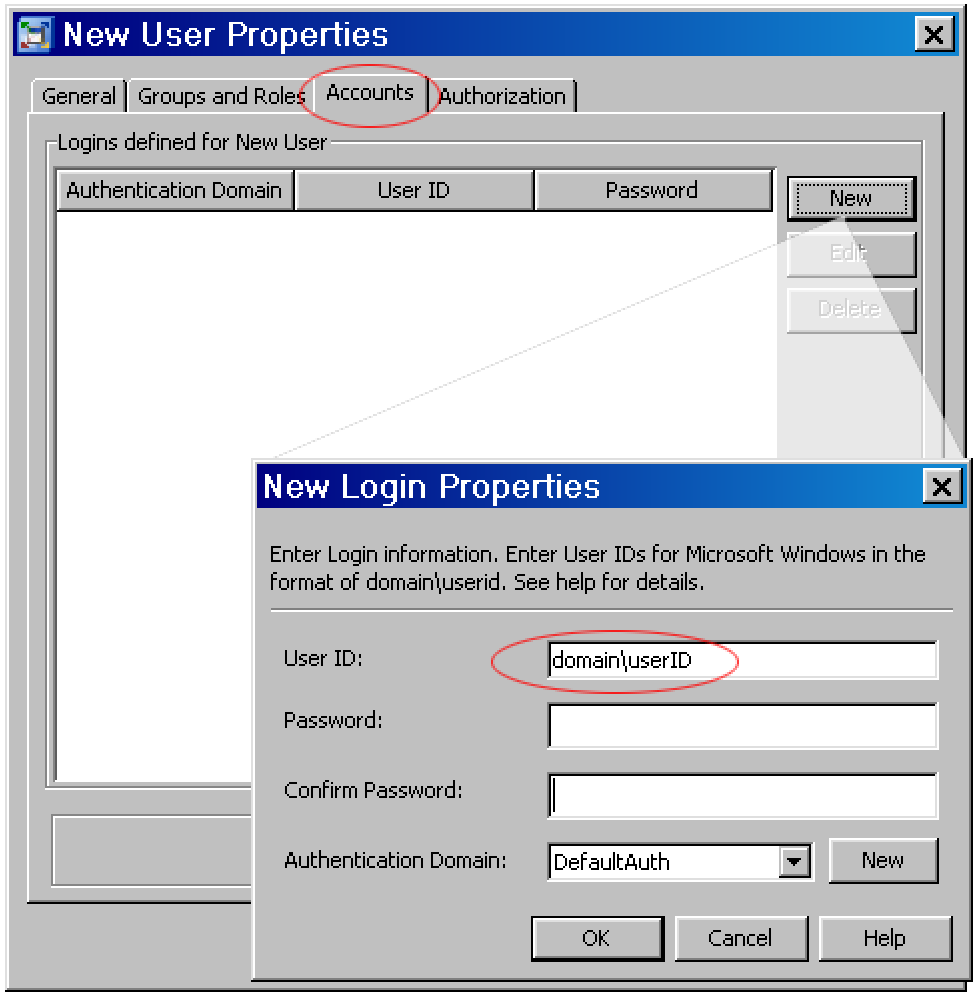
4. Click the **Groups and Roles** tab and add the group or groups of which the user is a member.

   This gives the user all the capabilities of the group. If the user is a member of multiple groups, then the user has the union of the capabilities of the groups of which the user is a member.



5. Click the **Accounts** tab and click **New** to add a new account. Enter the user's domain and user ID.

   You should leave the password blank. When the user logs onto SAS Activity-Based Management, the password that the user enters will be verified against the user's password on the system.



6. Click **OK** twice to finish creating the user.

Sample Allocation of Capabilities

Groups Created During Installation

During installation of SAS Activity-Based Management, the following groups are created automatically:

Group	is a member of:	inherits these ca
Activity-Based Management Administrators	Activity-Based Management Users (group) Activity-Based Management: Administration (role)	Administer
Activity-Based Management Cube Creators	Activity-Based Management Users (group)	none

Group	is a member of:	inherits these ca
Activity-Based Management Modelers	Activity-Based Management Users (group) Activity-Based Management: Create Models (role)	Create Models
Activity-Based Management Survey Takers	Activity-Based Management Users (group) Activity-Based Management: Take Survey (role)	Take Surveys
Activity-Based Management Users	none	none
Activity-Based Management Viewers	Activity-Based Management Users (group) Activity-Based Management: View Models (role)	View Models

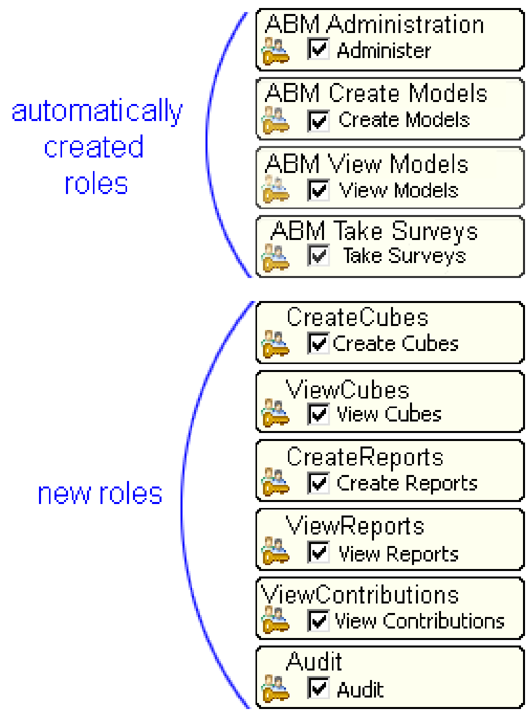
See “Creating Groups” on page 12.

Roles

During installation of SAS Activity-Based Management, the following roles are created automatically.

Role	Capabilities
Activity-Based Management: Administration	Administer
Activity-Based Management: Create Model	Create Models *Note:* The capability to create a model includes the capability to create a survey. See Chapter 7, “Creating Surveys,” on page 97.
Activity-Based Management: Take Surveys	Take Surveys See Chapter 11, “Taking a Survey,” on page 139.
Activity-Based Management: View Models	View Models

As an administrator, you can also create your own roles. One option is to create a separate role for each capability for which there is not a role created automatically. This gives you maximum flexibility in assigning capabilities. Alternatively, you can create roles each of which has multiple capabilities. This gives you less flexibility but makes it easier to assign multiple capabilities to a group.



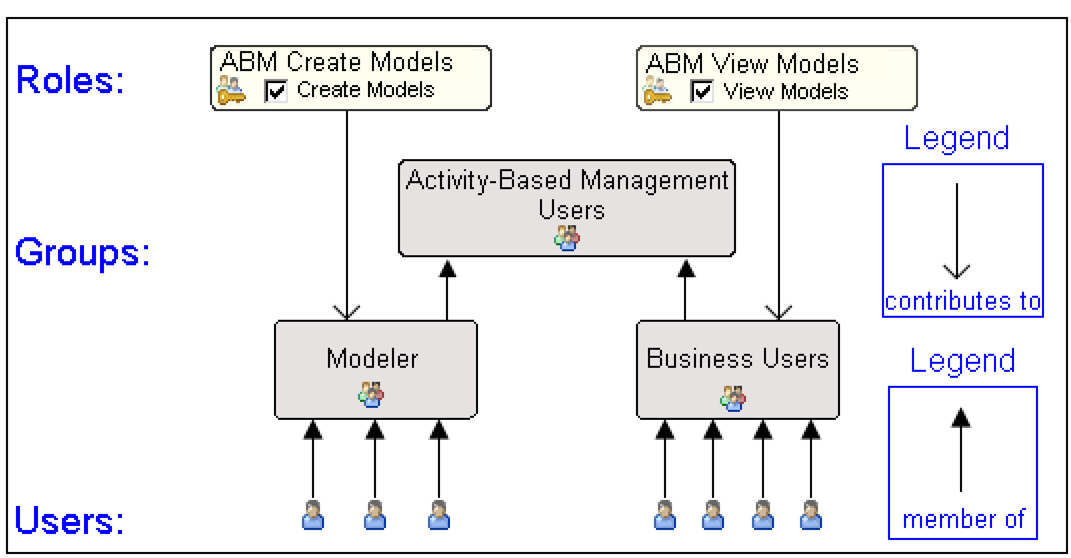
See “Creating Roles” on page 4.

Modelers and Business Users

The following picture shows an example of creating two groups:

- Modeler
- Business Users

Note: The arrangement that is pictured is only a suggestion. It is not required.



The Modeler group inherits the Create Models capability, which gives users in the group full access to a model. With full access, users can create and delete models, create cubes, view cubes, and also perform other tasks such as creating column layouts and setting up exchange rates. The Create Models capability provides the abilities of a Modeler in previous releases of SAS Activity-Based Management.

The Business Users group inherits the View Models capability, which gives users in the group the ability to create, view, and publish reports, plus some other abilities. The View Models capability provides the abilities of a Business User in previous releases of SAS Activity-Based Management.

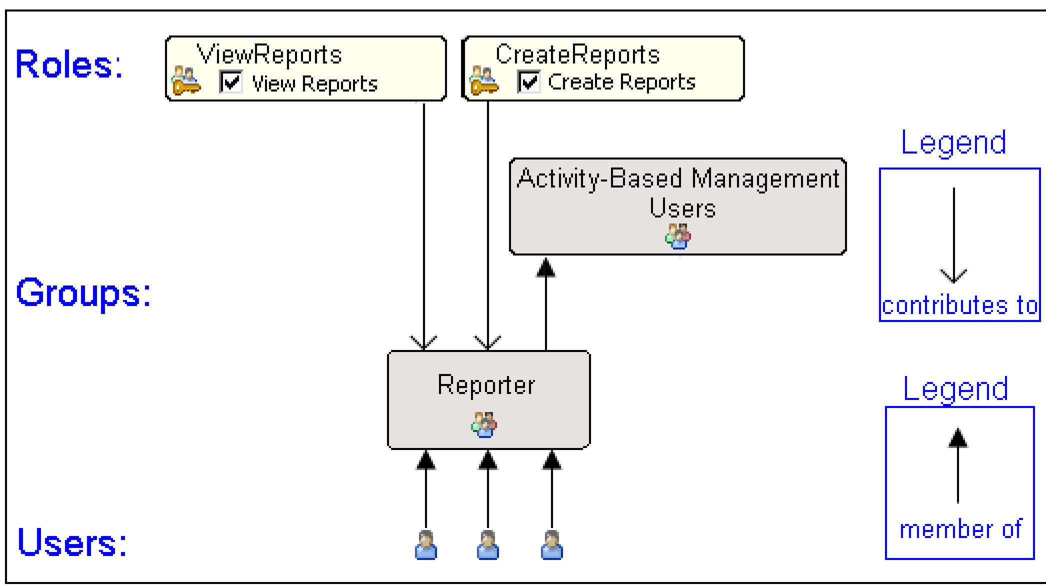
For full details of what access to features the capabilities provide, see "Table of Capabilities and Model Access" on page 25.

Note: The Modeler group and the Business Users group are both members of the Activity-Based Management Users group. This enables every user in the Modeler group and the Business Users group to log onto SAS Activity-Based Management.

Reporters

The following picture shows an example of creating a Reporter group. Members of this group inherit the View Reports and Create Reports capabilities.

Note: The arrangement that is pictured is only a suggestion. It is not required.



Note: The Reporter group is a member of the Activity-Based Management Users group. This enables every user in the Reporter group to log onto SAS Activity-Based Management.

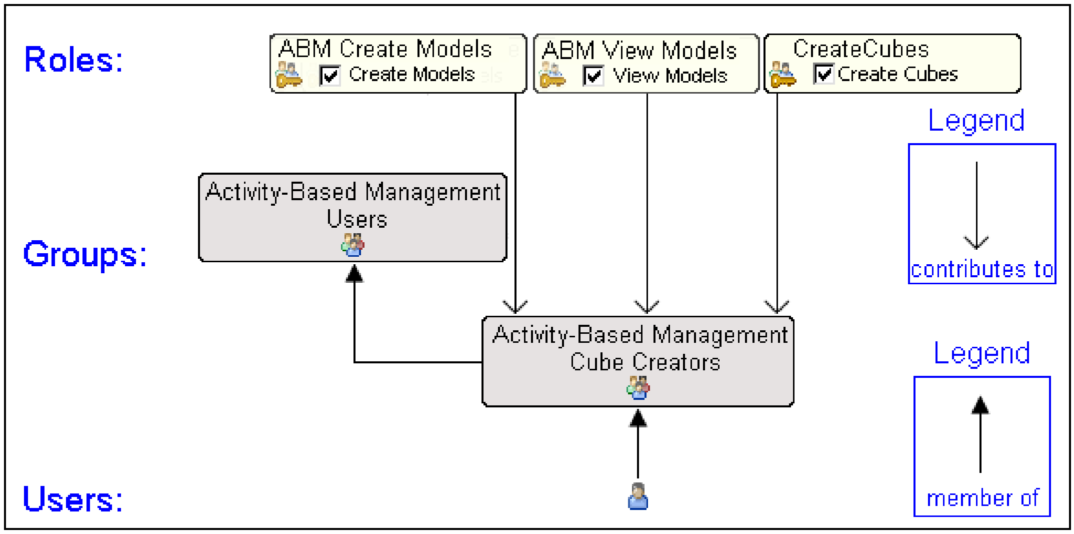
SAS OLAP Cube Creators

A user who wishes to create SAS OLAP cubes must be a member of the Activity-Based Management Cube Creators group. Membership in this group is not required for creating Microsoft Analysis Services cubes.

The Activity-Based Management Cube Creators group should have the following capabilities:

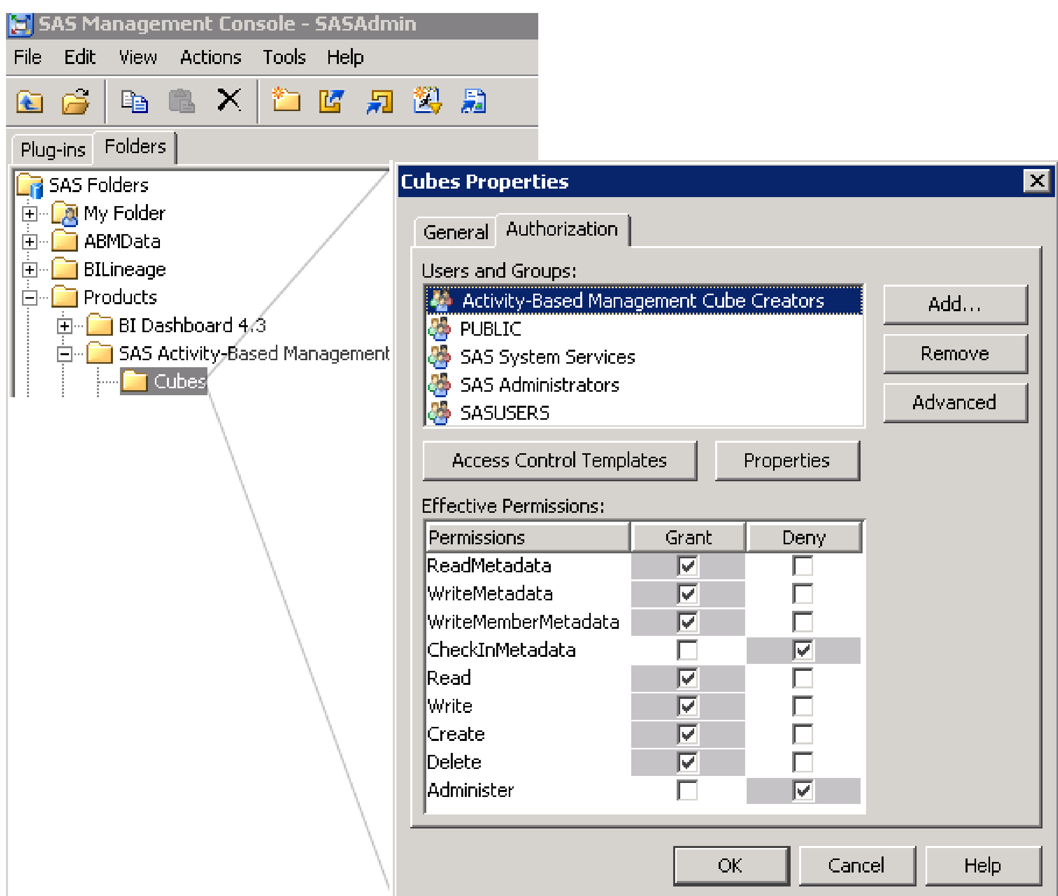
- Create Models
- View Models

- Create Cubes

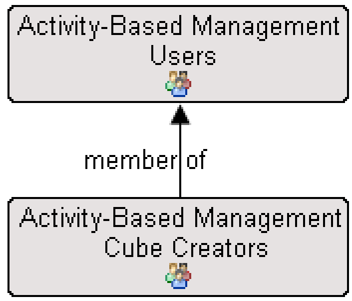


Alternatively, members of the Activity-Based Management Cube Creators group can inherit those capabilities by belonging to other groups with those capabilities.

Note: Members of the Activity-Based Management Cube Creators group inherit the authorizations shown in the following picture to access the Cubes folder for the SAS Activity-Based Management product. These authorizations are required for creating SAS OLAP cubes.



Note: The Activity-Based Management Cube Creators group is a member of the Activity-Based Management Users group. This enables every member of the Activity-Based Management Cube Creators group to access SAS Activity-Based Management.



Survey Takers

The following picture shows an example of creating a Survey Taker group. Members of this group inherit the Take Surveys capability.

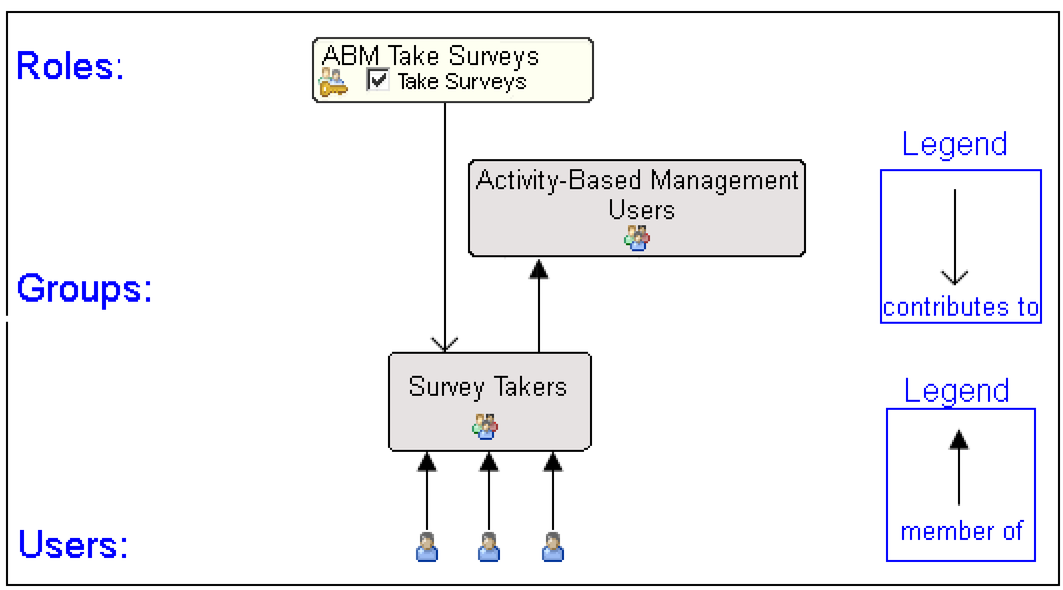


Table of Capabilities and Model Access

About the Table of Capabilities and Model Access

The following table shows how the combination of a capability and access to a model determines a user's access to features of SAS Activity-Based Management.

(link to how to read this table)

Note: All of the following capabilities assume that the user has proper access privileges to the data needed to perform the operation.

Product Feature or Area	Create Models		View Models		Create Reports	View Reports	Create Cubes	View Cubes	View Contributions	Audit	Administer
	Model Read	Model RW	Model Read	Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW
New Model Wizard (*)	✓	✓									
Import Model Data	✓	✓									
Export Model Data	✓	✓	✓	✓							
Export Report Data	✓	✓	✓	✓							
Report Wizard	✓	✓	✓	✓	✓						
Cube Configuration Wizard	✓	✓	✓	✓			✓				
Cube Configuration Import/Export	✓	✓	✓	✓			✓				
New OLAP View Wizard	✓	✓	✓	✓			✓	✓			
OLAP View Import/Export	✓	✓	✓	✓			✓	✓			
Contributions Viewer	✓	✓	✓	✓					✓		

Product Feature or Area	**Create Models**		**View Models**		**Create Reports**	**View Reports**	**Create Cubes**	**View Cubes**	**View Contributions**	**Audit**	**Administer**
	Model Read	**Model RW**	**Model Read**	**Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**
Column layouts Create/ Save/ Import/ Export	✓	✓	✓	✓			✓				
Publish Report (*)	R P	R P D	R P	R P	R P	R					
Insert Published Report (*)	✓	✓	✓	✓	✓						
Publish Model Period/ Scenario		✓	✓	✓							
Publish Behaviors		✓									
Register Metadata		✓									
Model Validation	✓	✓	✓	✓	✓		✓				
Calculate		✓									
Generate Fact Tables		✓					✓				

	Create Models		**View Models**		**Create Reports**	**View Reports**	**Create Cubes**	**View Cubes**	**View Contributions**	**Audit**	**Administer**
Product Feature or Area	**Model Read**	**Model RW**	**Model Read**	**Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**
Generate Cubes		✓					✓				
Audit Log	R	R	R	R						R Purge	
Periods & Scenarios (*)	R W D	R W D	R W	R W	R		R				
Exchange Rates (*)	R W	R W	R	R	R		R				
Dimensions	R	R W D	R	R W	R		R				
Model Properties Dialog	R	R W	R	R	R		R				
Period/ Scenario Associations	R	R W D	R	R W	R		R				
Copy Period/ Scenario Data		✓		✓							
Manage Attributes	R	R W D	R	R W	R		R				
Drivers	R	R W D	R	R W	R		R				

Product Feature or Area	**Create Models**		**View Models**		**Create Reports**	**View Reports**	**Create Cubes**	**View Cubes**	**View Contributions**	**Audit**	**Administer**
	Model Read	**Model RW**	**Model Read**	**Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**	**Model Read or Model RW**
Edit model using Model Views	R	R W D	R	R W	R		R				
Diagnostics	✓	✓	✓	✓	✓		✓				✓
Automation API		✓									
Workspace Items: models, cube configurations, cubes, column layouts, OLAP views, reports, report configurations, etc.											
Create	✓	✓	✓	✓	✓		✓				
Create Folders	✓	✓	✓	✓	✓		✓				
Open (Run, Invoke, View)	Owner X	Owner X	Owner X	Owner X	Owner X1	Owner X2	Owner X3	Owner X4			
Edit	Owner RW	Owner RW	Owner RW	Owner RW	Owner RW1		Owner RW2				
Delete/Rename/Cut (Move)	Owner	Owner	Owner	Owner	Owner		Owner				✓
Item Properties	Owner	Owner	Owner	Owner	Owner		Owner				✓
My Shortcuts	✓	✓	✓	✓	✓		✓				
Operation Summaries(*)	R E D	R E D	R E	R E	R E		R E				

Product Feature or Area	Create Models		View Models		Create Reports	View Reports	Create Cubes	View Cubes	View Contributions	Audit	Administer
	Model Read	Model RW	Model Read	Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW	Model Read or Model RW
Manage Tasks(*)	R D	R D	R	R	R D		R D				R D
User Options(*)	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W

Legend

***** = This facility is not model specific and cannot be controlled on a model-by-model basis

R = Read

W = Write

D = Delete

P = Publish

E = Export

Purge = Purge

Note: A user with Audit capability can purge an audit log without having either Read or Read/Write access to a model.

Owner = The Owner of the Workspace item has the ability

OwnerX = The Owner or any user in an authorized Read or Read/Write Group for the item has the ability.

- **OwnerX1** = The Owner or any user in an authorized Read or Read/Write Group for the item has the ability for Report configurations and Published Reports only. It excludes all other workspace items.
- **OwnerX2** = The Owner or any user in an authorized Read or Read/Write Group for the item has the ability for Published Reports only. It excludes all other workspace items.
- **OwnerX3** = The Owner or any user in an authorized Read or Read/Write Group for the item has the ability for Cube configurations and Saved Cube Views only. It excludes all other workspace items.
- **OwnerX4** = The Owner or any user in an authorized Read or Read/Write Group for the item has the ability for Saved Cube Views only. It excludes all other workspace items.

OwnerRW = The Owner or any user in an authorized Read/Write Group for the item has the ability.

- **OwnerRW1** = The Owner or any user in an authorized Read/Write Group for the item has the ability for Report configurations and Published Reports only. It excludes all other workspace items.
- **OwnerRW2** = The Owner or any user in an authorized Read/Write Group for the item has the ability for Cube Configurations and Saved Cube Views only. It excludes all other workspace items.

Create = This refers to the ability to create folders and links to other workspace items. Creation of workspace items is not done in the workspace. Creation of workspace items is performed via operations on models and model data. For the ability to create workspace items, see the section at the top of the table that enumerates the wizards and dialogs that are used to create workspace items.

Publish Model = Model Viewers with read only access to a model cannot see data in a model unless it is in a published period/scenario.

Workspace items = Ability to perform operations on workspace items means that the user has permission to the workspace item and access to the model data (if any) needed for it. For example, to launch a Cube View, the user must have privileges to the Cube View and to the model it refers to.

Edit = This means to edit workspace items (other than models). For example, to edit an existing report configuration, column layout, etc. and change its contents without changing its name.

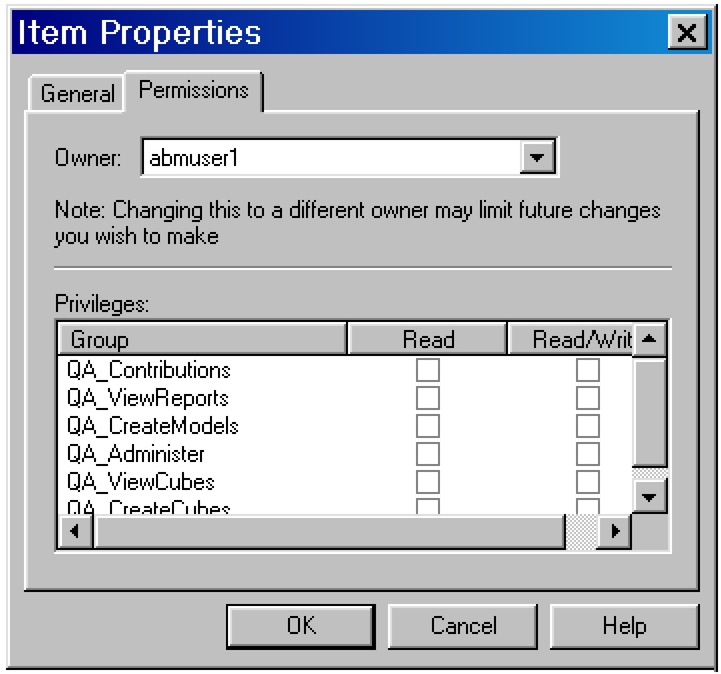
Import Model Data = The permissions on a model are applied if a user overwrites an existing model on import. The user is not allowed to overwrite the model unless the user has Write permission on the model.

User Options = Items that affect how the tool behaves and appears to the user. Example: background colors, number of displayable significant digits. These do not affect the Model data stored on the server in any way.

Model Views = Includes the Resource, Activity, Cost Object, and External Unit modules, Attributes view, Dimensions view, and Drivers view. Editing a model includes all editing tasks within those model views.

My Shortcuts = This includes all operations that can be performed under the My Shortcuts section of the Workspace Manager.

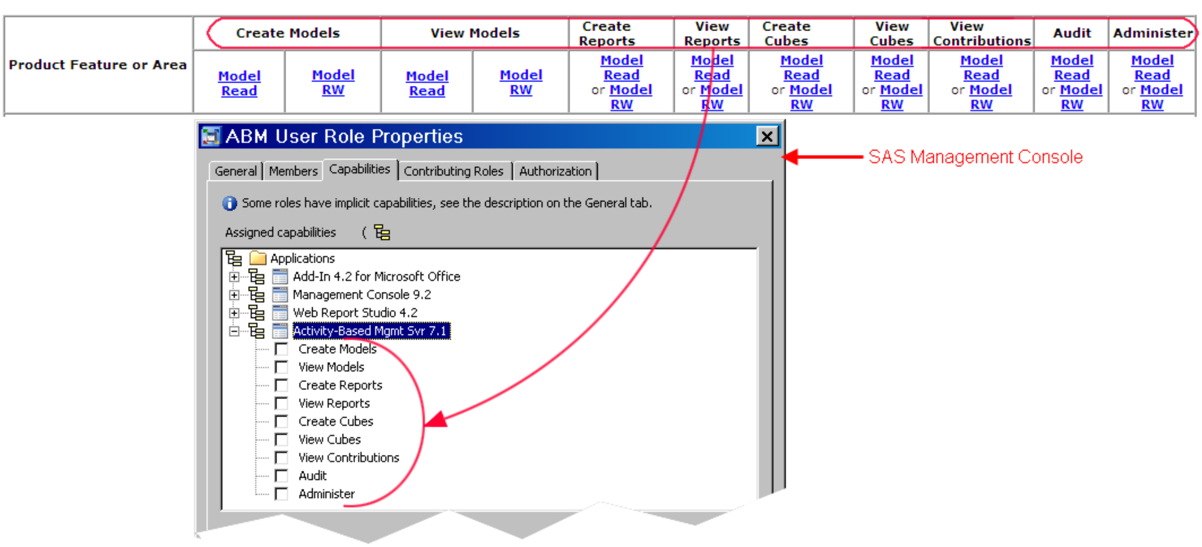
Item Properties = Open the Item Properties dialog box to change the owner of the item, edit its description, or assign permission to the item to groups.



How to Read this Table

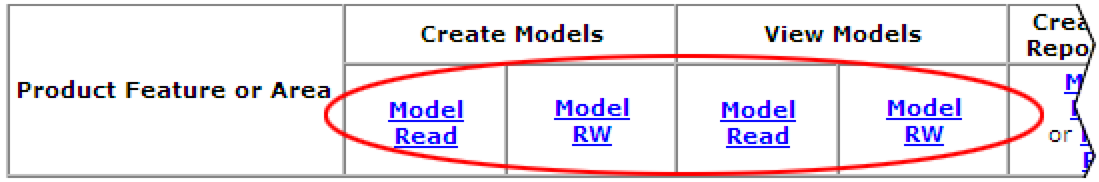
1. Capabilities

The column headings (Create Models, View Models, Create Reports, View Reports, Create Cubes, View Cubes, View Contributions, Audit, and Administer) refer to capabilities that are granted to groups, and roles via SAS Management Console.



2. Model Read and Model RW Access

The column headings Model Read and Model RW refer to the access that a user has to a particular model.

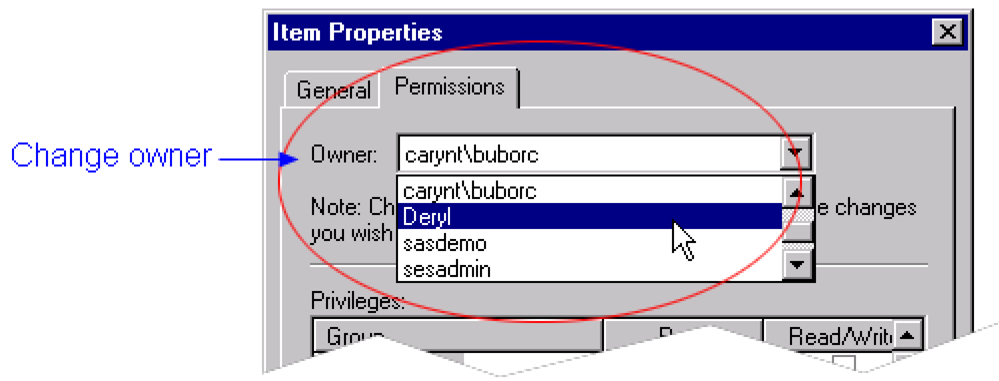


Product Feature or Area	Create Models		View Models		Crea… Repo…
	Model Read	Model RW	Model Read	Model RW	M… or …

A user can acquire the access in either of the following ways:

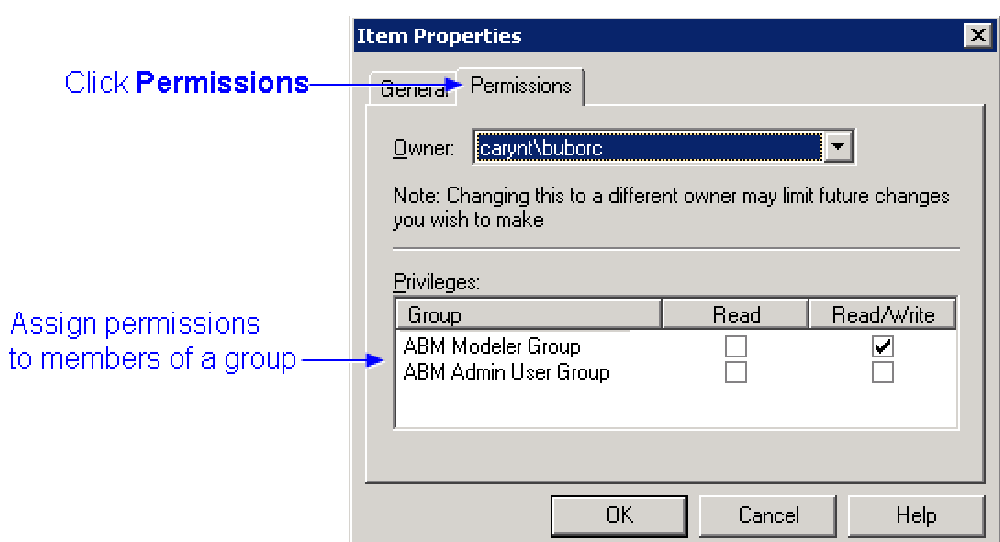
- The owner of a model automatically has Read/Write (RW) access to the model.

  When a model is created, the creator becomes the model owner. However, the owner of a model (or an administrator) can transfer ownership of the model to another user by selecting the model in the Models Workspace and selecting **Edit** ⇨ **Item** Properties and then selecting a new owner from the drop-down list of users.



Note: The drop-down list of potential owners includes those users who have Create capability for the item. For example, you can transfer owership of a model only to those users who have Create Models capability.

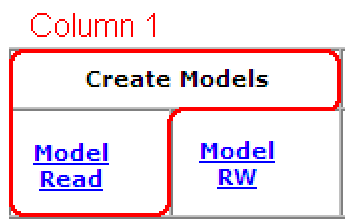
- The owner of a model (or administrator) can grant Read or Read/Write model access to members of a group by selecting the model in the Models Workspace and selecting **Edit** ⇨ **Item Properties**. Then, by checking either Read or Read/Write, the owner grants that model access to all members of the group.



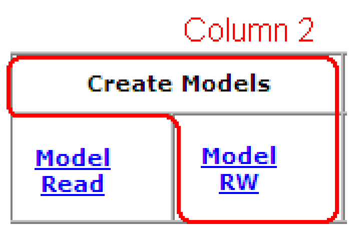
3. Understanding the Columns

The columns in the table refer to a set of users with a combination of a capability and a Read or Read/Write access to a model.

For example column 1 refers to the set of user with Create Models capability and Read access to a model. Because the owner of a model automatically has Read/Write access the model, users with only Read access are those users who are members of a group that has been granted Read access to the model by the model's owner.



Similarly, column 2 refers to the set of users with Create Models capability andRead/ Write access to a model. Users with Read/Write access are the model owner and users who are members of a group that has been granted Read/Write access to the model by the model's owner.



Note: By itself, theCreate Models capability primarily provides the ability to run the New Model Wizard and a few other facilities such as running diagnostics. It is mostly in conjunction with model permissions that the Create Models capability fulfills its potential.

4. Understanding the Rows

Each row in the table refers to an action or facility of SAS Activity-Based Management. For example row 1 refers to the New Model Wizard. Row 2 refers to the action of importing model data.

5. Understanding the Cells

The cells in the table specify either of the following (or both):

- Who specifically has the ability referred to in the row.
- What aspects of the ability does someone have

Who specifically has the ability referred to in the row

The following table lists the cell indicators that specify who specifically has the ability referred to in the row, and what those cell indicators mean.

cell indicator	what it means
✓	A user with both the capability and the Read or Read/Write access in the column heading has the ability.

cell indicator	what it means
Owner	A user has the ability if the user: • has both the capability and the Read or Read/Write access in the column heading, and • is the model owner.
OwnerX	A user has the ability if the user: • has both the capability and the Read or Read/Write access in the column heading, and • is the model owner, or • is a member of an authorized Read or Read/Write group for the item.
OwnerRW	A user has the ability if the user: • has both the capability and the Read or Read/Write access in the column heading, and • is the model owner, or • is a member of an authorized Read/Write group for the item.

What Aspects of the Ability Does Someone Have

The following table lists the cell indicators that specify what aspect of an ability someone has, and what those cell indicators mean.

cell indicator	what it means
R	A user with both the capability and the Read or Read/Write access in the column heading has Read access to the item.
W	A user with both the capability and the Read or Read/Write access in the column heading has Write access to the item.
D	A user with both the capability and the Read or Read/Write access in the column heading has Delete access to the item.
P	A user with both the capability and the Read or Read/Write access in the column heading has Publish access to the item.
Purge	A user with both the capability and the Read or Read/Write access in the column heading has Purge access to the item.

Part 2

Working with Reports

Chapter 2

Predefined Report Templates

Predefined Report Templates

A report template is a file that specifies the layout of a report and the fields of data in a report (but not the report's data itself). When you create a report, you choose a report template.

SAS Activity-Based Management comes with several report templates that provide set formats and permit great flexibility in the amount and type of data to include in a report.

- Destination Furthest
- Dimensional Attribute Cost
- Dimensional Attribute Unit Cost
- Dimensional View
- Driver - Cost and Rate
- Idle Capacity
- Module Hierarchy
- Multi-level Contributions
- Profit Cliff
- Resource Contributions
- Resource Contributions by Attribute
- Resource Contributions Intermediate
- Single-stage Assignment
- Single-stage Contributions
- Unassigned Costs
- Unit Cost

Report Header

Each report has a header that lists pertinent information for that report. All or some of the following information can be listed in a report's header:

Information	Description
Model Name	The model selected for the report
Module	One or more modules selected for the report; each module starts on a new page
Period	The period selected for the report
Scenario	The scenario selected for the report
View Perspective	The dimension selected for the report
Filtered	Indicates that one or more attributes were used to select items for the report; attributes used to select report data are listed on a report's last page

Correlation Report Template

Overview

This report displays the correlation between Cost and Used Quantity.

Information in this report

Report header	See "Report Header" at top of chapter.
ModelId	Model ID
ModelName	Model name
ModuleId	Module Id for ExternalUnit (0), Resource (1), Activity (2), and CostObject (3)
Module	Module name
DriverName	Driver name
AccountId	Account Id. Useful for joining with other tables
AccountRefnum	Account Reference

Report header	See “Report Header” at top of chapter.
AccountName	Account Name
Correlation	The correlation value in the range [-1,+1]
Periods	The number of period/scenarios the account exists in
CostMean	The mean average of cost for account across the period/scenarios
UsedQuantityMean	The mean average of UsedQuantity across the period/scenarios

Destination Furthest Report Template

Overview

This report template traces the costs from an account, through the model, to the accounts that ultimately receive the costs. For example, you can generate a report that lists all the cost objects that receive costs from a specific department in the Resource module. The Destination Furthest report can be run to display two currencies.

Note: Before you run the Destination Furthest report, you must generate the Resource Contributions cube, the Multi-stage Contributions cube is not required for this report.

Information in this report

Report header	See “Report Header” at top of chapter.
Name	Name of the account that contributes or receives costs
Parent	Name of the account where the costs originated
Reference	Account reference
Module	The module of each account that receives costs: Resource (R), Activity (A), or Cost Object (C)
%	Percentage assigned from the parent account to the account that receives costs or the total assigned or the idle capacity
Assigned Cost	Assigned cost or total assigned for the period
Assigned To	Names of accounts that receive costs
Total Assigned	Total of the % (percentages) or total of the Assigned Cost of the accounts that receive costs

Dimensional Attribute Cost Report Template

Overview

This report template lists costs of accounts for the selected dimension attributes, independent of the module hierarchy.

For example, you can look at all accounts with a specific attribute, regardless of their level or association in the module hierarchy. Because the Dimensional Attribute Cost report supplies information about selected dimension attributes, if you do not select any dimension attributes, the report is not generated.

If an account has more than one dimension attribute, that account is listed for each dimension.

Information in this report

Dim	Dimension name
Attribute	Dimension attribute name
Name	Account name
Reference	Account reference
Cost	Cost of the item or sum of the costs of the items that have the selected dimension attribute or, for the listed module, the sum of the costs of all the items that have the selected dimension attributes
BOC Cost	Cost from the bill of costs for the item or sum of the bills of costs of the items that have the selected dimension attribute or, for the listed module, the sum of the bills of costs of the items that have the selected dimension attributes

Dimensional Attribute Unit Cost Report Template

Overview

This report template lists costs and unit costs of accounts for the selected dimension attributes, independent of the module hierarchy. It is similar to the Dimensional Attribute Cost report template, except it has unit cost and output information.

For example, you can look at all accounts with a specific attribute, regardless of their level or association in the module hierarchy. Because the Dimensional Attribute Unit Cost report supplies information about selected dimension attributes, if you do not select any dimension attributes, the report is not generated.

If an account has more than one dimension attribute, that account is listed for each dimension.

Information in this report

Report header	See “Report Header” at top of chapter.
Dim	Dimension name
Attribute	Dimension attribute name
Name	Account name
Reference	Account reference
Output Type	Output type of the account that receives costs; Default, User, or Sold
Output Qty	The quantity can be one of the following: • Default: Quantity is 1 • User: The user-entered output quantity • Sold: The quantity sold
Cost	Cost of the item or sum of the costs of the items that have the selected dimension attribute or, for the listed module, the sum of the costs of the items that have the selected dimension attributes
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty

Dimensional View Report Template

Overview

This report template lists revenue, cost, profit, and unit cost for the selected dimension.

By selecting dimensions, you build a hierarchy to report the profitability for those dimensions. For example, you can build a dynamic hierarchy that lists each region, each sales person in each region, each sales person's customers, and each customer's product purchases. You can select up to 10 dimensions.

In the Report Wizard, in the Select dimensions step, order the dimensions so that in the Selected box, the dimensions are ordered with the dimension at the top of the box that represents the top of the dimension hierarchy.

Only accounts are listed in this report.

Information in this report

Report header	See “Report Header” at top of chapter.
Name	Dimension attribute name
Reference	Dimension attribute reference

Report header	See “Report Header” at top of chapter.
Dim Depth	Level in the dimension hierarchy; the first dimension you selected in the Report Wizard is Dim Depth 1; selection order determines Dim Depth
Revenue	The roll up, through the dimensional hierarchy, of user-entered revenue for individual accounts
Cost	Amount of money consumed to support the intersection of the dimensions in the dimensional view; calculates a total of all the accounts with cost elements tagged with the dimension attribute
Profit	Revenue minus Cost
Profit %	Profit divided by Revenue
Sold Qty	For each dimension attribute, the sum of the sold quantities in the Sales table; each higher level Sold Qty is the sum of its sub-items
Unit Revenue	Revenue divided by Sold Qty
Unit Cost	Cost divided by Sold Qty
Unit Profit	Profit divided by Sold Qty

- Tip Print this report in landscape orientation.

Driver - Cost and Rate Report Template

Overview

This report template lists each driver used in the selected module and the accounts that use that driver.

Information in this report

Report header	See “Report Header” at top of chapter.
Driver Name	Driver name, followed by a list of accounts that use the driver
Reference	Driver reference (and, for a calculated driver, the formula); Reference contains the Driver Type (Basic, Weighted, calculated) and Driver Qty Type (Unique or Shared)
Cost	Account cost or total cost of the accounts that use the driver
User Total Driver Qty	User-entered total driver quantity for an account
Total Driver Qty	Total driver quantity for the account that the system calculated by adding the Basic driver quantities and Weighted driver quantities
Driver Rate	Calculated by dividing an account's Cost by its Total Driver Qty

Report header	See "Report Header" at top of chapter.
Driver Type	Type of driver: Basic or Weighted
Driver Qty Type	Type of driver quantity: Shared, Unique, or Unknown

Idle Capacity Report Template

This report template lists all of the accounts that have an idle capacity greater than zero. This information can help users locate accounts whose costs have not been fully assigned and accounts with total user-entered driver quantities that do not equal total assigned driver quantities.

Information in this report

Report header	See "Report Header" at top of chapter.
Name	Module name, period, scenario, module, or account name
Driver	Driver name
Reference	Account reference
Idle Percent	Driver percentage that has not been assigned from this account; calculated by subtracting the sum of the account's driver percentages from 100 percent
Idle Qty	Calculated number between the total user-entered driver quantity and the total calculated driver quantity; for a percentage driver, because the default total driver quantity is 100, the Idle Qty is a calculated remainder from 100, minus the user-entered driver quantities
Idle Cost	Cost that has not been assigned for the percentage, account, or module; calculated by subtracting the sum of the account's assigned costs from its total cost
Grand Total	Sum of all <Scenario> Idle Cost for all accounts for the entire model

Module Hierarchy Report Template

Overview

This report template lists information about accounts and their contents for each module. This information can help users validate and analyze a model or document the full detail in a model. The complete hierarchy of any section of the model is displayed with all of the centers, accounts, and cost elements and their costs.

- Tip When the Module Hierarchy report is run to display two currencies, the unit cost is replaced with the cost in the alternative currency.

Information in this report

Report header	See "Report Header" at top of chapter.
Level	Level of item being displayed as a contributor; resource accounts are defined as level 1 contributions and cost elements are defined as level 2 contributions because they roll up to the account cost
Name	Name of the account (or cost element) that contributes or receives costs
Reference	Account reference
Terminal Dimension	The dimensional value of the structural dimension displayed in the column layout
Type	The type of the item being displayed, such as Root, Rollup Account, Account, Entered Cost Element
Cost	Cost of the item or sum of the costs of the items that have the selected dimension attribute or, for the listed module, the sum of the costs of the items that have the selected dimension attributes
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty

Multi-level Contributions Report Template

Overview

This report template lists the contributing accounts and their costs at every level back to the original contributing accounts.

In the Report Wizard, if you choose to show cost elements, the Multi-level Contributions report also lists the cost elements of the original contributing accounts and is called the Multi-level Contributions With Cost Elements report.

Note: A maximum of 10 previous contributing accounts are listed. If your model has more contributing accounts, the results will be incomplete or wrong.

The Multi-level Contributions report does not correctly present accounts in a reciprocal system because it follows each contributing cost and ends up in an assignment loop that exceeds the supported contribution depth of 10 accounts.

The Final Account is the account that receives all the costs.

Information in this report

Report header	**See “Report Header” at top of chapter.**
Source Depth	Number of steps through the assignment path from the final destination
Name	Name of the account or cost element
Reference	Reference of the account or cost element
Module	Module of the account (or cost element or bill of costs) that contributes costs • Assigned cost; values are Resource module (R), Activity module (A), or Cost Object module (C) • Entered costs; value is CEE (Cost Element Entered) • Bills of costs; value is Ext (External Unit)
Cost	Cost of the account or cost element
%	Percentage of the account; calculated by dividing the cost of the account that contributes costs by the cost of the account that receives costs
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty
Output Type	Output type of the account that receives costs; Default, User, or Sold; the quantity can be one of the following: • Default: Quantity is 1 • User: The user-entered output quantity • Sold: The quantity sold

Profit Cliff Report Template

Overview

This report template generates a graph of the cumulative profit for one dimension. This information can help users determine the break-even point of items and unprofitable items.

Only accounts are listed in the Profit Cliff report.

Information in this report

Report header	**See “Report Header” at top of chapter.**
Record	Sorted record number graphed on the x-axis that matches the Profit Cliff graphic to the data table detail; all records are sorted by Profit. Cumulative Profit is graphed on the y-axis

Report header	**See "Report Header" at top of chapter.**
Name	Dimension attribute name
Reference	Dimension attribute reference
Revenue	The roll up, through the dimensional hierarchy, of user-entered revenue for individual accounts
Cost	Amount of money consumed to support the selected dimension attribute; displays a total of all the cost elements in the accounts with the dimension attribute
Profit	Revenue minus Cost; Profit only applies to accounts with a dimension attribute of the selected dimension
Sold Qty	Total quantity sold for the dimension attribute
Cumulative Profit	Cumulative profit across all dimension attributes
Grand Total	Total Revenue, Cost, Profit, and Sold Qty for the dimension attributes in the selected dimension

Resource Contributions Report Template

Overview

This report template lists the first accounts in which costs are entered into the model. And, it displays how those costs are ultimately assigned to the final cost object in the model. The Resource Contributions report is useful for analyzing the beginning and end of the assignment path.

In the Report Wizard, if you choose to show cost elements, the Resource Contributions report also lists the cost elements of the original contributing accounts and is called the Resource Contributions with Cost Elements report.

This report lists the contributing costs from reciprocal allocations.

- Before you run this report, you must generate the Resource Contributions cube.

Information in this report

Report header	**See "Report Header" at top of chapter.**
Level	Level of item being displayed as a contributor; resource accounts are defined as level 1 contributions and cost elements are defined as level 2 contributions because they roll up to the account cost
Name	Name of the account (or cost element) that contributes or receives costs
Reference	Account reference

Report header	See “Report Header” at top of chapter.
Module	Module of the account (or cost element or bill of costs) that contributes costs • Assigned costs; values are Resource module (R), Activity module (A), or Cost Object module (C) • Entered costs; value is CEE (Cost Element Entered) • Bills of costs; value is Ext (external items)
Cost	Cost of the account or cost element
%	Percentage of the cost contributed by account (or cost element)
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty
Output	Output type of the account that receives costs; Default, User, or Sold; the quantity can be one of the following: • Default: Quantity is 1 • User: The user-entered output quantity • Sold: The quantity sold
Total Cost	Total costs from the accounts that contribute costs

Resource Contributions by Attribute Report Template

Overview

This report template lists the first accounts in which costs are entered into the model. And, it displays how those costs are ultimately assigned to the final cost object in the model. The Resource Contributions by Attribute report is identical to the Resource Contributions report, but adds the ability to see resource dimension member values. This report is useful for analyzing the beginning and end of the assignment path.

The Resource Contributions by Attribute report lists the contributing costs from reciprocal allocations.

- Before you run this report, you must generate all cubes

Information in this report

Report header	See “Report Header” at top of chapter.
Name	Name of the account (or cost element) that contributes or receives costs
Reference	Account reference
Source Dimension Member	The dimension member to which the account that gives the costs belongs

Report header	See "Report Header" at top of chapter.
Cost	Cost of the account or cost element
%	Percentage of the cost contributed by account (or cost element)
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty

Resource Contributions Intermediate Report Template

This report template enables you to export a report of the resources contributing to any account in the model. Unlike the Resource Contributions report, which supports only the analysis of the cost contribution to the final cost objects, the Resource Contributions Intermediate report enables you to run the export report for an activity account or a non-final cost object. This gives you more flexibility in defined export content.

- This report cannot be run to display a report result in SAS Activity-Based Management.

Single-stage Assignments Report Template

Overview

This report template lists all available information about accounts and the costs that flow from one account to another. This information can help users understand how different (or unrelated) accounts impact the costs of other accounts.

Information in this report

Report header	See "Report Header" at top of chapter.
Name	Name of the account (or cost element) that contributes or receives costs
Reference	Account reference
Module	The module of each account that receives costs: Resource (R), Activity (A), or Cost Object (C)
%	Percentage assigned from the parent account to the account that receives costs, idle capacity or total assigned
Driver Qty	Total driver quantity, account driver quantity, total assigned driver quantity, or idle capacity driver quantity
Assigned Cost	Cost of the account that contributes costs, assigned cost for account that receives costs, idle capacity, or total assigned

Report header	**See “Report Header” at top of chapter.**
Driver	Driver name specified for the account that contributes costs (and, for a calculated driver, the formula); in the Reference column, the driver type is Shared or Unique
Total Driver Qty	Total driver quantity for the account that contributes costs that the system calculated by adding the Basic driver quantities and Weighted driver quantities for the accounts that receive costs
User Total Driver Qty	Only listed when an account that contributes costs has a user-entered Total Driver Qty
Assigned To	Names of accounts that receive costs
Total Assigned	Total of the % (percentages), driver quantities, and assigned costs of the accounts that receive costs
Idle Capacity	Total Assigned subtracted from the Total Driver Qty

Single-stage Contributions Report Template

Overview

This report template lists cost information about accounts and the costs that contribute to them.

In the Single-stage Contributions report, you see the contributing unit cost and the contribution of each unit to an account. This information can help users review all the costs that contribute to an account. It lists all contributing items: entered costs, assigned costs, costs from bills of costs, and costs assigned.

Information in this report

Report header	**See “Report Header” at top of chapter.**
Name	Name of the account (or cost element) that contributes costs or receives costs
Contributions	Name of the account where the costs originated with an element type of Assigned, Entered, or Bills of Costs
Reference	Account reference
Module	Module of the account that contributes costs to the noted account: • Assigned costs; values are Resource (R), Activity (A), or Cost Object (C) • Entered costs; value is CEE (Cost Element Entered) • Bills of costs; value is Ext (external items) or Int (internal items)
Cost	Cost of the account or cost element

Report header	**See "Report Header" at top of chapter.**
%	Percentage of the account; calculated by dividing the cost of the account that contributes costs or unit cost by the cost of the account that receives costs
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty
Output	Output type of the account that receives costs; Default, User, or Sold; the quantity can be one of the following: • Default: Quantity is 1 • User: The user-entered output quantity • Sold: The quantity sold
Total Contributions	Subtotal for each type of cost (assigned costs, entered costs, or bills of costs)
Total cost	Sum of all the Total Contributions

Unassigned Costs Report Template

Overview

This report template lists accounts that do not have outgoing assignments. This information can help users validate a model because, in general, all accounts have outgoing assignments.

Information in this report

Report header	**See "Report Header" at top of chapter.**
Name	Name of the module or unassigned account that does not have outgoing assignments or has not been used as an internal bill of costs unit in another account
Driver	Driver name
Reference	Reference of the unassigned account
Cost	Cost or total costs of unassigned accounts for the module

Unit Cost Report Template

Overview

This report template lists unit costs that reveal the costs of a single product or customer.

Information in this report

Report header	See "Report Header" at top of chapter.
Name	Name of the account
Reference	Account reference
Output Type	The source for the output type: Driver, Default, User, or Sold; a given output type could be based on the default value=1; or the output type could be based on a user-entered value (OutputUE); or the output type could be based on the Quantity Sold (SoldQty).
Output Qty	A number that is based on one of the following: • default value=1 • Output Quantity UE (OutQtyUE) • Sold Quantity=<value that the destinations in an Assignment have as Sold Qty> • Total Driver Quantity Calculated (TDQCalc)
Cost	Cost of the account or cost element
Unit Cost	Unit cost of the account; calculated by dividing the Cost by the Output Qty

Chapter 3

Report Export Tables

Correlation Report – Export Table

This report displays the correlation between Cost and Used Quantity.

Column name	Data type	Length	Explanation
ModelId	Integer		Model ID
ModelName	Alphanumeric	64	Model name
ModuleId	Integer		Module Id for ExternalUnit (0), Resource (1), Activity (2), and CostObject (3)
Module	Alphanumeric	64	Module name
DriverName	Alphanumeric	64	Driver name

Column name	Data type	Length	Explanation
AccountId	Integer		Account Id. Useful for joining with other tables
AccountRefnum	Integer		Account Reference
AccountName	Alphanumeric	64	Account Name
Correlation	Float		The correlation value in the range [-1,+1]
Periods	Integer		The number of period/scenarios the account exists in
CostMean	Float		The mean average of cost for account across the period/scenarios
UsedQuantityMean	Float		The mean average of UsedQuantity across the period/scenarios

Destination Furthest Report – Export Table

This report export table contains information about contributions made by accounts in the model that precede a specific account.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
SourceName	Alphanumeric	64	The name of the account that gives the costs
SourceReference	Alphanumeric	64	The reference of the account that gives the costs
SourceModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
SourceModuleName	Alphanumeric	64	The name of the module containing the account that gives the costs
SourceParentName	Alphanumeric	64	The name of the rollup account that contains the account that gives the costs
SourceParentReference	Alphanumeric	64	The reference of the rollup account that contains the account that gives the costs
SourceOutputQuantity	Float		The output quantity of the account that receives the costs

Column name	Data type	Length	Explanation
SourceOutputType	Integer		The **output quantity type** of the account that gives the costs 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
SourceCost	Float	64	The cost of the account that gives the costs
DestinationName	Alphanumeric	64	The name of the account that receives the costs
DestinationReference	Alphanumeric	64	The reference of the account that receives the costs
DestinationModuleId	Integer		The **ID of the module** that contains the account that receives the costs 1 Resource 2 Activity 3 Cost Object
DestinationModuleName	Alphanumeric	64	The module that contains the account that receives the costs
DestinationOutputQuantity	Float		The output quantity of the account that gives the costs
DestinationOutputType	Integer		The **type** of the account that receives the costs
DriverName	Alphanumeric	64	The name of the driver on the account that gives the costs
ContributionCost	Float		The cost contributed by the account
ContributionPercent	Float		The percentage of the total cost contributed by the account
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account that receives the costs
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account that receives the costs

Dimensional Attribute Cost Report – Export Table

This report export table contains information about costs of accounts for selected dimension attributes.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
Id	Integer		The ID of the account
Type	Integer		The type of cost: 3=Normal Cost 27=Reversing Entry to Avoid Dimensional Double Counts
Name	Alphanumeric	64	The name of the account
Reference	Alphanumeric	64	The reference of the account
DimName	Alphanumeric	64	The name of the attribute folder containing the attributes attached to the account
DimMemberName	Alphanumeric	64	The name of the attribute attached to the account
Cost	Float		The cost contributed by bills of costs and assigned costs
BOCCost	Float		The cost contributed by bills of costs
OutputQuantity	Float		The output quantity for the account

Column name	Data type	Length	Explanation
OutputType	Integer		The **output quantity type** for the account (Sold or User or Driver or Default) 1 User-entered 2 Used 3 Driver 4 Sold 5 Default

Dimensional View Report – Export Table

This report export table contains information about revenue, cost, profit, and unit cost for the selected dimension.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
AccountName	Alphanumeric	64	The dimensional value of the structural dimension displayed in the column layout
AccountReference	Alphanumeric	64	The dimensional value of the structural dimension displayed in the column layout
DimA - J Name	Alphanumeric	64	The name of the first to tenth dimension attribute selected for defining the view in the Report Wizard
DimA - J MemberName	Alphanumeric	64	The name of the first to tenth dimension member attribute selected for defining the view in the Report Wizard
DimA - J MemberReference	Alphanumeric	64	The reference of the first to tenth dimension member attribute selected for defining the view in the Report Wizard
Cost	Float		The cost of the dimensional intersection

Column name	Data type	Length	Explanation
Revenue	Float		The revenue of the dimensional intersection account
Profit	Float		The Revenue minus the Cost of the dimensional intersection account
SoldQuantity	Float		The sold quantity for the dimensional intersection account

Driver - Cost and Rate Report – Export Table

This report export table contains information about drivers.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
DriverName	Alphanumeric	64	The name of the driver on the account that gives the costs
DriverType	Integer		The **driver type** 1 EvenlyAssigned 2 Percentage 3 Normal 4 Weighted 5 BillOfCost 6 Calculated 7 SalesVolume
DriverFormula	Alphanumeric	memo	The formula for a calculated driver
DriverQuantityType	Integer		The driver quantity type of the account that gives the costs
AccountName	Alphanumeric	64	The name of the account

Column name	Data type	Length	Explanation
AccounttReference	Alphanumeric	64	The reference of the account
ParentName	Alphanumeric	64	The name of the rollup account that contains the account
ParentReference	Alphanumeric	64	The reference of the rollup account that contains the account
TDQ	Float		The total driver quantity of the account
UserTDQ	Float		The user-entered total driver quantity of the account
Cost	Float		The cost of the account
Dimension	Alphanumeric	64	The attribute folders, up to four selected in the Report Wizard, on the account
DimensionMember	Alphanumeric	64	The attribute names, up to four selected in the Report Wizard, on the account

Idle Capacity Report – Export Table

This report export table contains information about the idle capacity of accounts.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** that contains the account that receives the costs 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
Name	Alphanumeric	64	The name of the account
Reference	Alphanumeric	64	The reference of the account
DriverName	Alphanumeric	64	The name of the driver on the account
IdleQuantity	Float		The idle driver quantity for the account
IdleCost	Float		The idle cost for the account

Column name	Data type	Length	Explanation
IdlePercent	Float		The idle percentage for the account
Cost	Float		The costs of the account
TDQ	Float		The total driver quantity of the account
UserTDQ	Float		The user-entered total driver quantity of the account
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account

Module Hierarchy Report – Export Table

This report export table contains information about accounts and their contents for each module.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleName	Alphanumeric	64	The name of the module
Id	Integer		The ID of the item
Type	Integer		The type of item 1 Root 2 Rollup account 3 Account 4 Entered cost element 5 Assigned cost element 11 Internal unit cost element 12 External unit cost element 13 External unit
Name	Alphanumeric	64	The name of the item (rollup account, account, or cost element)
Reference	Alphanumeric	64	The reference of the account

Column name	Data type	Length	Explanation
ModuleLevel	Integer		The level in the hierarchy
TerminalDimension	Alphanumeric	64	The dimensional value of the structural dimension displayed in the column layout
ParentId	Integer		The ID of the parent rollup account
Cost	Float		The cost of the item
OutputQuantity	Float		The output quantity for the account
OutputType	Integer	256	The **output quantity type** for the account 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account that receives the costs
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account that receives the costs

Multi-level Contributions Report – Export Table

This report export table contains information about contributing accounts and their costs at every level back to the original contributing accounts.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
FinalId	Integer		The ID of the final account in the assignment path
FinalModuleId	Integer		The **ID of the module** that contains the final account in the assignment path 1 Resource 2 Activity 3 Cost Object

Column name	Data type	Length	Explanation
FinalModuleName	Alphanumeric	64	The module that contains the final account in the assignment path
FinalName	Alphanumeric	64	The name of the final account in the assignment path
FinalReference	Alphanumeric	64	The reference of the final account in the assignment path
FinalCost	Float		The total cost of the final account in the assignment path
OutputQuantity	Float		The output quantity of the final account in the assignment path
OutputType	Integer		The **output quantity type** of the final account in the assignment path 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
Dimension	Alphanumeric	64	The attribute folders, up to four selected in the Report Wizard
DimensionMember	Alphanumeric	64	The attribute names, up to four selected in the Report Wizard
ModId_D1 - D10	Integer		The **ID of the module** containing the account one to 10 steps back in the assignment path that contributes to the final account 1 Resource 2 Activity 3 Cost Object
Mod_D1 - D10	Alphanumeric	64	The name of the module containing the account one to 10 steps back in the assignment path that contributes to the final account
CeSrcId_D1 - D10	Integer		The ID of the account one to 10 steps back in the assignment path that contributes to the final account
Name_D1 - D10	Alphanumeric	64	The name of the account one to 10 steps back in the assignment path that contributes to the final account
Reference_D1 - D10	Alphanumeric	64	The reference of the account one to 10 steps back in the assignment path that contributes to the final account

Column name	Data type	Length	Explanation
Percent_D1 - D10	Float		The percentage of cost that the account one to 10 steps back in the assignment path contributes to the final account

Profit Cliff Report – Export Table

This report export table contains information about the cumulative profit for one dimension.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
DimensionName	Alphanumeric	64	The name of the dimension, selected in the Report Wizard, used to create the Profit Cliff
DimMemberName	Alphanumeric	64	The name of the attribute attached to the account
DimMemberReference	Alphanumeric	64	The reference of the attribute attached to the account
Cost	Float		The cost of the account
Revenue	Float		The revenue for the noted dimensional intersection account
Profit	Float		The profit for the noted dimensional intersection account
SoldQuantity	Float		The quantity sold for the noted dimensional intersection account

Resource Contributions Report – Export Table

This report export table contains information about the highest level accounts that contribute costs to selected accounts.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model

Column name	Data type	Length	Explanation
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
SourceId	Integer		The ID of the account that gives the costs
SourceName	Alphanumeric	64	The name of the account that gives the costs
SourceReference	Alphanumeric	64	The reference of the account that gives the costs
SourceModuleId	Integer		The **ID of the module** containing the account that gives the costs 1 Resource 2 Activity 3 Cost Object
SourceModuleName	Alphanumeric	64	The name of the module containing the account that gives the costs
SourceModuleNameShort	Alphanumeric	64	The abbreviation of the name of the module
SourceUnitCost	Float		The unit cost (contributing account costs/ destination account output quantity)
SourceOutputQuantity	Float		The output quantity for the account that gives the costs
SourceOutputType	Integer		The **output quantity type** of the account that gives the costs 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
SourceCeId	Integer	64	The ID of the cost element
SourceCeName	Alphanumeric	64	The name of the cost element of the account that gives the costs
SourceCeReference	Alphanumeric	64	The reference of the cost element of the account that gives the costs
SourceCeContribution	Float		The costs contributed from the cost element of the account that gives the costs

Column name	Data type	Length	Explanation
SourceCeType	Integer		The **type of the cost element** of the account that gives the costs 1 Root 2 Rollup account 3 Account 4 Entered cost element 5 Assigned cost element 11 Internal unit cost element 12 External unit cost element 13 External unit
DestinationId	Integer		The ID of the account that receives the costs
DestinationName	Alphanumeric	64	The name of the account that receives the costs
DestinationReference	Alphanumeric	64	The reference of the account that receives the costs
DestinationModuleId	Integer		The **ID of the module** that contains the account that receives the costs 1 Resource 2 Activity 3 Cost Object
DestinationModuleName	Alphanumeric	64	The module that contains the account that receives the costs
DestinationParentName	Alphanumeric	64	The name of the rollup account that contains the account that receives the costs
DestinationParentReference	Alphanumeric	64	The reference of the rollup account that contains the account that receives the costs
DestinationOutputQuantity	Float		The output quantity (Sold, User, Driver, Default) for the account that receives the costs
DestinationOutputType	Integer		The **output quantity type** of the account that receives the costs 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
DestinationSoldQuantity	Float		The sold quantity of the account that receives the costs
DestinationCost	Float		The total costs of the account that receives the costs

Column name	Data type	Length	Explanation
ContributionCost	Float		The costs contributed (assigned and bills of costs) to the account that receives the costs
ContributionPercent	Float		The percentage of costs contributed to the account that receives the costs
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account that gives the costs
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account that gives the costs

Single-stage Assignments Report – Export Table

This report export table contains information about assignments.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
SourceId	Integer		The ID of the account that gives the costs
SourceName	Alphanumeric	64	The name of the account that gives the costs
SourceReference	Alphanumeric	64	The reference of the account that gives the costs
SourceModuleId	Integer	64	The **ID of the module** containing the account that gives the costs 1 Resource 2 Activity 3 Cost Object
SourceModuleName	Alphanumeric	64	The name of the module containing the account that gives the costs
SourceParentName	Alphanumeric	64	The name of the rollup account that contains the account that gives the costs
SourceParentReference	Alphanumeric	64	The reference of the rollup account that contains the account that gives the costs
SourceTDQ	Float		The total driver quantity of the account that gives the costs

Column name	Data type	Length	Explanation
SourceUserTDQ	Float		The user-entered total driver quantity of the account that gives the costs
SourceCost	Float	64	The cost of the account that gives the costs
DestinationId	Integer		The ID of the account that receives the costs
DestinationName	Alphanumeric	64	The name of the account that receives the costs
DestinationReference	Alphanumeric	64	The reference of the account that receives the costs
DestinationModuleId	Integer	64	The **ID of the module** that contains the account that receives the costs 1 Resource 2 Activity 3 Cost Object
DestinationModuleName	Alphanumeric	64	The module that contains the account that receives the costs
DestinationOutputQuantity	Float		The output quantity for the account that receives the costs
DestinationOutputType	Integer		The **output quantity type** for the account that receives the costs 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
DriverName	Alphanumeric	64	The name of the driver on the account that gives the costs
DriverType	Integer	64	The **type of driver** 1 EvenlyAssigned 2 Percentage 3 Normal 4 Weighted 5 BillOfCost 6 Calculated 7 SalesVolume
DriverFormula	Alphanumeric	memo	The formula for a calculated driver
DriverQuantity	Float		The driver quantity that flows from one account to another

Column name	Data type	Length	Explanation
DriverQuantityType	Integer		The **driver quantity type** of the account that gives the costs 0 Shared 1 Unique
DriverWeight	Float		The driver weight for the quantity that flows from one account to another
CalculatedDriverQuantity	Float		The calculated driver quantity that flows from one account to another
DriverCost	Float		The driver cost that flows from one account to another
DriverPercent	Float		The driver percent that flows from one account to another
IdleQuantity	Float		The idle driver quantity for the account that gives the costs
IdleCost	Float		The idle cost for the account that gives the costs
IdlePercent	Float		The idle percentage for the account that gives the costs
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account that receives the costs
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account that receives the costs

Single-stage Contributions Report – Export Table

This report export table contains information about contributions made by accounts that precede a selected account.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
SourceName	Alphanumeric	64	The name of the account that gives the costs
SourceReference	Alphanumeric	64	The reference of the account that gives the costs

Column name	Data type	Length	Explanation
SourceModuleId	Integer		The **ID of the module** containing the account that gives the costs 1 Resource 2 Activity 3 Cost Object
SourceModuleName	Alphanumeric	64	The name of the module containing the account that gives the costs
SourceModuleNameShort	Alphanumeric	64	The abbreviation of the name of the module containing the account that gives the costs
SourceUnitCost	Float		The unit cost
SourceOutputQuantity	Float		The output quantity for the account that gives the costs
SourceOutputType	Integer		The output quantity type for the account that gives the costs
DestinationId	Integer		The ID of the account that receives the costs
DestinationName	Alphanumeric	64	The name of the account that receives the costs
DestinationReference	Alphanumeric	64	The reference of the account that receives the costs
DestinationModuleId	Integer		The **ID of the module** that contains the account that receives the costs 1 Resource 2 Activity 3 Cost Object
DestinationModuleName	Alphanumeric	64	The module that contains the account that receives the costs
DestinationParentName	Alphanumeric	64	The name of the rollup account that contains the account that receives the costs
DestinationParentReference	Alphanumeric	64	The reference of the rollup account that contains the account that receives the costs
DestinationSoldQuantity	Float		The sold quantity of the account that receives the costs
DestinationCost	Float		The cost of the account that receives the costs
DestinationOutputQuantity	Float		The output quantity for the account that gives the costs (Sold, User, Driver, Default)

Column name	Data type	Length	Explanation
DestinationOutputType	Integer		The **output quantity type** of the account that receives the costs 1 User-entered 2 Used 3 Driver 4 Sold 5 Default
DriverName	Alphanumeric	64	The name of the driver on the account that receives the costs
DriverType	Integer		The **driver type** 1 EvenlyAssigned 2 Percentage 3 Normal 4 Weighted 5 BillOfCost 6 Calculated 7 SalesVolume
DriverQuantityType	Integer		The **driver quantity type** of the account that receives the costs 0 Shared 1 Unique
CostElementType	Integer		The **type of contributed costs** 1 Root 2 Rollup account 3 Account 4 Entered cost element 5 Assigned cost element 11 Internal unit cost element 12 External unit cost element 13 External unit
ElementType	Alphanumeric	64	The type of contributing costs
CalculatedDriverQuantity	Float		The calculated driver quantity that flows from one account to another
BOCCost	Float		The costs from bills of costs of the account that receives the costs
ContributionCost	Float		The cost from bills of costs and assigned costs of the account that receives the costs

Column name	Data type	Length	Explanation
DriverCost	Float		The driver costs that flow from one account to another
ContributionPercent	Float		The percentage of costs of the account that receives the costs
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account that gives the costs
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account that gives the costs

Unassigned Cost Report Export – Table

This report export table contains information about accounts that do not have outgoing assignments.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
Type	Integer		The type of account: 1=Root 2=RollupAccount 3= Account 13=ExternalUnit 27=BalanceAdjustment
Name	Alphanumeric	64	The name of the account
Reference	Alphanumeric	64	The reference of the account
DriverName	Alphanumeric	64	The name of the driver on the account

Column name	Data type	Length	Explanation
Cost	Float		The cost of the account
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account

Unit Cost Report – Export Table

This report export table contains information about unit costs.

Column name	Data type	Length	Explanation
ModelName	Alphanumeric	64	The name of the model
Period	Alphanumeric	64	The name of the period
Scenario	Alphanumeric	64	The name of the scenario
ModuleId	Integer		The **ID of the module** 1 Resource 2 Activity 3 Cost Object
ModuleName	Alphanumeric	64	The name of the module
Type	Integer		The type of item: 3=Account
Name	Alphanumeric	64	The name of the account
Reference	Alphanumeric	64	The reference of the account
Cost	Float		The cost of the account
BOCCost	Float		The costs contributed by the account from bills of costs
OutputQuantity	Float		The output quantity for the account
OutputType	Integer		The **output quantity type** for the account 1 User-entered 2 Used 3 Driver 4 Sold 5 Default

Column name	Data type	Length	Explanation
Dimension	Alphanumeric	256	The attribute folders, up to four selected in the Report Wizard, on the account
DimensionMember	Alphanumeric	256	The attribute names, up to four selected in the Report Wizard, on the account

Part 3

Surveys

Chapter 4

Introduction to Surveys

Introduction to Surveys

One of the most difficult tasks in maintaining a model is keeping its data accurate and up-to-date. Now you can create Web surveys to solicit data from the people who are directly responsible for the activities and accounts in your model. Data from the surveys is written directly to staging tables that have been exported from the model.

The following table shows the types of surveys that you can create for each module and the fields that a survey taker can update for each type of survey.

Note: Each field name is qualified by the staging table that it is in.

Module	Type of survey	Fields that can be updated
External Unit	Quantity: (*account*)	Assignment.DriverQuantityFixed See “Assignment table” on page 226.
	Unit Costs	ExternalUnit.UnitCostEntered See “ExternalUnit table” on page 235.

Module	Type of survey	Fields that can be updated
Resource	Resource Drivers	Assignment.DriverQuantityFixed See "Assignment table" on page 226.
	Resource Costs	EnteredCostElement.EnteredCost See "EnteredCostElement table" on page 234.
	Numeric Attribute	ValueAttributeAssociation.NumericValue See "ValueAttributeAssociation table" on page 247.
Activity	Activity Drivers	Assignment.DriverQuantityFixed See "Assignment table" on page 226.
	Numeric Attributes	ValueAttributeAssociation.NumericValue See "ValueAttributeAssociation table" on page 247.
Cost Object	Cost Object Drivers	Assignment.DriverQuantityFixed See "Assignment table" on page 226.
	Revenues and Sold Quantities	Account.Revenue Account.SoldQuantity See "Account table" on page 223.
	Output Quantities	Account.OutputQuantityUE See "Account table" on page 223.
	Numeric Attributes	ValueAttributeAssociation.NumericValue See "ValueAttributeAssociation table" on page 247.

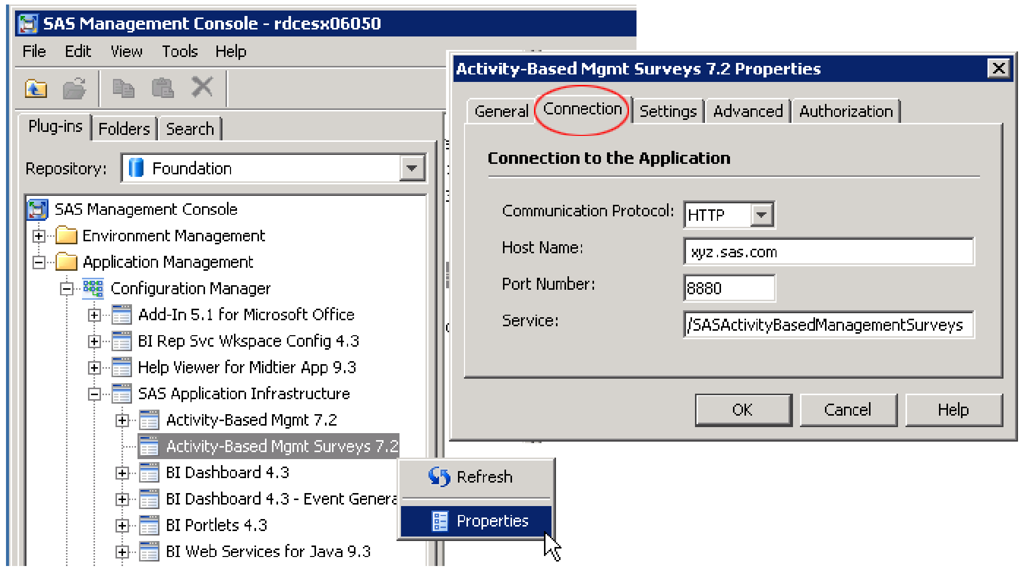
Logging On

Log on to surveys by typing the survey URL into the command line of a browser. To determine the URL, you can do the following:

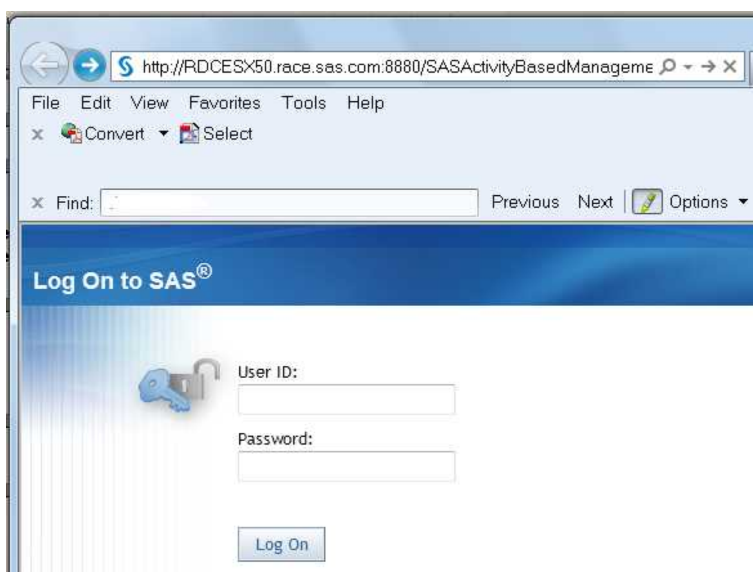
1. Log on to SAS Management Console as an administrator, and access the SAS Activity-Based Management middle-tier server.
2. Click the **Plug-Ins** tab.
3. Expand **Application Management**.
4. Expand **Configuration Manager**.
5. Expand **SAS Application Infrastructure**.
6. Right-click **Activity-Based Mgmt Surveys 7.2** and select **Properties**.
7. Click the **Connection** tab on the Properties window.

The connection information appears. In the following picture, you can see that the URL for invoking surveys is:

```
http://xyz.sas.com:8880/SASActivityBasedManagementSurveys
```



8. Type the survey URL into the command line of a browser.



9. Log on with a user ID and password. The user must exist in the SAS Metadata Server with the following capabilities:

   - To create a survey, the Create Model capability
   - To take a survey, the Take Surveys capability

   See Chapter 1, “User Capabilites and Groups,” on page 3.

The Survey Process in a Nutshell

Overview

The steps for using surveys are the following:

- "Export a model" on page 82
- "Create a survey" on page 82
- "Assign survey items to survey takers" on page 82
- "Take the survey" on page 82
- "Administer the survey" on page 85
- "Reimport the model" on page 85

Export a model

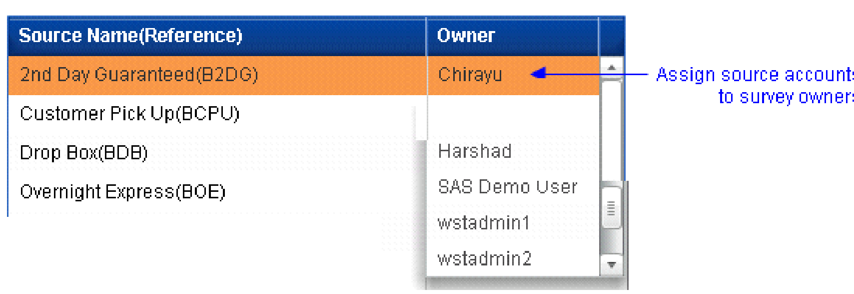
See Chapter 5, "Exporting Data to Use with Surveys," on page 87.

Create a survey

See Chapter 7, "Creating Surveys," on page 97.

Assign survey items to survey takers

The following picture shows how to assign a driver survey.



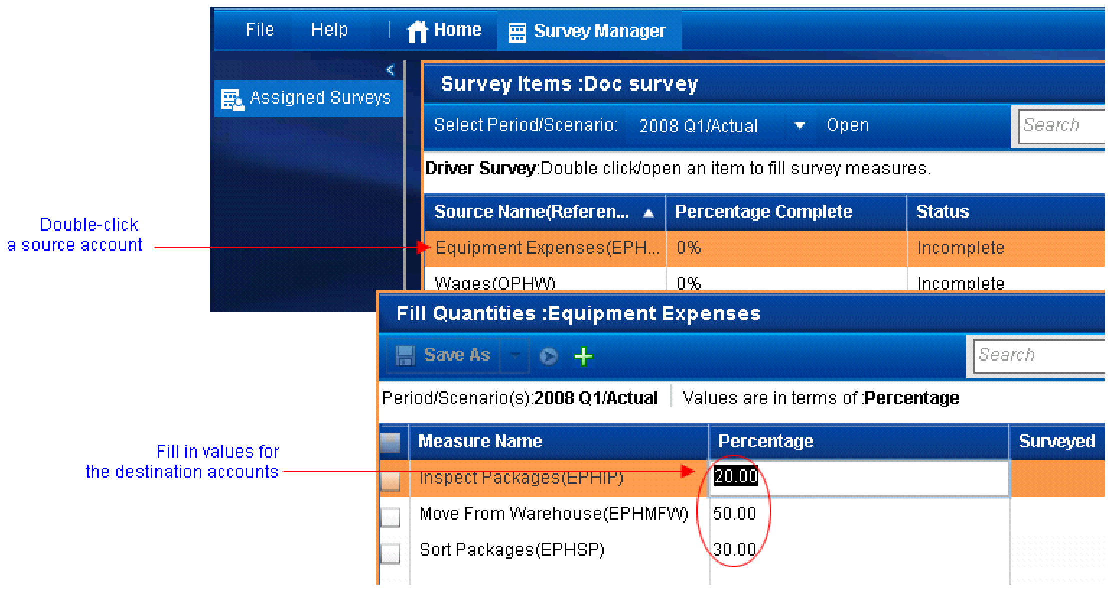
When the administrator finishes creating the survey, survey takers are notified by e-mail.

Take the survey

1. Answer the survey questions.

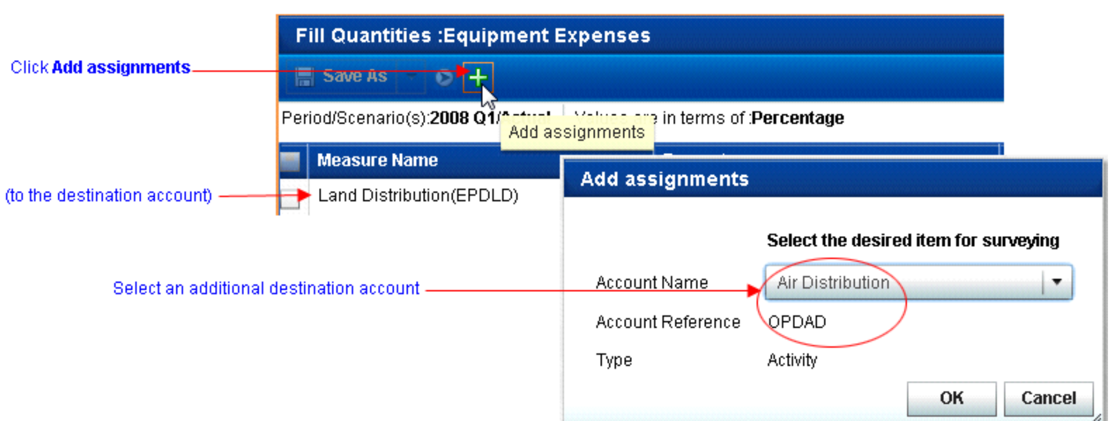
   The survey taker logs on and begins taking the assigned survey.

The following picture shows a driver survey.



2. Add additional assignments.

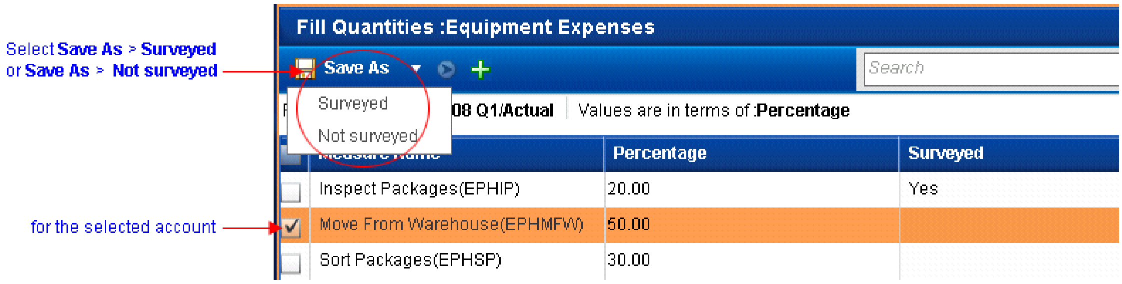
   While taking a driver survey, a survey taker can add assignments that are not currently in the model. See "Add New Assignments" on page 141.



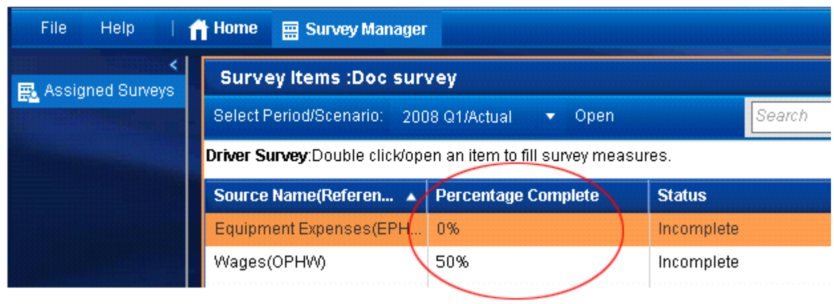
3. Save the survey items as surveyed.

   When each survey item has been satisfactorily responded to, the survey taker saves it as surveyed.

   Note: As soon as a survey taker saves an item as surveyed, it is written to a staging table in the database. If the survey taker subsequently saves the item as not surveyed, the value that was previously saved to the database is not rolled back.

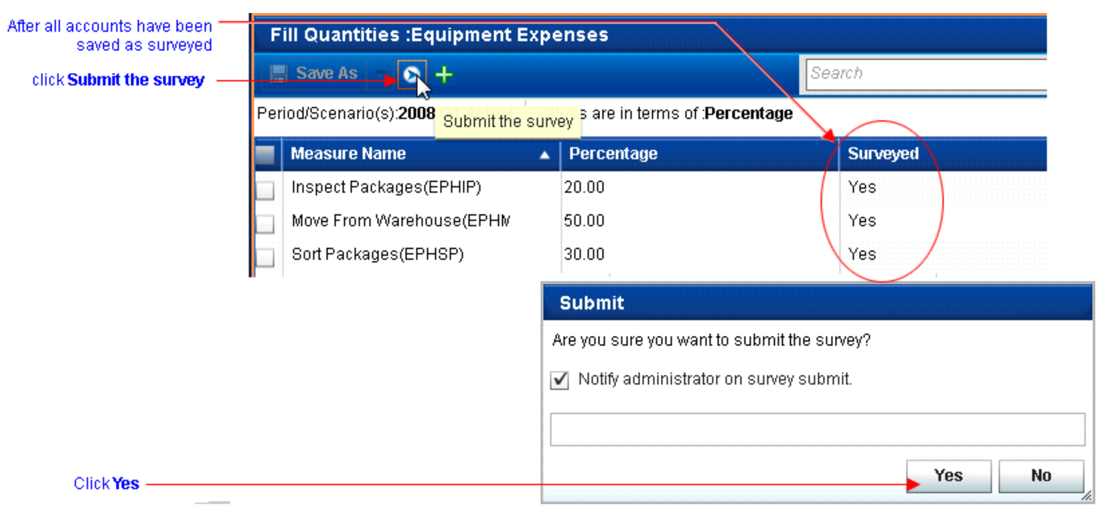


The Survey Manager displays the Percentage Complete—the percentage of survey items that have been saved as surveyed.

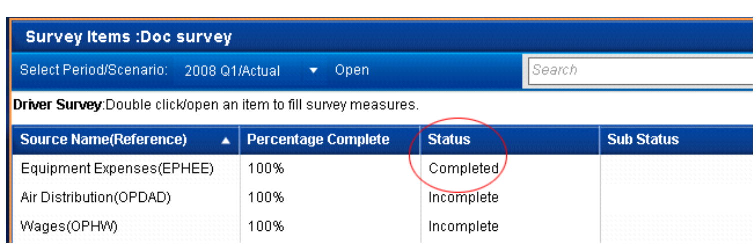


4. Submit the survey.

When all survey items have been saved as surveyed, the survey taker can submit the survey.



Once the survey taker submits the survey, its status changes to **Completed**, and the survey taker can no longer change any responses.



The administrator receives a notification e-mail.

Administer the survey

Among other administration tasks, a survey administrator can do the following:

1. Reopen a completed survey.

   On receiving notification that a survey has been submitted, an administrator has the option of reopening the survey by marking it as **Incomplete**. If an administrator reopens the survey, then:

   - the survey status changes to **Incomplete**.
   - the survey taker receives a notification.
   - the survey taker retakes the survey and submits it again after saving all items as surveyed.

   Note: Staging tables in the database are not rolled back if an administrator reopens a survey.

2. Audit new assignments.

   If the survey taker has added any new assignments, then an administrator must audit the assignments to either approve or disapprove.

   - If the administrator approves, then the new assignment values, as entered by the survey taker, are written to the staging table in the database.
   - If the administrator disapproves, then the new assignment is removed from the Assignment staging table in the database—it is as though the assignment was never made.

   See “Audit Surveys” on page 124.

Reimport the model

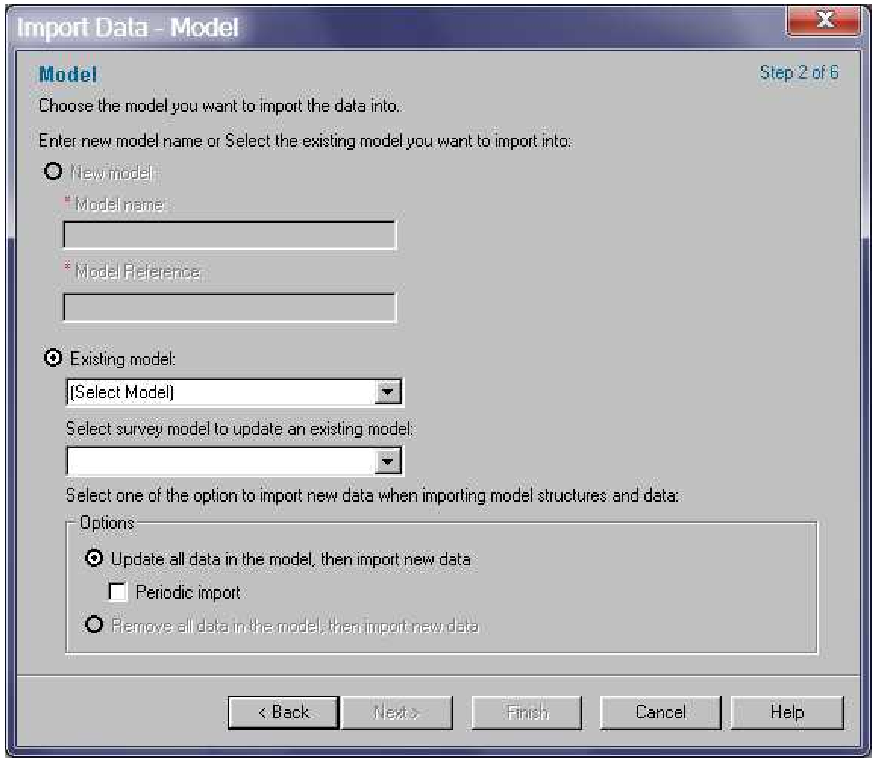
To reimport survey data:

1. Select **File** ⇨ **Import**.
2. Select **Surveys**.

   For general information on importing, see Chapter 13, “Importing,” on page 153.

3. Select an existing model to update with survey data. You cannot create a new model from survey data.

   Note: When importing into an existing model, make sure that the survey data is for the correct model. If the data is from a different model, the import can corrupt the existing model.



See Also

Chapter 12, "Importing Survey Data back into the Model," on page 147

Chapter 5

Exporting Data to Use with Surveys

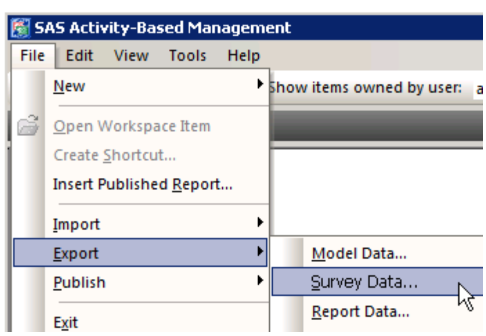
Exporting Survey Data

To create a survey, you do not have to export everything out of your model. Here are the steps to ensure you export only the required fields.

Note: Unless specified otherwise, accept all the default Export Wizard selections.

To export survey data, do the following. Also, see “Exporting Model Data to a Database” on page 169. And, see “Using the Export Wizard” on page 175.

1. Select **File** ⇨ **Export** ⇨ **Survey Model Data**.



2. On the Select Model window:

   a. Select the model to be exported.

   b. Select one of the following:

      - **New Survey Model**, and then enter a new survey model name.

        You will use this name to access the model while working with the survey.

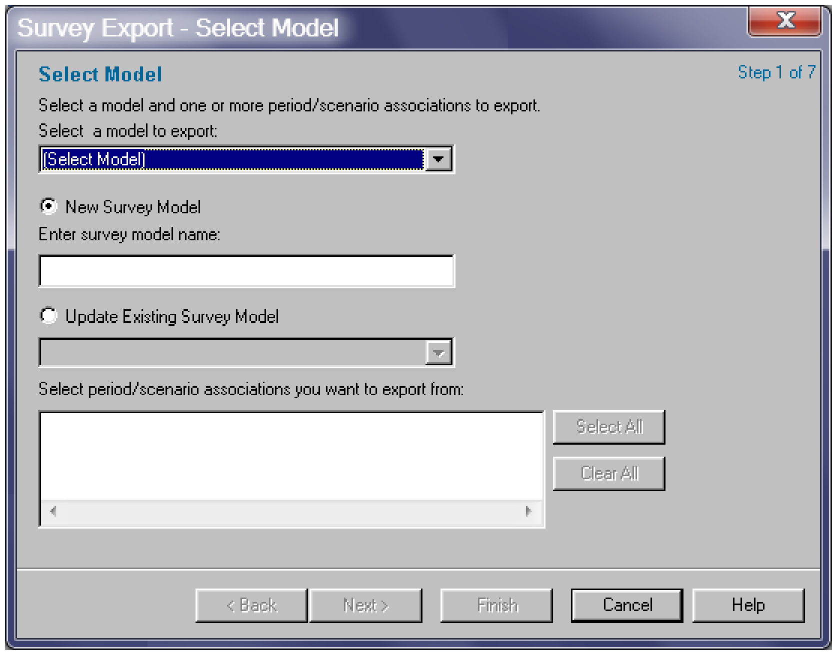
        Note: You can export survey data multiple times for the same model—for example, once for each period in the model. In this case, you would use a different survey model name for each export.

      - **Update Existing Survey Model**, and then select an existing survey model name.

Selecting this option overwrites the staging tables for the survey model in the database.

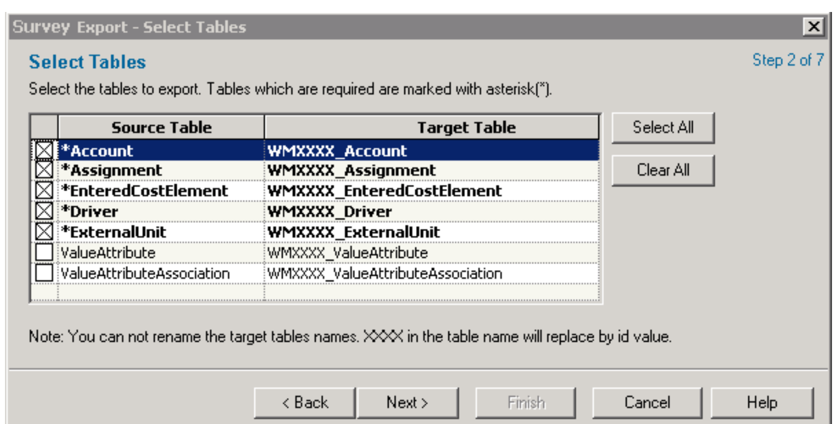
c. Select the Period/Scenario associations to be exported.

d. Click **Next**.



3. Select the tables to export.

Note: You cannot unselect the required tables.



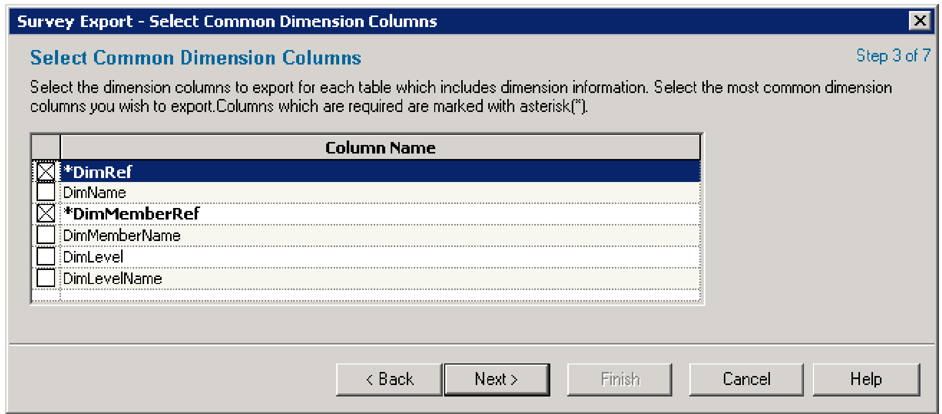
If you plan to survey Numeric Attributes, then you also must check the following two tables:

ValueAttribute table
See "ValueAttribute table" on page 247.

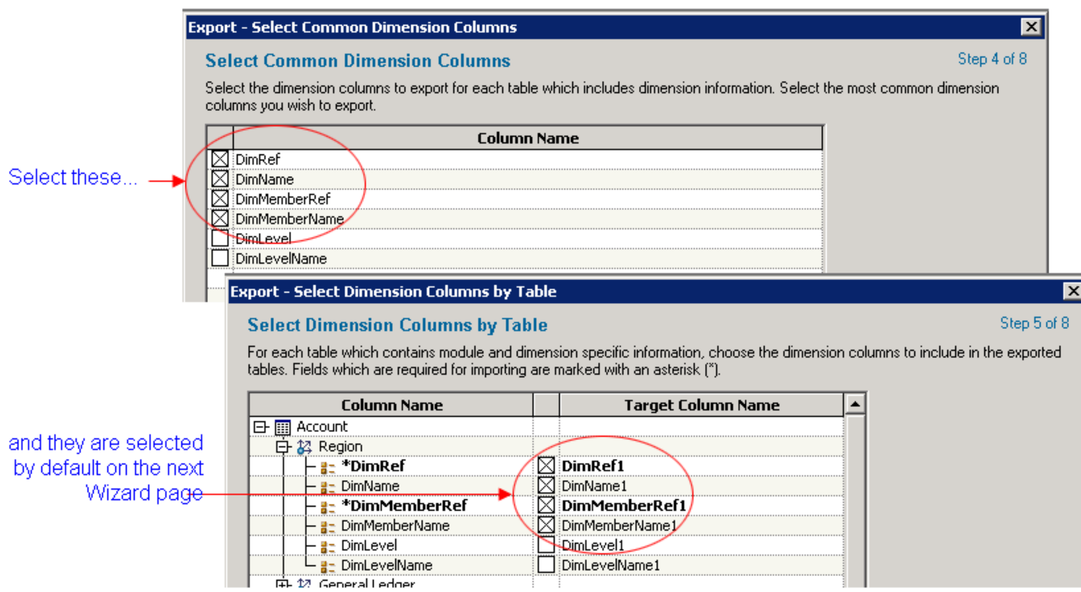
ValueAttributeAssociation table
See "ValueAttributeAssociation table" on page 247.

4. Select common dimension columns (that is, select the columns that will always be exported for each table).

Note: You cannot unselect required columns.

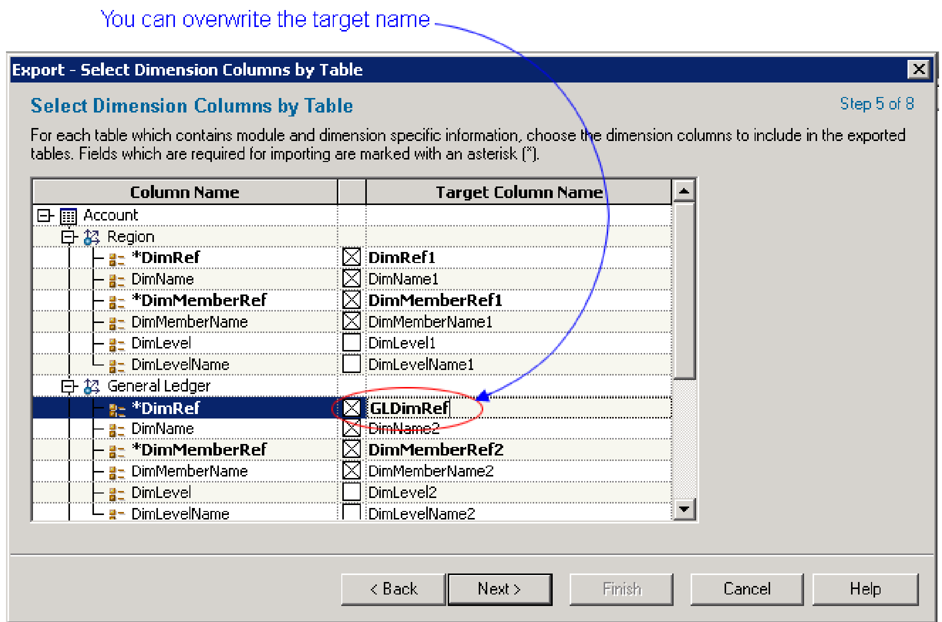


Columns that you select are selected by default on the next Export Wizard page. You can, however, change your selection on the next page. That is, you can deselect a field that you had selected, or select a field that you had deselected.

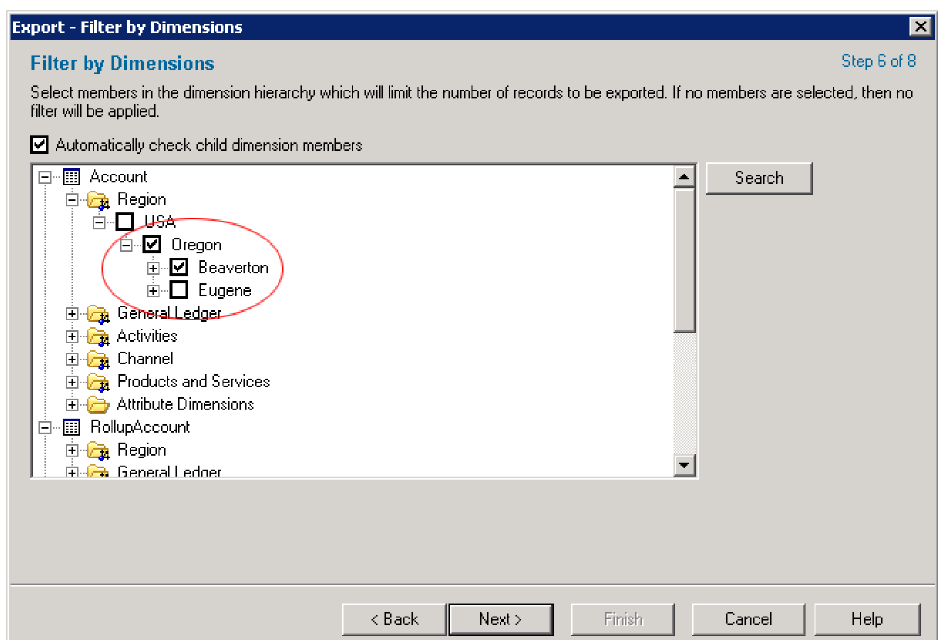


5. Select dimension columns to export for each table.

   Note: You can overwrite the name of the target column.



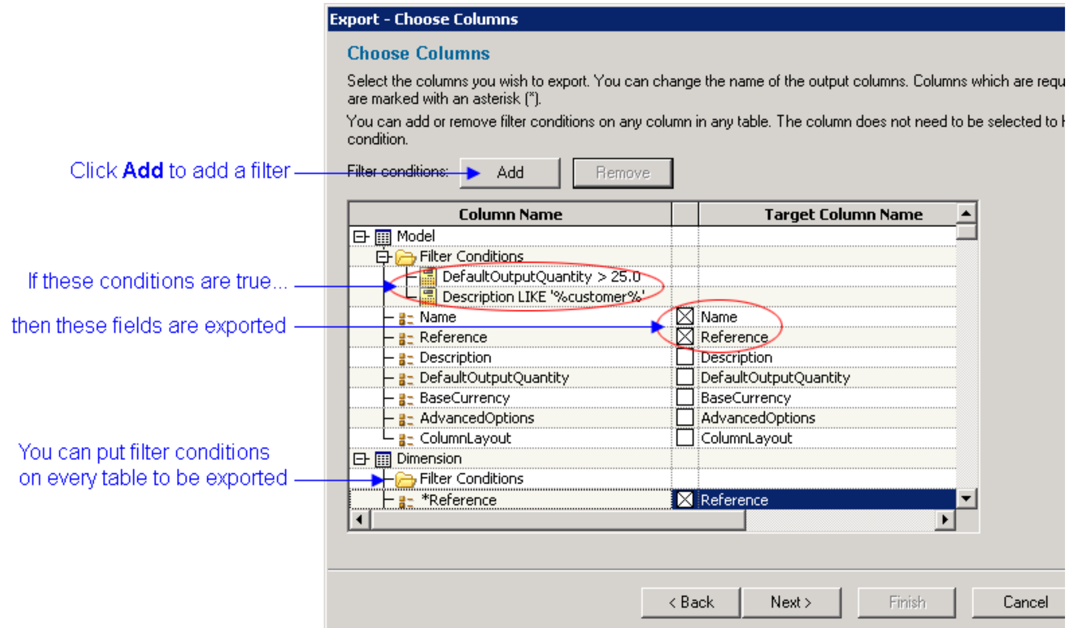
6. Filter by dimensions (that is, select those dimension members that will be exported for each table).



7. Choose the columns to export for each table.

   Click **Add** to add filters to further select the columns to export. Selected columns are exported only if they pass the filter. In other words, for a column to be exported, it must both

   - be selected
   - pass whatever filters exist for the table



In addition to all default selections, make sure the following columns are checked:

Account table

```
DriverName
Name
OutputQuantityUE
Revenue
SoldQuantity
PeriodicNote (optional - only if you have Account Notes)
```

Assignment table

```
Source Accounts.DriverName
DriverQuantityFixed
```

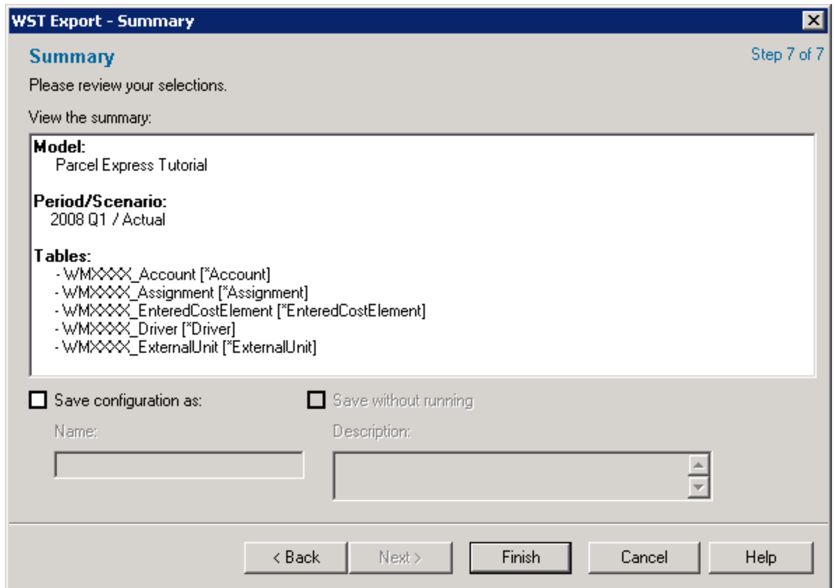
EnteredCostElement table

```
EnteredCost
```

ExternalUnit table

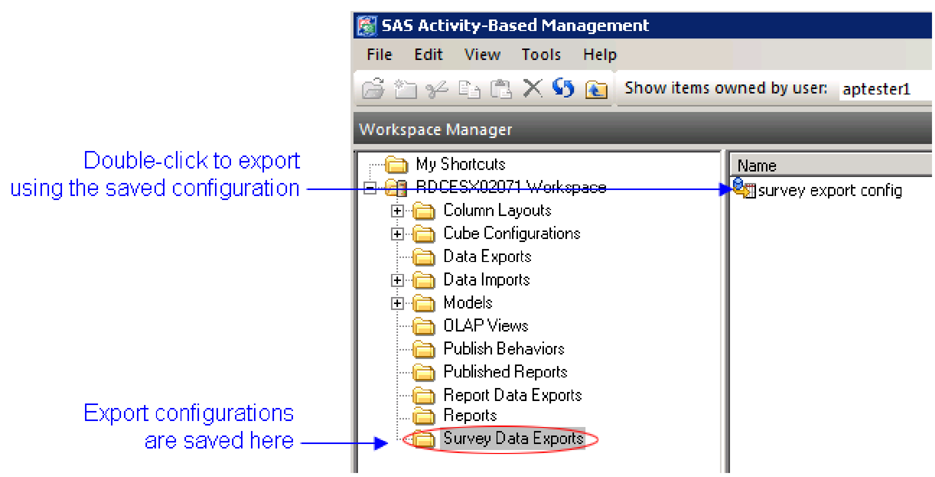
```
UnitCostEntered
Name
PeriodicNote (optional - only if you have ExternalUnit Notes)
```

8. Verify the summary and click **Finish**.



Select **Save configuration as** to save your selections. The selections are saved in the **Survey Data Exports** folder.

Double-click a saved configuration to begin exporting using the saved options. You can modify the options while using the Export Wizard.



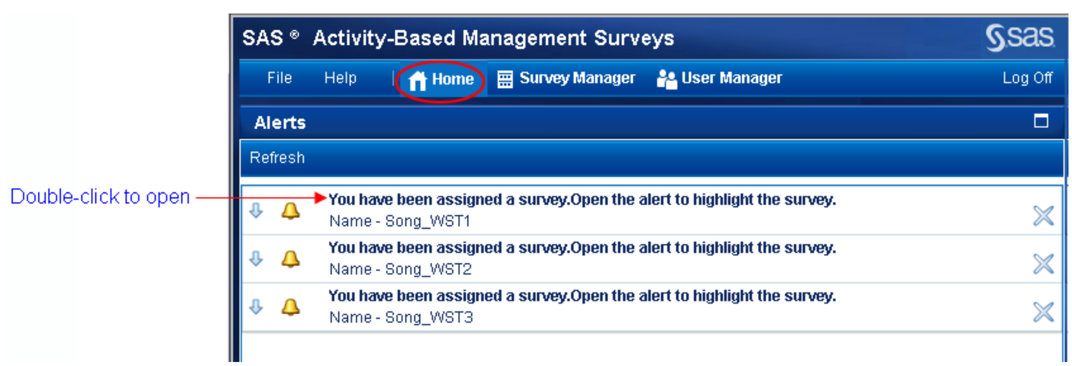
Chapter 6

Surveys: User Interface

Home

The Home page shows you all the alerts that you have received.

Double-click an alert to open the corresponding item.

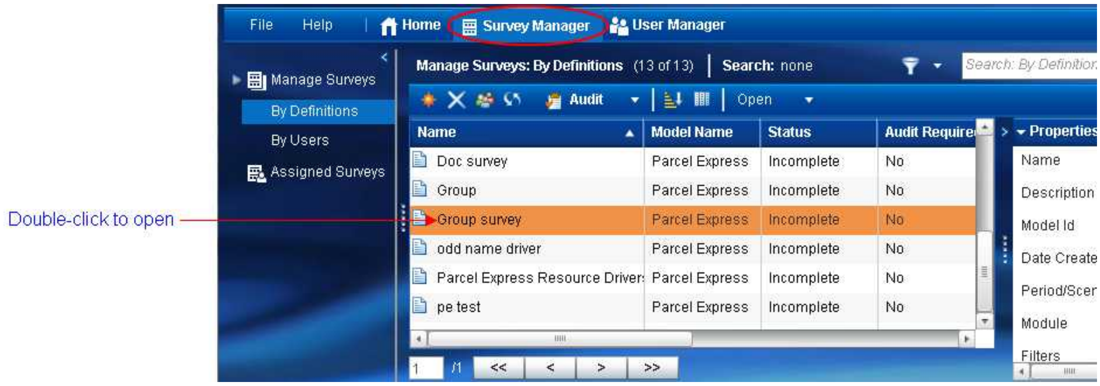


Survey Manager

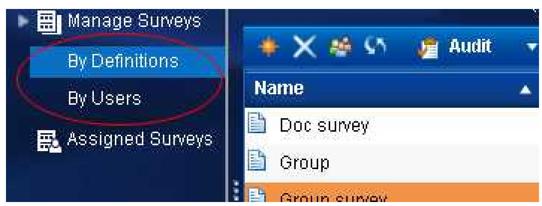
The Survey Manager lists surveys. Double-click a survey to open it. Use the Survey Manager to

- Create a survey. See “Create the Survey” on page 97.
- Take a survey. See “Take a Driver Survey” on page 139.
- Audit surveys. See “Audit Surveys” on page 124.
- View approved assignments. See “View Approved Assignments” on page 125.
- Delete an assignment. See “Delete a Survey” on page 128.

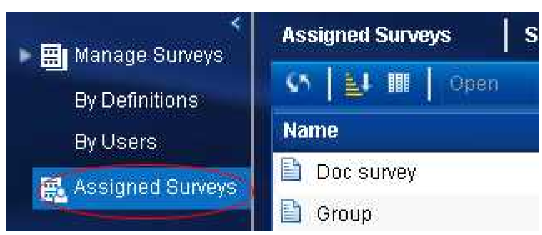
- Send an e-mail. See “Send an e-mail” on page 126.



You can choose to list surveys by definition or by user.



Click **Assigned Surveys** to see the surveys assigned to you.

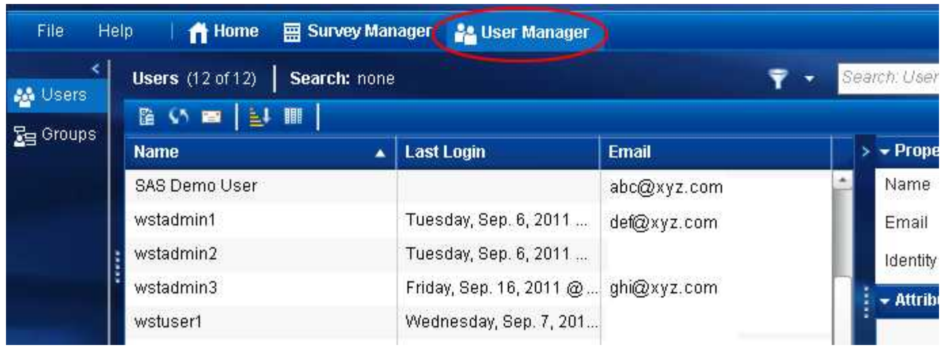


User Manager

The User Manager is available only to users who have Create Model capability. There is no Administer Surveys capability. For a discussion of roles and capabilities, see “Creating Roles” on page 4.

Use the User Manager to

- Reassign survey items to users. See “Reassign Users” on page 119.
- Manage user attributes. See “Add Attributes to a User” on page 120.
- Send e-mails to individual users. See “Send an e-mail” on page 126.


File
Help
Home
Survey Manager
User Manager
Users (12 of 12)
Search: none
Search: User
Users
Groups
Name
Last Login
Email
SAS Demo User
abc@xyz.com
wstadmin1
Tuesday, Sep. 6, 2011 ...
def@xyz.com
wstadmin2
Tuesday, Sep. 6, 2011 ...
wstadmin3
Friday, Sep. 16, 2011 @ ...
ghi@xyz.com
wstuser1
Wednesday, Sep. 7, 201...
Prope
Name
Email
Identity
Attrib

Chapter 7
Creating Surveys

Create the Survey

General Procedure

To create a survey, a user must have Create Model capability. There is no Create Survey capability. For a discussion of roles and capabilities, see “Creating Roles” on page 4.

To create a survey:

1. Log on using a browser.

   The logon URI uses the form: http://*machine_name:port*/SASActivityBasedManagementSurveys.

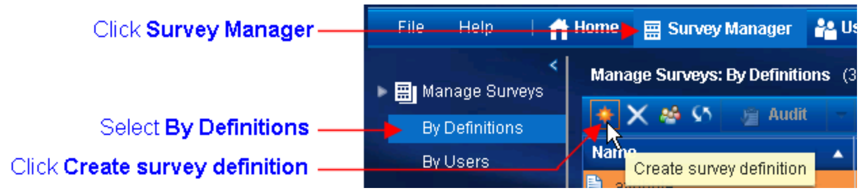
   The machine name depends on where the SAS Activity-Based Management middle-tier server is installed, for example:

   ```
   http://sasabm.xyz.com:8880/SASActivityBasedManagementSurveys
   ```

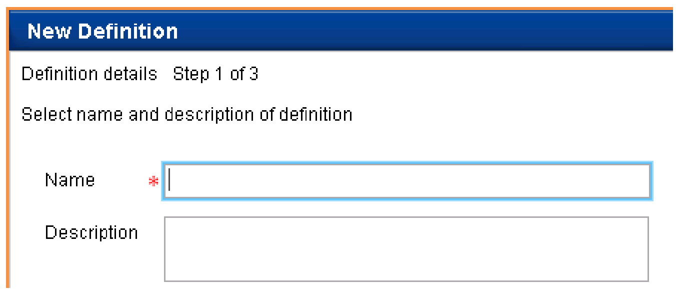
   See “Logging On” on page 80.

2. Click **Survey Manager**.
3. Select **By Definitions**.

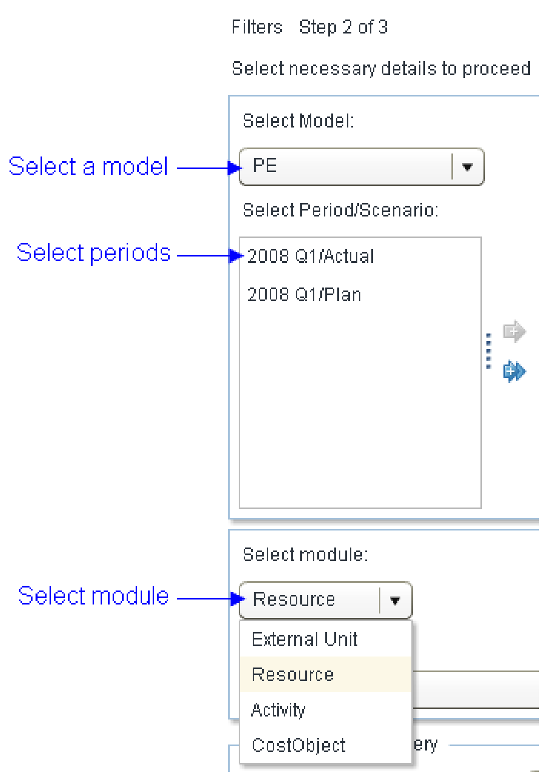
4. Select **Create a survey definition**.



5. Name the survey and (optionally) provide a description, and then click **Next**.



6. Select the model to survey.
7. Select the period/scenario associations to survey.
8. Select the module to survey. You can select
   - External Unit
   - Resource
   - Activity
   - Cost Object



9. The next steps depend on the type of survey that you want to define. The steps for each type of survey are described here:

- "External Unit Module: Unit Costs" on page 100
- "External Unit Module: Quantity (account)" on page 99
- "Resource Module: Resource Drivers" on page 101
- "Resource Module: Resource Costs" on page 102
- "Resource Module: Numeric Attributes" on page 103
- "Activity Module: Activity Drivers" on page 104
- "Activity Module: Numeric Attributes" on page 106
- "Cost Object Module: Cost Object Drivers" on page 107
- "Cost Object Module: Revenues and Sold Quantities" on page 108
- "Cost Object Module: Output Quantities" on page 109
- "Cost Object Module: Numeric Attributes" on page 110

Note: In assigning surveys items to survey owners, you cannot assign items to yourself.

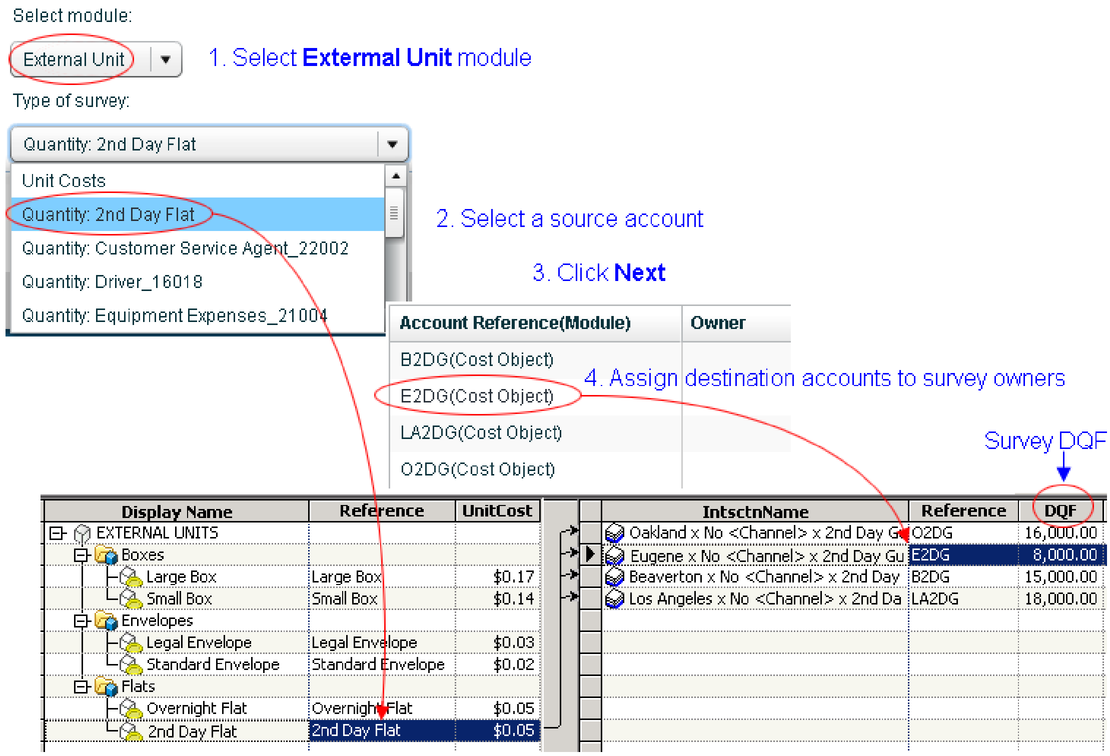
External Unit Module: Quantity (account)

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
External Unit	Quantity: (*account*)	Assignment.DriverQuantityFixed

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **External Unit** module.
3. Select a source account in the External Unit, and then click **Next**.
4. Assign survey owners to destination accounts.

The following picture summarizes the process.



External Unit Module: Unit Costs

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
External Unit	Unit Costs	ExternalUnit.UnitCostEntered

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **External Unit** module.
3. Select **Unit Costs**.
4. For **Top-Level Dimension Reference**, do one of the following:

   Specify **Any Dimension**

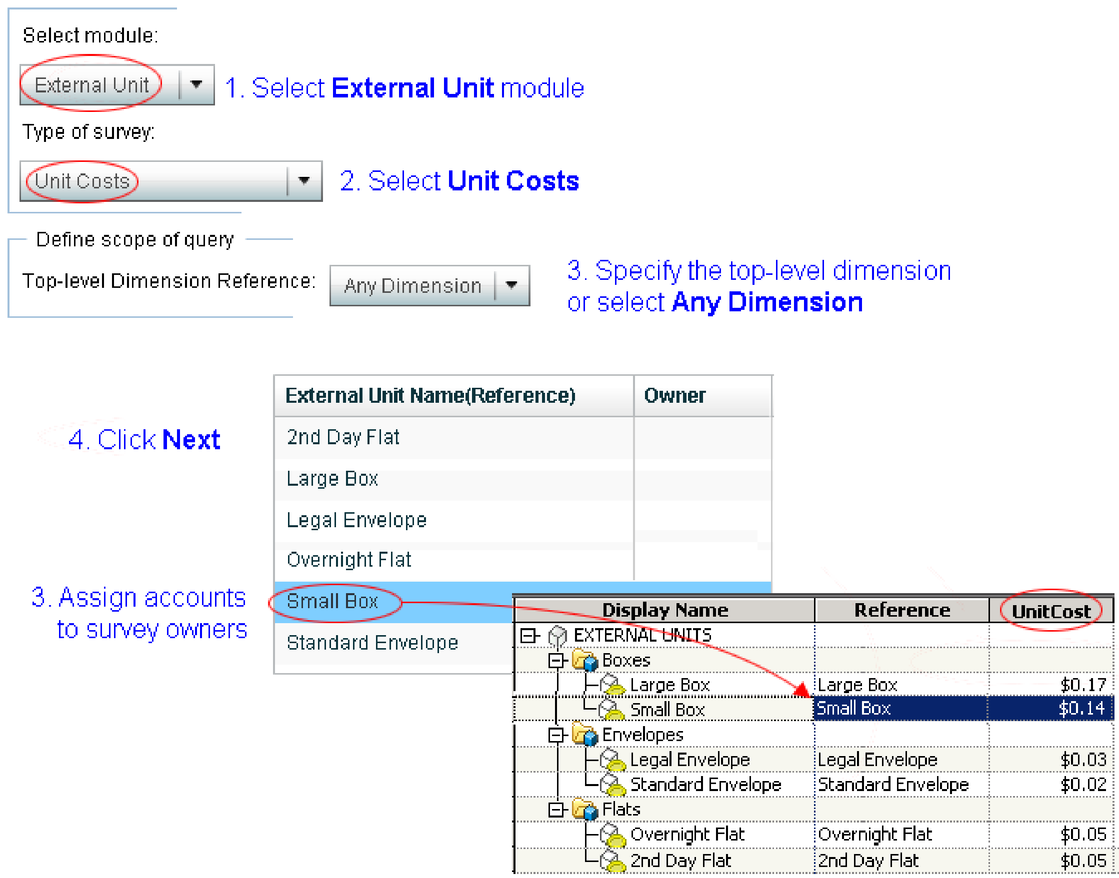
   All External Unit accounts are presented for assignment to survey owners.

   Select a specific dimension

   Only those External Unit accounts under the specified dimension are presented for assignment to survey owners.

   Note: The selection list includes the lowest-level dimension members of the External Units module.
5. Click **Next**.
6. Assign owners to those items that you want to survey.
7. Click **Finish**.

The following picture summarizes the process.



Resource Module: Resource Drivers

With this type of survey, a survey taker updates the following field:

Module	**Type of survey**	**Fields that can be updated**
Resource	Resource Drivers	Assignment.DriverQuantityFixed

1. Follow the general procedure for creating a survey. See “General Procedure” on page 97.
2. Select the **Resource** module.
3. Select **Resource Drivers**.
4. For **Top-Level Dimension Reference**, do one of the following:

   Specify **Any Dimension**
   : All Resource accounts are eligible for assignment to survey owners

   Select a specific dimension
   : Only those Resource accounts under the specified dimension are eligible for assignment to survey owners.

> *Note:* A restricted list of dimension members is displayed in the drop-down list of dimension members available for filtering. The list is based on the first dimension of the Resource module.

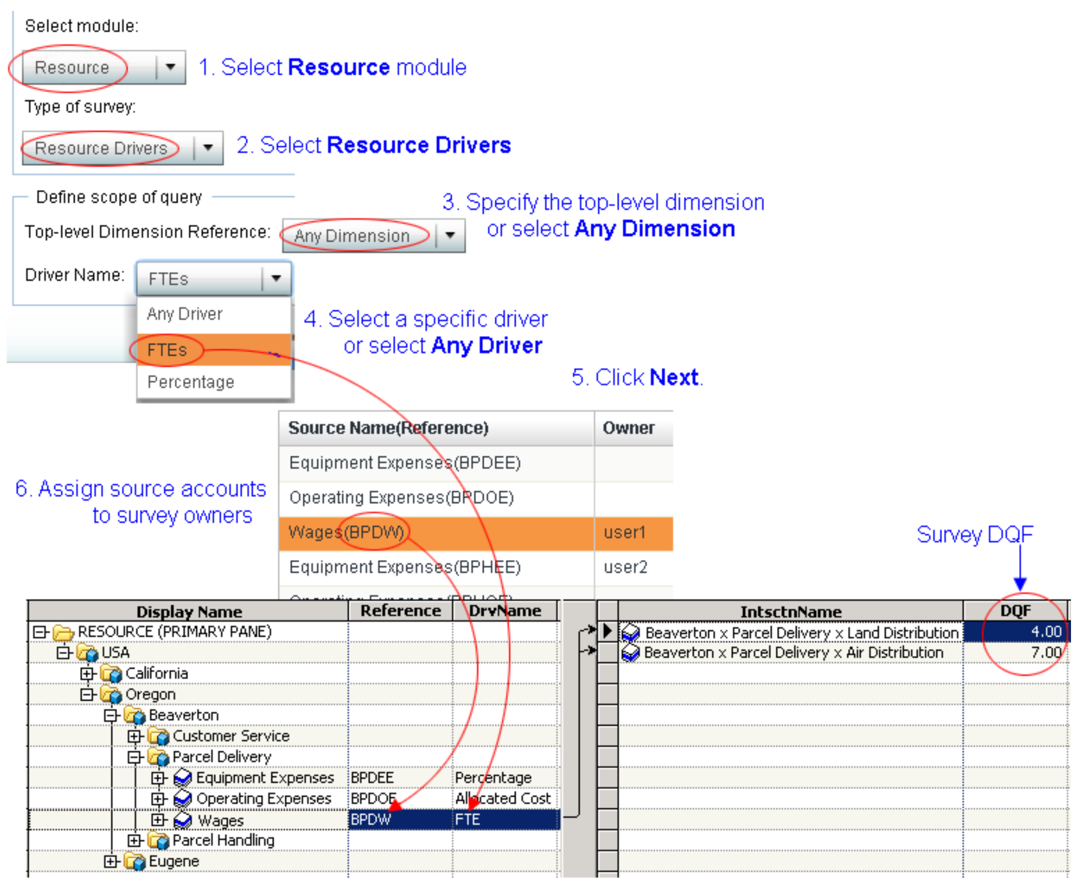
5. For **Driver Name**, do one of the following:

   Specify **Any Driver**
   : All Resource accounts with drivers are presented for assignment to survey owners

   Select a specific driver
   : Only those Resource accounts using the selected driver are presented for assignment to survey owners.

6. Click **Next**.
7. Assign source accounts for the driver to survey owners.

The following picture summarizes the process.



Resource Module: Resource Costs

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
Resource	Resource Costs	EnteredCostElement.EnteredCost

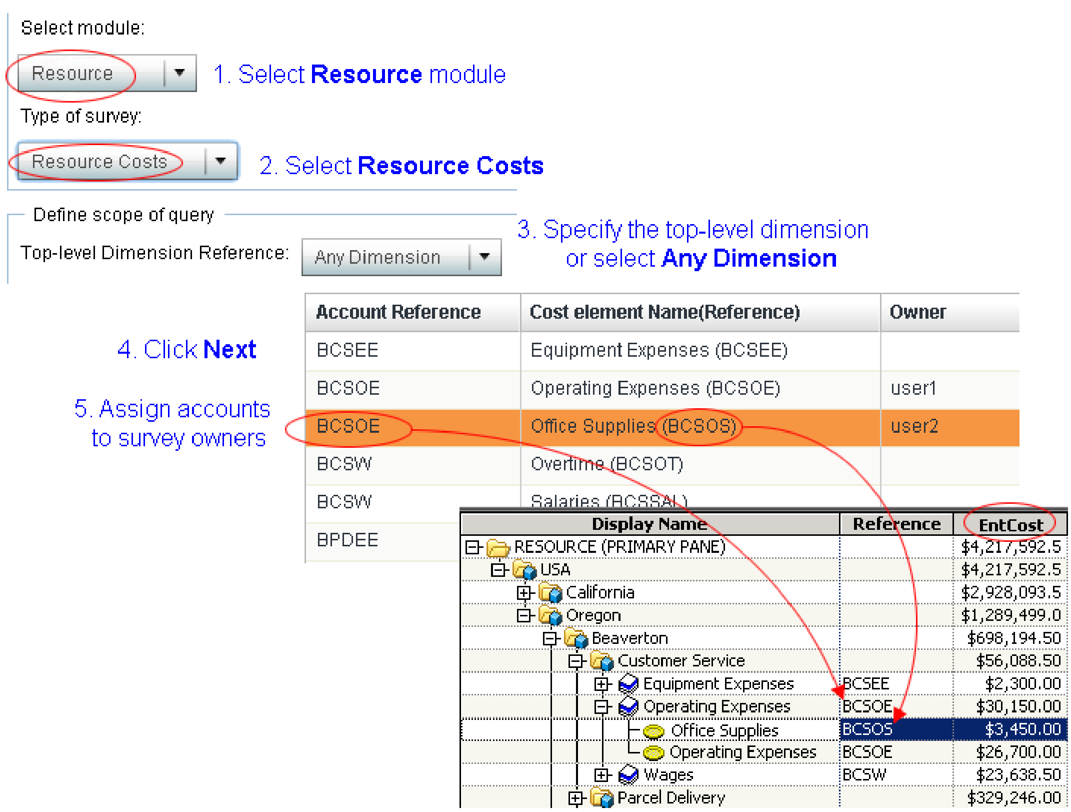
1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **Resource** module.
3. Select **Resource Costs**.
4. For **Top-Level Dimension Reference**, do one of the following:

   Specify **Any Dimension**
   : All Resource accounts are eligible for assignment to survey owners.

   Select a specific dimension
   : Only those Resource accounts under the specified dimension are eligible for assignment to survey owners.

     Note: A restricted list of dimension members is displayed in the drop-down list of dimension members available for filtering. The list is based on the first dimension of the Resource module.
5. Click **Next**.
6. Assign accounts to survey owners.

The following picture summarizes the process.



Resource Module: Numeric Attributes

With this type of survey, a survey taker updates the following field:

Module	**Type of survey**	**Fields that can be updated**
Resource	Numeric Attribute	ValueAttributeAssociation.NumericValue

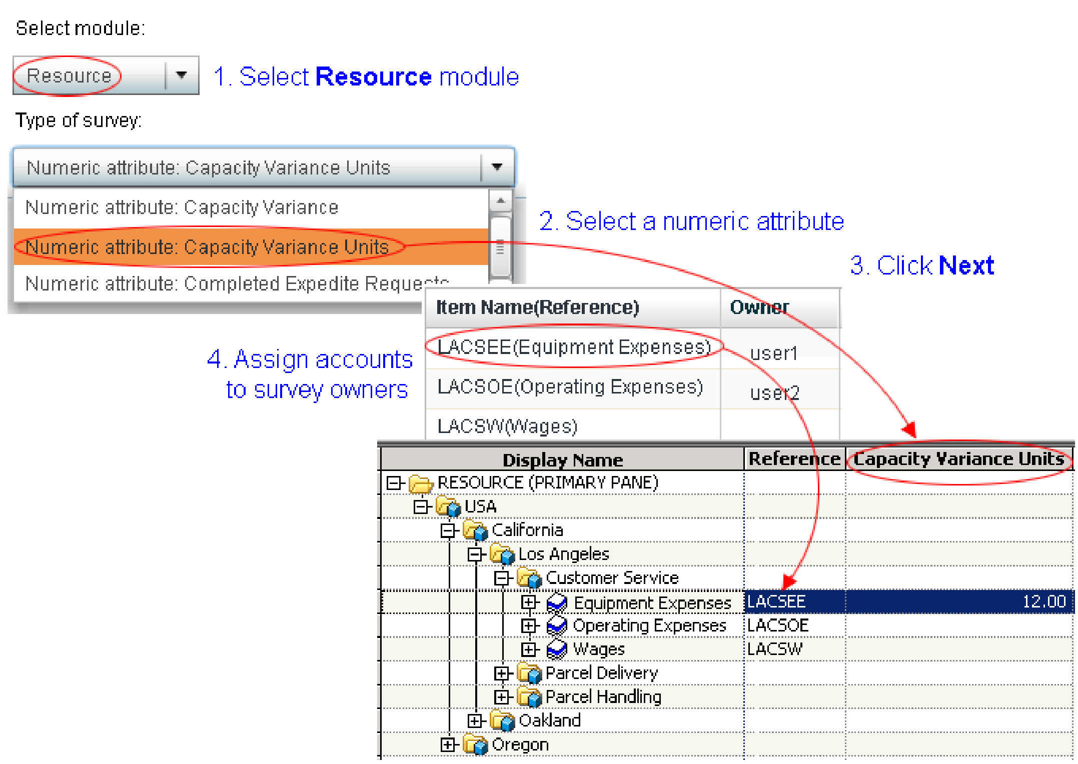
Note: To create a survey for numeric attributes, you must have included the following two tables when you exported survey data for the model:

- ValueAttribute table
- ValueAttributeAssociation table

See "Exporting Survey Data" on page 87.

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **Resource** module.
3. Select a particular attrilbute.
4. Click **Next**.
5. Assign accounts to survey owners.

The following picture summarizes the process.



Activity Module: Activity Drivers

Note: You can only define a survey for Activity Drivers if the **Follow-up driver surveys** flag for the model is off. If the flag is on, then survey items for activity drivers are automatically added to the surveys for Resource Drivers. See "Follow-up Driver Surveys" on page 135.

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
Activity	Activity Drivers	Assignment.DriverQuantityFixed

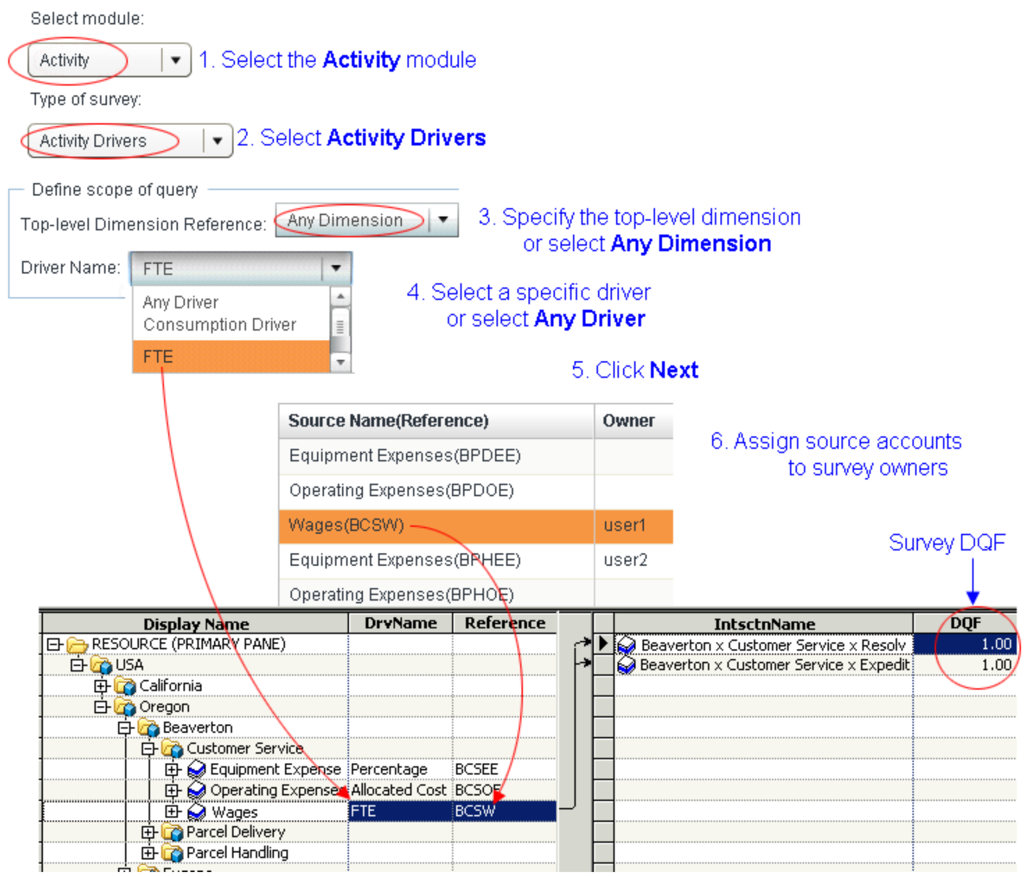
1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **Activity** module.
3. Select **Activity Drivers**.
4. For **Top-Level Dimension Reference**, do one of the following:

   Specify **Any Dimension**
   All Activity accounts are eligible for assignment to survey owners

   Select a specific dimension
   Only those Activity accounts under the specified dimension are eligible for assignment to survey owners.

   Note: A restricted list of dimension members is displayed in the drop-down list of dimension members available for filtering. The list is based on the first dimension of the Resource module.
5. Select a particular driver or select **Any Driver**.
6. Click **Next**.
7. Assign accounts to survey owners.

The following picture summarizes the process.



Activity Module: Numeric Attributes

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
Activity	Numeric Attributes	ValueAttributeAssociation.NumericValue

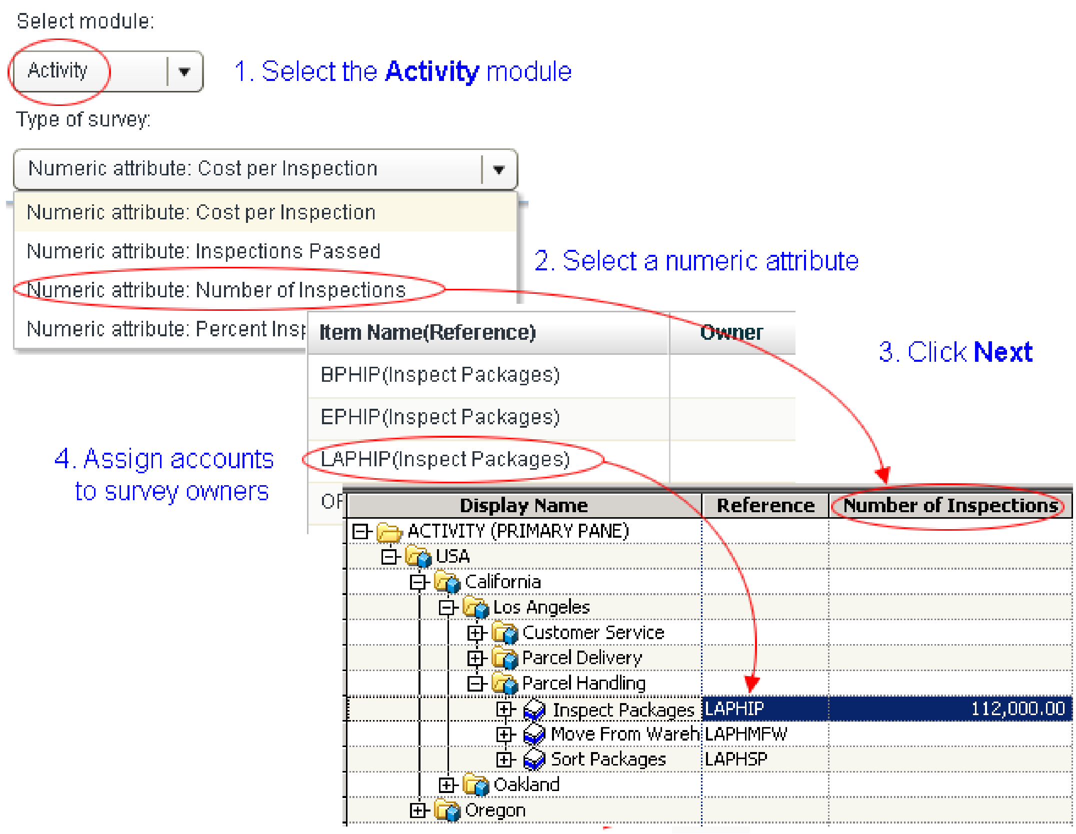
Note: To create a survey for numeric attributes, you must have included the following two tables when you exported survey data for the model:

- ValueAttribute table
- ValueAttributeAssociation table

See "Exporting Survey Data" on page 87.

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **Activity** module.
3. Select a particular attribute.
4. Click **Next**.
5. Assign accounts to survey owners.

The following picture summarizes the process.



Cost Object Module: Cost Object Drivers

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
Cost Object	Cost Object Drivers	Assignment.DriverQuantityFixed

Note: You can only define a survey for Cost Object Drivers if the **Follow-up driver surveys** flag for the model is off. If the flag is on, then survey items for Cost Object Frivers are automatically added to the surveys for Resource Drivers. See “Follow-up Driver Surveys” on page 135.

1. Follow the general procedure for creating a survey. See “General Procedure” on page 97.
2. Select the **Cost Object** module.
3. Select **Cost Object Drivers**.
4. For **Top-Level Dimension Reference**, do one of the following:

   Specify **Any Dimension**

   All Cost Object accounts are eligible for assignment to survey owners.

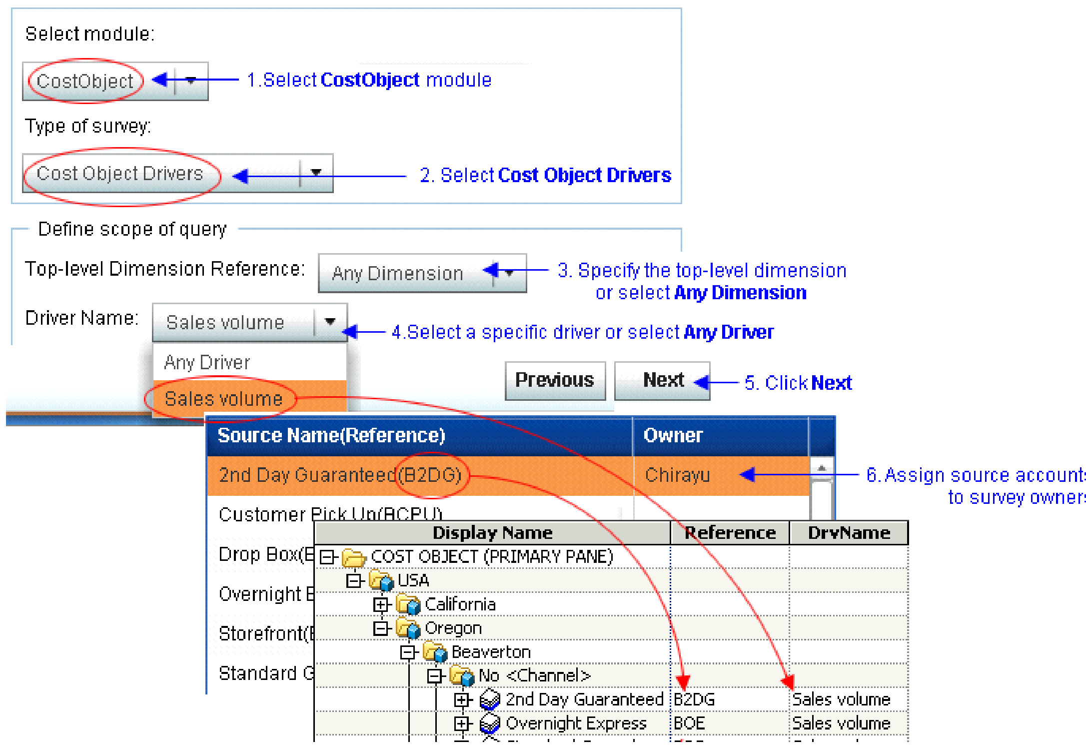
   Select a specific dimension

   Only those Cost Object accounts under the specified dimension are eligible for assignment to survey owners.

Note: A restricted list of dimension members is displayed in the drop-down list of dimension members available for filtering. The list is based on the first dimension of the Resource module.

5. Select a particular driver or select **Any Driver**.
6. Click **Next**.
7. Assign accounts to survey owners.

The following picture summarizes the process.



Cost Object Module: Revenues and Sold Quantities

With this type of survey, a survey taker updates the following fields:

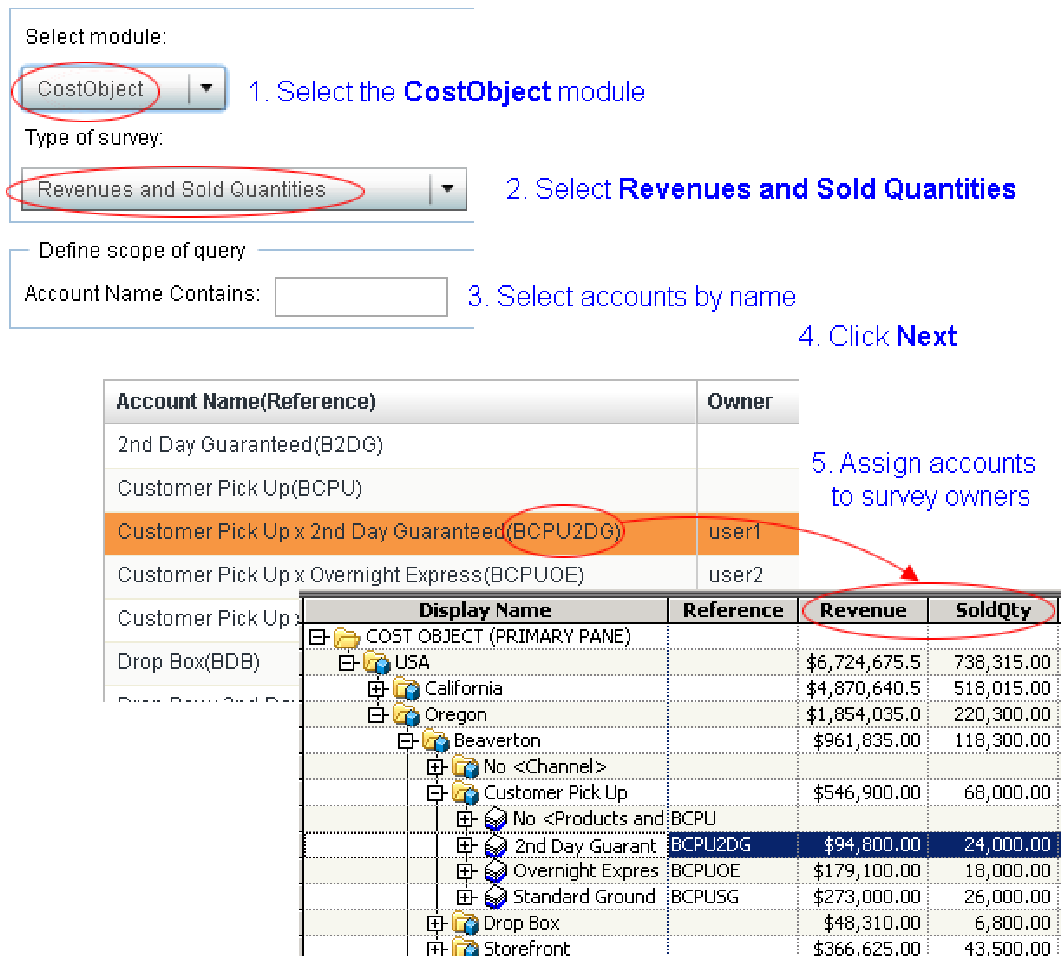
Module	Type of survey	Fields that can be updated
Cost Object	Revenues and Sold Quantities	Account.Revenue Account.SoldQuantity

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.
2. Select the **Cost Object** module.
3. Select **Revenues and Sold Quantities**.
4. For **Account Name Contains**, you can limit the accounts to survey by specifying a character string that each account's name must contain.

The character string can contain blanks, and case does not matter. The string can occur at the beginning, in the middle, or at the end of the account name.

5. Click **Next**.
6. Assign accounts to survey owners.

The following picture summarizes the process.



Cost Object Module: Output Quantities

With this type of survey, a survey taker updates the following field:

Module	**Type of survey**	**Fields that can be updated**
Cost Object	Output Quantities	Account.OutputQuantityUE

1. Follow the general procedure for creating a survey. See “General Procedure” on page 97.
2. Select the **Cost Object** module.
3. Select **Output Quantities**.

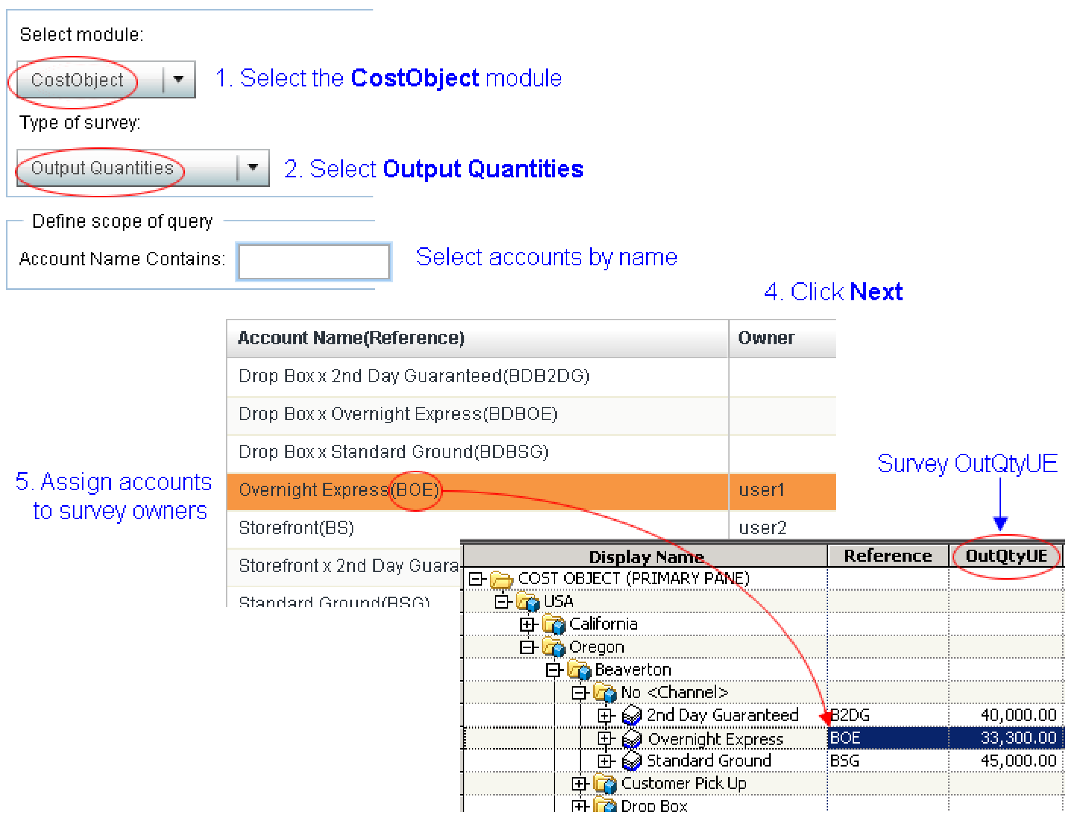
4. For **Account Name Contains**, you can limit the accounts to survey by specifying a character string that each account's name must contain.

   The character string can contain blanks, and case does not matter. The string can occur at the beginning, in the middle, or at the end of the account name.

5. Click **Next**.

6. Assign accounts to survey owners.

The following picture summarizes the process.



Cost Object Module: Numeric Attributes

With this type of survey, a survey taker updates the following field:

Module	Type of survey	Fields that can be updated
Cost Object	Numeric Attribute	ValueAttributeAssociation.NumericValue

Note: To create a survey for numeric attributes, you must have included the following two tables when you exported survey data for the model:

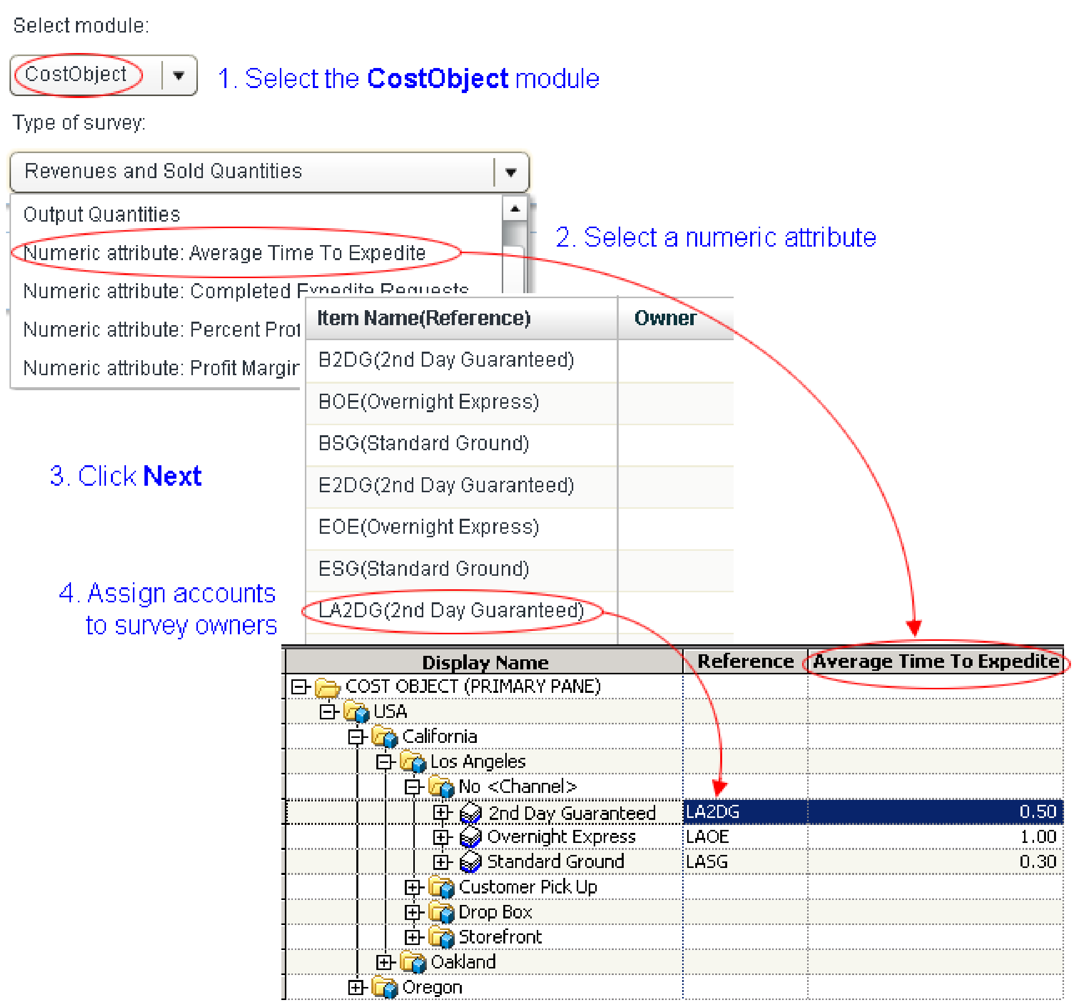
- ValueAttribute table
- ValueAttributeAssociation table

See "Exporting Survey Data" on page 87.

1. Follow the general procedure for creating a survey. See "General Procedure" on page 97.

2. Select the **Cost Object** module.
3. Select a particular attribute.
4. Click **Next**.
5. Assign accounts to survey owners.

The following picture summarizes the process.



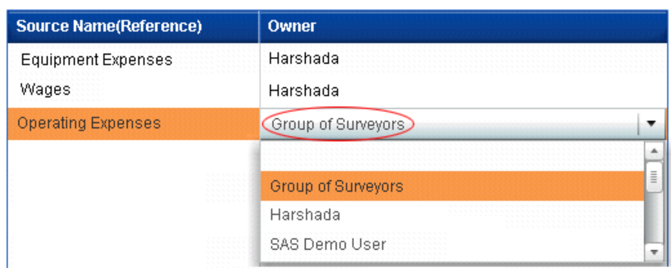
Create a Group Survey

In addition to assigning a survey to an individual survey taker, you can assign it to a group. Then, when all the members of the group have completed the survey, their individual responses are automatically added together and entered into the database.

You can assign only driver surveys (Resource, Activity, and Cost Object) to a group. And, you can assign any type of driver survey to a group except for Percentage drivers. This means that if you select **Any Driver** for the driver name, and if any of the drivers to be assigned is a Percentage driver, then you can not assign the survey to a group.

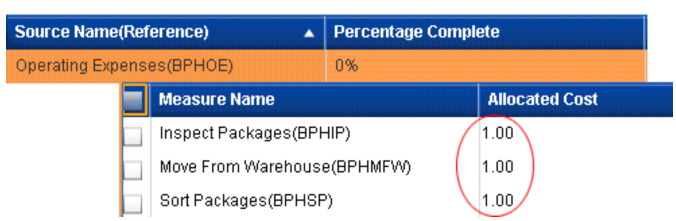
As an example, suppose there is a group of survey takers (named Group of Surveyors) with two members: Ashwin and Chirayu. For information about groups, see "Creating Groups" on page 12. Also see "Survey Takers" on page 25.

Suppose also that as an administrator you create a Resource Driver survey for a particular driver and assign the Operating Expenses account to the group.

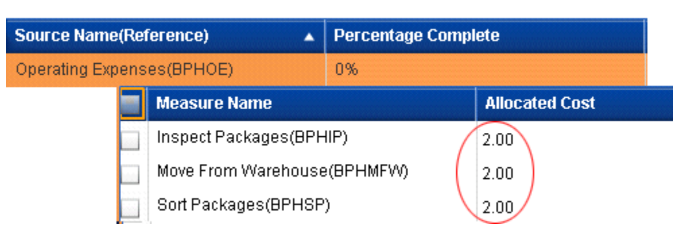


Then, both Ashwin and Chirayu will receive an e-mail informing them that they have been assigned a survey.

Suppose Ashwin takes the survey, and for each destination account assigns a quantity of 1.



Suppose also that when Chirayu takes the survey, he assigns a quantity of 2 to each of the same destination accounts.



The Driver Quantity Fixed that is written to the Assignment staging table is 3—the sum of the quantities entered by both Ashwin and Chirayu.

Create Follow-up Surveys

If you select the model preference, **Follow-up driver surveys**, then whenever you create a Resource Driver survey an Activity Driver survey is automatically created and assigned to the same survey takers as the Resource Driver survey.

For more information, see "Follow-up Driver Surveys" on page 135.

Create an Aggregated Survey

In an *aggregated survey* the Driver Quantity Fixed (DQF) that is written to a staging table is calculated (aggregated) according to a formula. The formula allows you to fix a weight to the relative contributions to the DQF of the participants who determine the DQF.

Note: Do not confuse aggregated surveys with calculated drivers. Unlike a driver formula, the formula in an aggregated survey is not attached to the driver whose DQF is being surveyed. The formula exists only when the survey is being taken.

You can create an aggregated survey for the following types of drivers:

- basic
- bill of cost
- weighted

What these driver types have in common is that their Driver Driven Cost for each assignment is determined by the ratio of the DQF for each assignment relative to the total DQF for all assignments—not by the absolute value of each DQF. The formula for an aggregated survey calculates the relative contribution to the DQF—not the absolute value of the DQF.

An aggregated survey allows one survey taker to answer for multiple people and to assign a weight to the contribution of each type of person.

Aggregated surveys are best explained through an example. The following text describes one such survey in detail.

1. Add attributes to a user. For information, see "Add Attributes to a User" on page 120.

   Note: The attributes that you add to a user for surveys have nothing to do with the attributes that you add to accounts in SAS Activity-Based Management. The attributes that you add to a user for surveys are not written to any staging table.

   a. Click **User Manager**.

   b. Click **Users**.

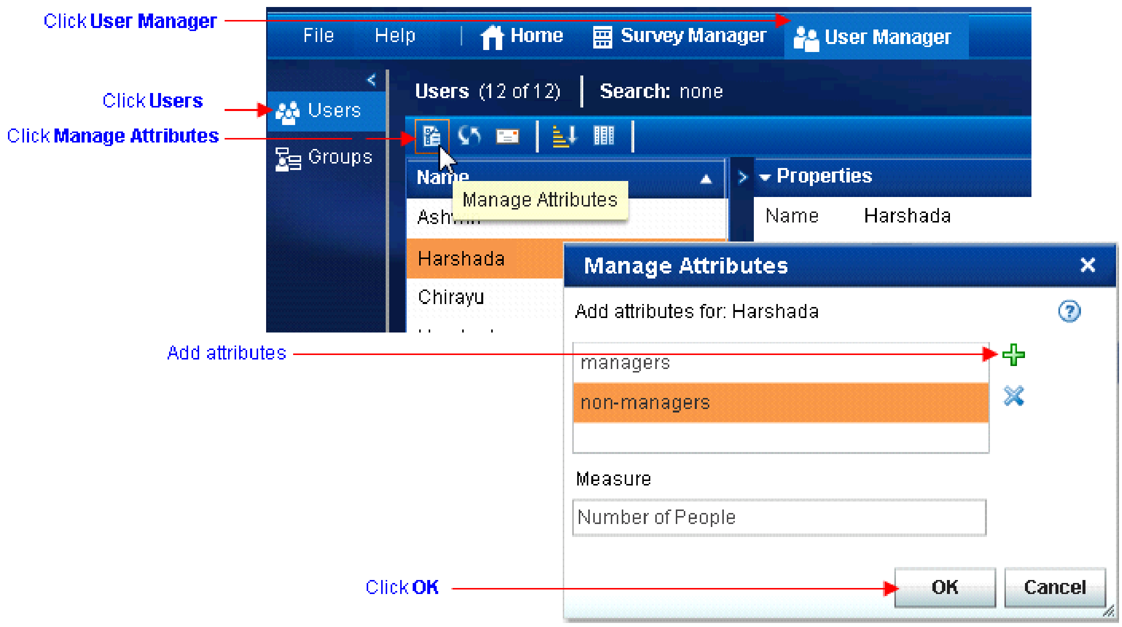
   c. Click **Manage Attributes**.

   d. Add attributes.

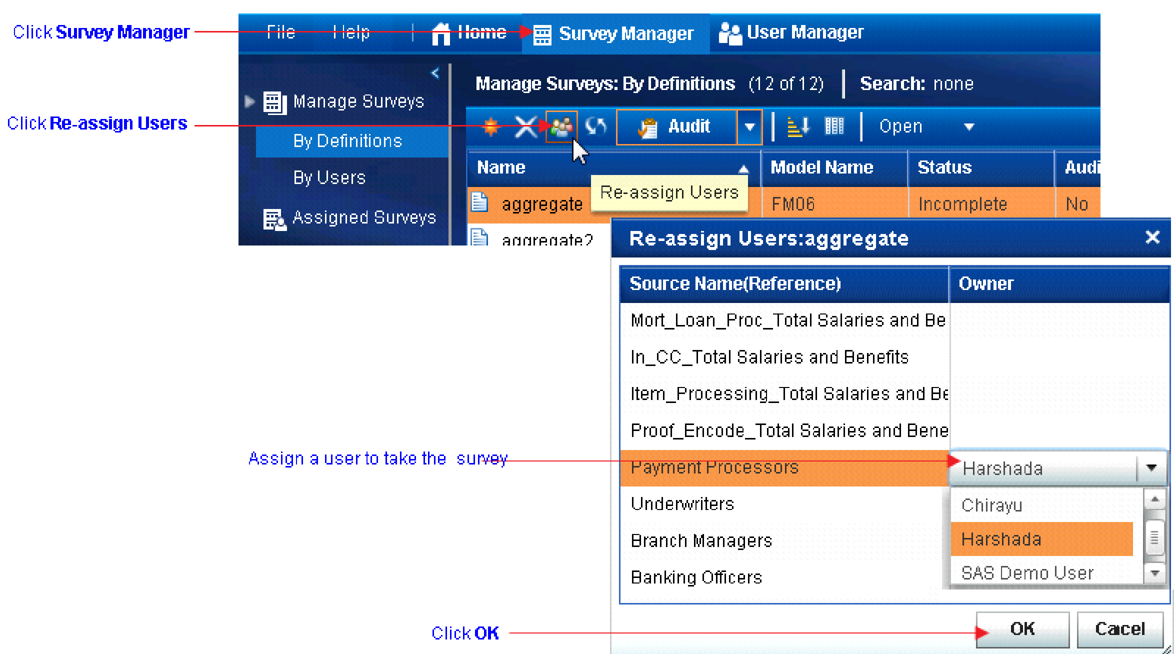
      Note: The number of attributes that can be added is limited by the application preference **Maximum Attributes**. See "Application Preferences" on page 132.

e. Click **OK**.

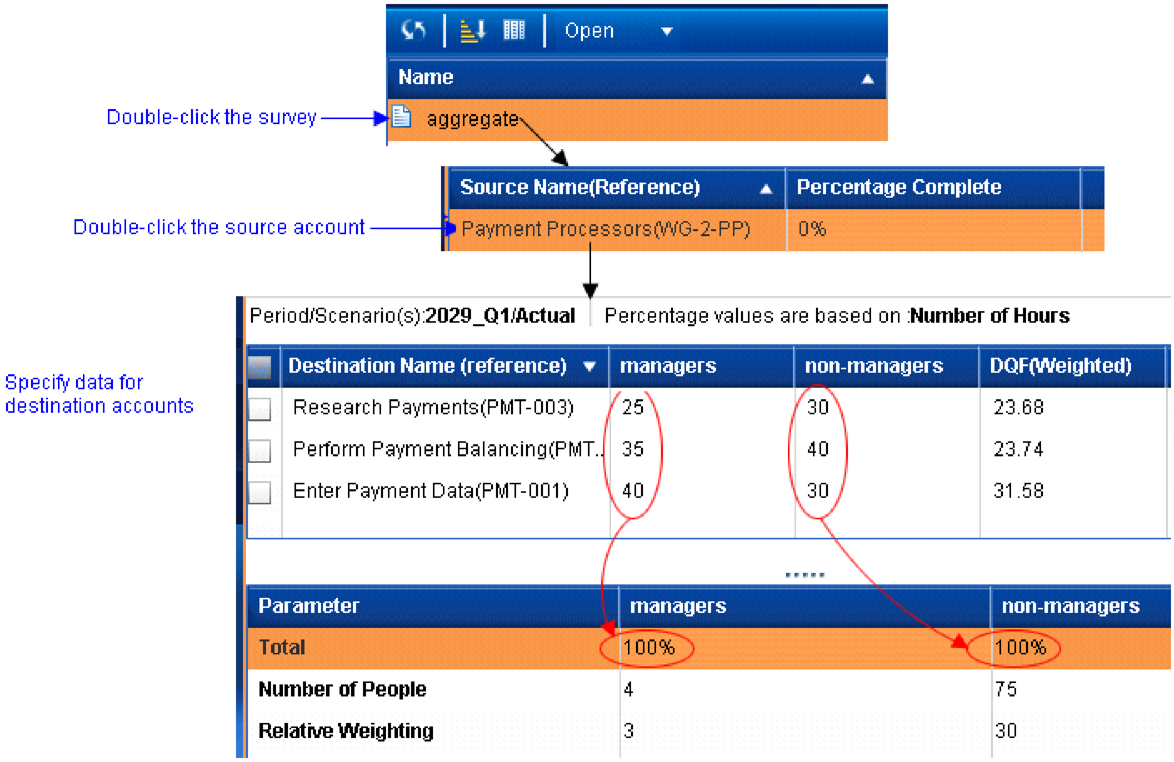
The following picture shows how to add two attributes, *managers* and *non-managers*, to the user Harshada.



2. Assign a survey to the user with the attributes.



3. The user takes the survey.

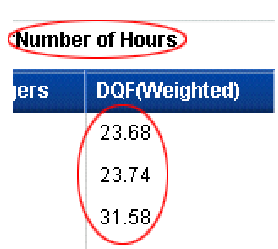


This picture shows the user, Harshada, entering the following values for the percent of DQF accounted for by managers and non-managers, respectively:

Account Name (Reference)	**Percent of DQF accounted for by managers**	**Percent of DQF accounted for by non-managers**
Enter Payment Data(PMT-001)	25	30
Perform Payment Balancing(PMT-002)	35	40
Research Payments(PMT-003)	40	30
Total percent:	100	100

There are two things to notice about these numbers:

- They are entered as percentages regardless of the type of driver. However, the resulting DQF that is calculated by the system represents the unit of measure for the driver being surveyed. In the case of this example, the unit of measure is Number of Hours. It is this number that is stored in the Assignment staging table.



- The total percentage equals 100 in the case of both managers and non-managers. Whether the total must equal 100% or not is determined by the application preference, **Enforce hundred percent for percentage values**. See “Application Preferences” on page 132.

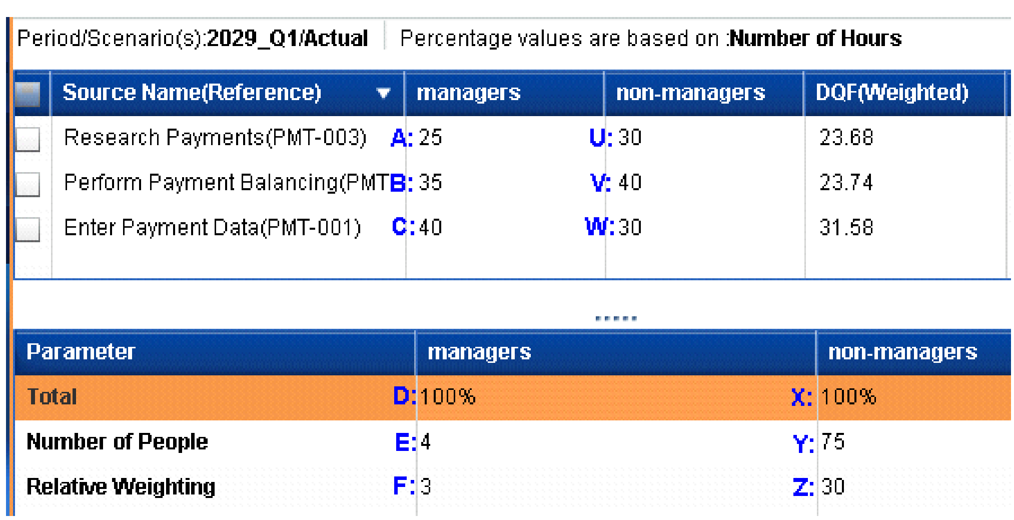
The previous picture also shows a user entering the following values for the number of managers vs. non-managers and the relative weighting to be allocated to managers vs. non-managers.

Parameter	Managers	Non-managers
Number of people	4	75
Relative weighting	3	30

To explain the formula that is used to derive the resulting DQF, we can label the cells in the previous two tables as follows:

Account Name (Reference)	Percent of DQF accounted for by managers	Percent of DQF accounted for by non-managers
Enter Payment Data(PMT-001)	A: 25	U: 30
Perform Payment Balancing(PMT-002)	B: 35	V: 40
Research Payments(PMT-003)	C: 40	W: 30
Total percent:	D: 100	X: 100
Number of people	E: 4	Y: 75
Relative weighting	F: 3	Z: 30

The following picture shows the labelling



The formula used to derive the resulting DQF for the first account (Enter Payment Data) is the following:

```
((A/D)*E*F) + ((U/X)*Y*Z)
```

```
((25/100)*4*3) + ((30/100)*75*30)=3+675=678
```

divided by:

```
(E*F)+(Y*Z)
```

```
(4*3)+(75*30)=12+2250=2262
```

times:

```
 ((A/D)*E) + ((U/X)*Y)
+((B/D)*E) + ((V/X)*Y)
+((C/D)*E) + ((W/X)*Y)
```

```
((25/100)*4) + ((30/100)*75)=1+22.5=23.5
+((35/100)*4) + ((40/100)*75)=1.4+30=31.4
+((45/100)*4) + ((30/100)*75)=1.6+22.5=24.1
23.5 + 31.4 + 24.1=79
```

or:

```
(678/2262)*79)=23.679045
```

The result is rounded to the number of decimal places as specified in the application preference, **Maximum Decimal Places**. See "Application Preferences" on page 132. In the case of this example, it is rounded to 23.68.

The resulting DQF is calculated in similar fashion for the other two accounts. The resulting DQF for the three assignments are:

- `23.68`
- `23.74`
- `31.58`

And the total Driver Quantity Calculated is `23.68+23.74+31.58=79`.

So, the Driver Rates for the three assignments are:

- `23.68/79=0.2997`
- `23.74/79=0.3005`
- `31.58/79=0.3997`

Chapter 8

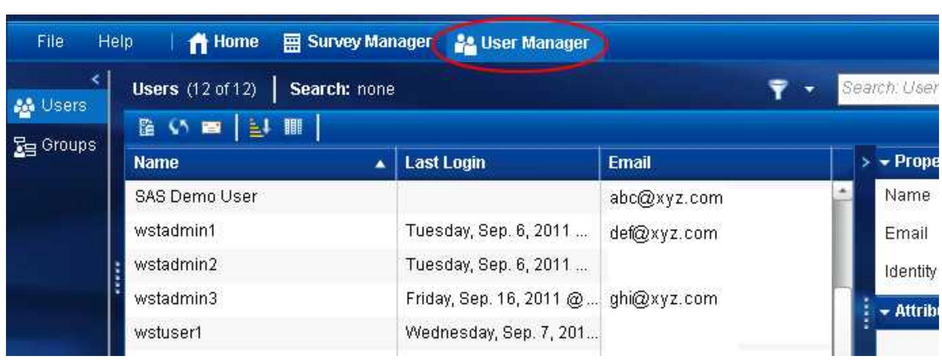
Manage Users

Overview

The User Manager is available only to users who have Create Model capability. There is no Administer Surveys capability. For a discussion of roles and capabilities, see "Creating Roles" on page 4.

Use the User Manager to

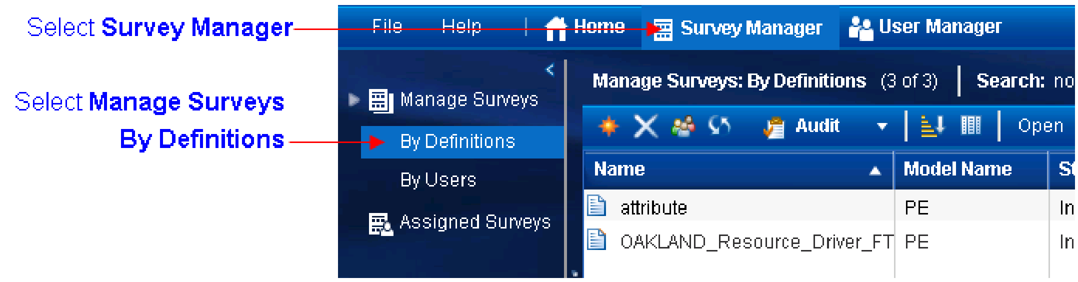
- Reassign survey items to users. See "Reassign Users" on page 119.
- Manage user attributes. See "Add Attributes to a User" on page 120.
- Send e-mails to individual users. See "Send an e-mail" on page 126.



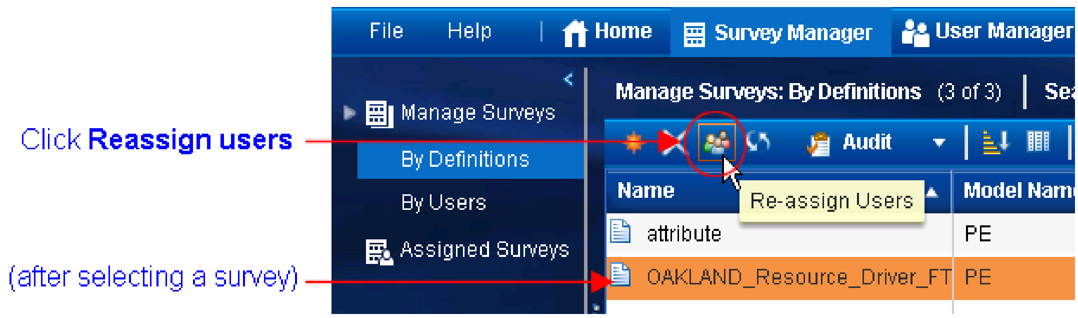
Reassign Users

To reassign the fields in a survey to a different user:

1. Select **Survey Manager**.
2. Select **Manage Surveys** ⇨ **By Definitions**.



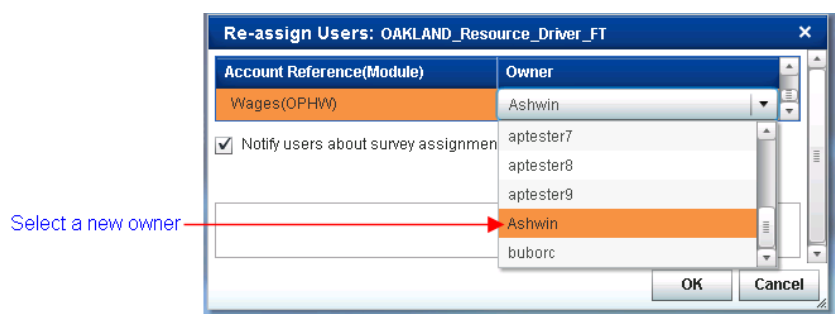
3. Select a survey.
4. Click the **Re-assign users** icon.



The Re-assign Users window appears.

5. Select a user from the drop-down list and click **OK**.

Note: Select **Notify users about survey assignments** to send an e-mail to the new user.

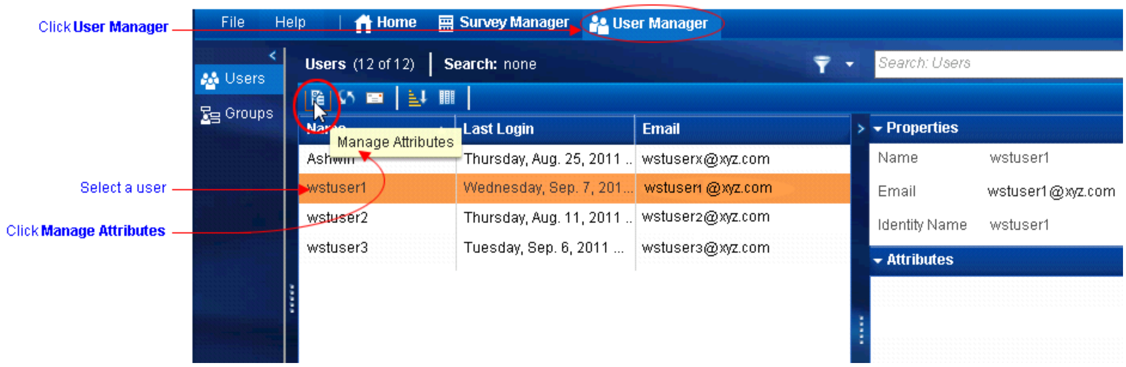


Add Attributes to a User

You can add attributes to a survey taker so that the survey taker can take an aggregated survey. See “Create an Aggregated Survey” on page 113.

To add or delete attributes for a user:

1. Click **User Manager**.
2. Select a user.
3. Click **Manage Attributes**.



4. The Manage Attributes window appears.

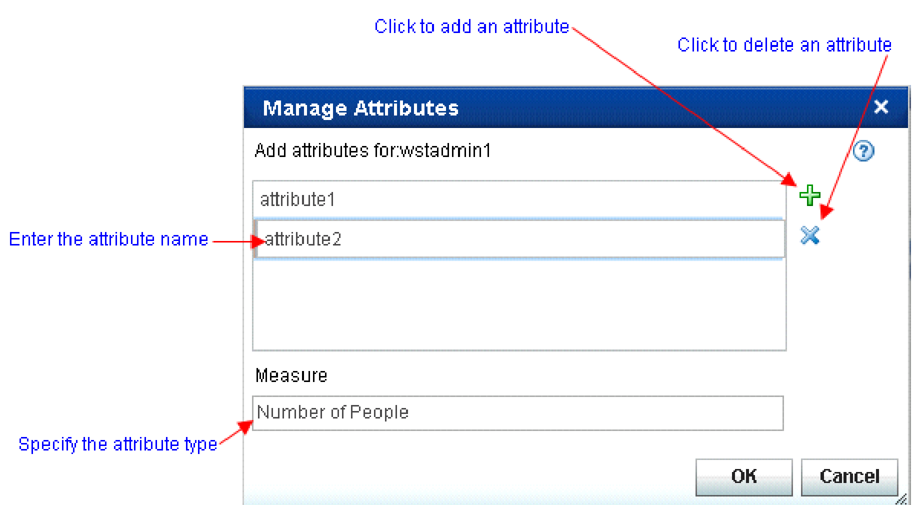
   a. Click the plus button ( ) to add an attribute. Then enter the attribute name.

   b. Select an attribute and click the X button))to delete an attribute.

   c. Specify the **Measure Name**.

      Note: What you specify for the name does not make any difference in how the DQF is derived for an aggregated survey—it is only for your own documentation.

   d. Click **OK**.



Chapter 9

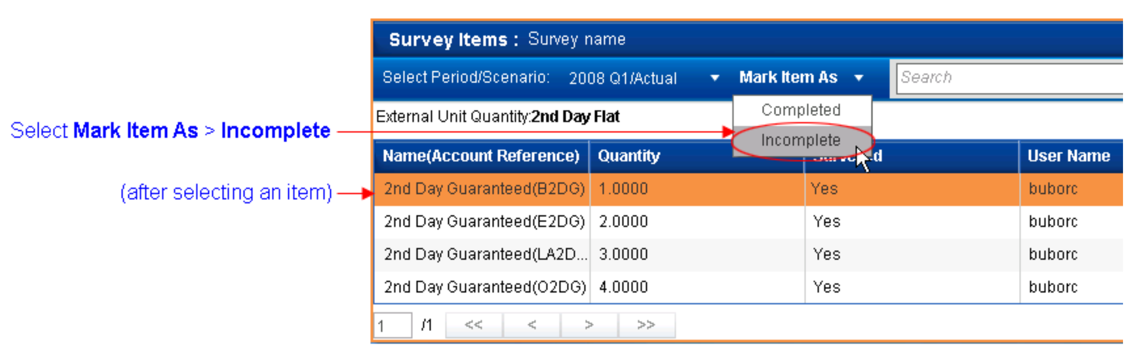
Administer Surveys

Mark a Survey as Incomplete

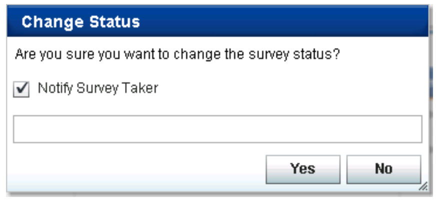
When a survey taker submits a survey, the survey is marked as Completed. The administrator is notified that the survey has been submitted, and the survey taker can no longer modify any responses. Once the administrator is notified, the administrator can mark survey items as Incomplete so that the survey taker can change the responses and resubmit the survey.

To mark a survey item as Incomplete:

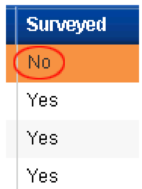
1. Open the survey.
2. Select a survey item.
3. Select **Mark Item As** ⇨ **Incomplete**.



You are asked to confirm the status change, and you can choose to notify the survey taker.



If you click Yes, the **Surveyed** status changes to No.



Once the **Surveyed** status is No, the survey taker can retake the survey for that item.

Note: You can also mark a survey item as Completed, in which case the survey taker can no longer make changes to that item.

Audit Surveys

When an administrator selects the model option to allow a survey taker to add new destination accounts to a driver survey, and a user adds a new destination account, the administrator must either approve or disapprove the addition. See “Allow a Survey Taker to Add New Accounts” on page 134.

- If the administrator approves, then the new assignment values, as entered by the survey taker, are written to the Assignment staging table in the database.
- If the administrator disapproves, then the new assignment is removed from the database staging table—it is as if the assignment was never made.

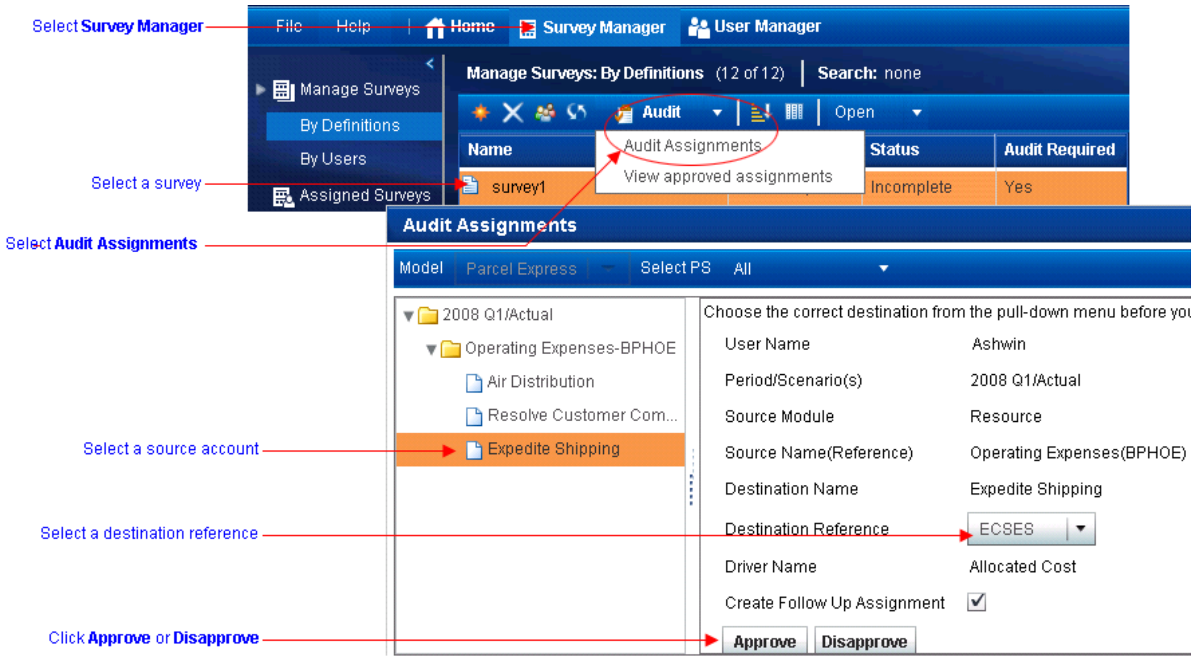
Note: An administrator cannot subsequently change a decision to approve or disapprove. Disapproving a user’s addition has the same effect as though the survey taker had never made the new assignment at all.

To audit a survey:

1. Select the **Survey Manager**.
2. Select a survey to audit.
3. Select **Audit** ⇨ **Audit Assignments**.
4. Select a source account.
5. Select the reference of the destination account.
6. Click either **Approve** or **Disapprove**.

   Once all the destination accounts have been approved, the **Audit Required** column changes to No.

7. Click **Close**.



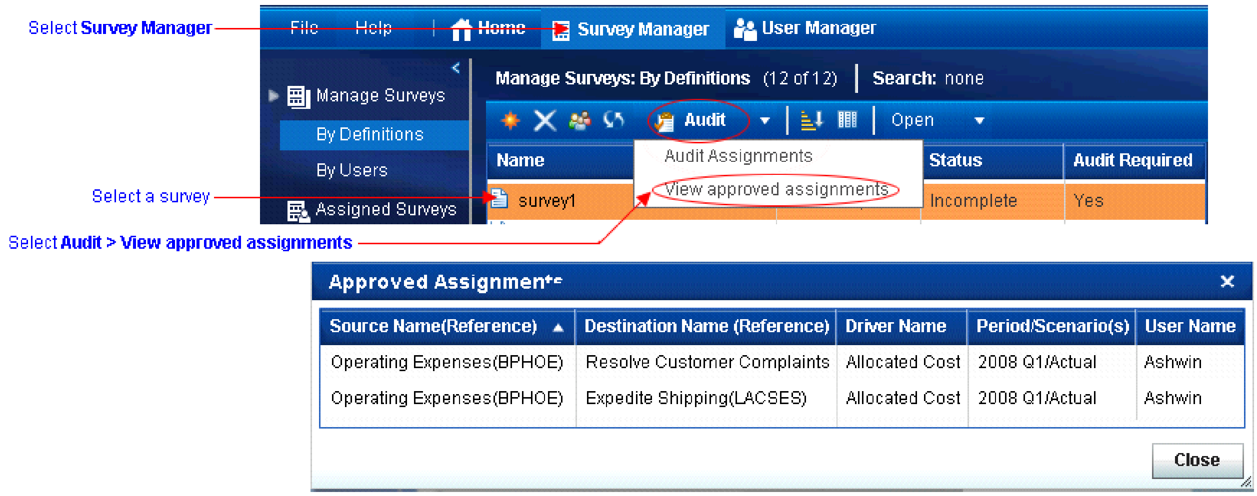
View Approved Assignments

As an administrator, you can view approved assignments, but you cannot reverse a decision to disapprove them. See “Audit Surveys” on page 124.

Note: You cannot view disapproved assignments. They are as though the survey taker never made them.

To view approved assignments:

1. Select the **Survey Manager**.
2. Select a survey.
3. Select **Audit** ⇨ **View approved assignments**.
4. Click **Close**.



Send an e-mail

As an administrator, you can send an e-mail to a survey taker in three different ways:

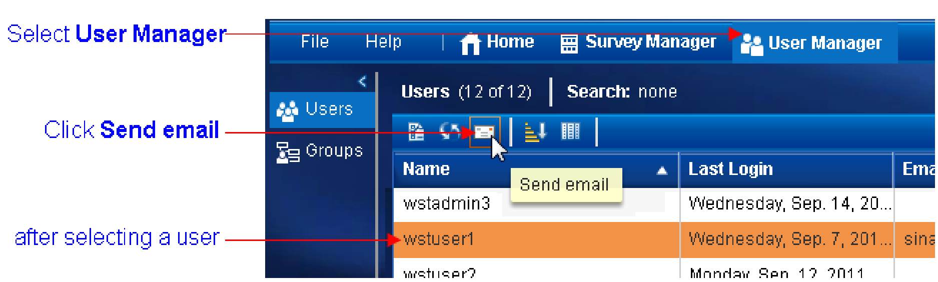
- To a selected user
- To a new survey taker
- To a reassigned survey taker

To a selected user

To send an e-mail to a selected user:

1. Select **User Manager**.
2. Select a user—the recipient of the e-mail.
3. Click **Send e-mail**.

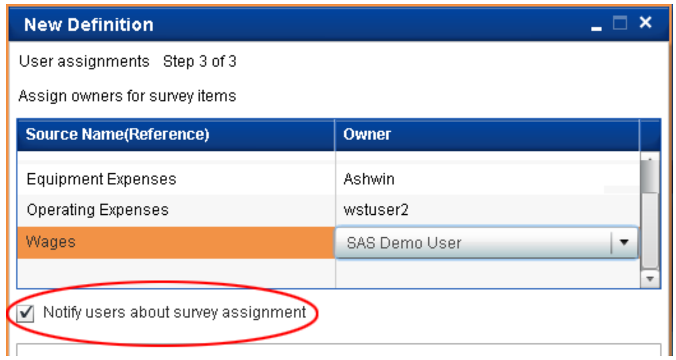
   Note: The user's e-mail address must be stored in the SAS metadata server. See "Creating Users" on page 17.



To a new survey taker

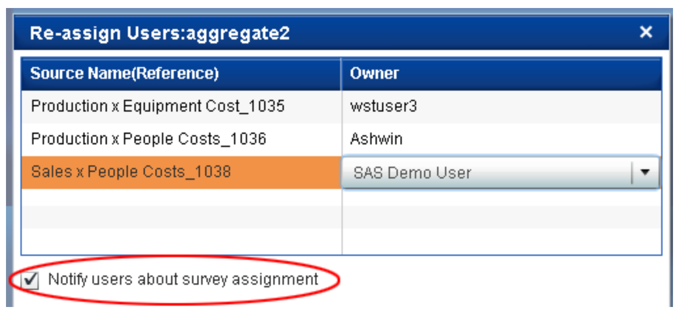
You can send an e-mail to each survey taker who is assigned a survey item when you create a survey.

Select **Notify users about survey assignment** on the New Definition – User Assignments window.



To a reassigned survey taker

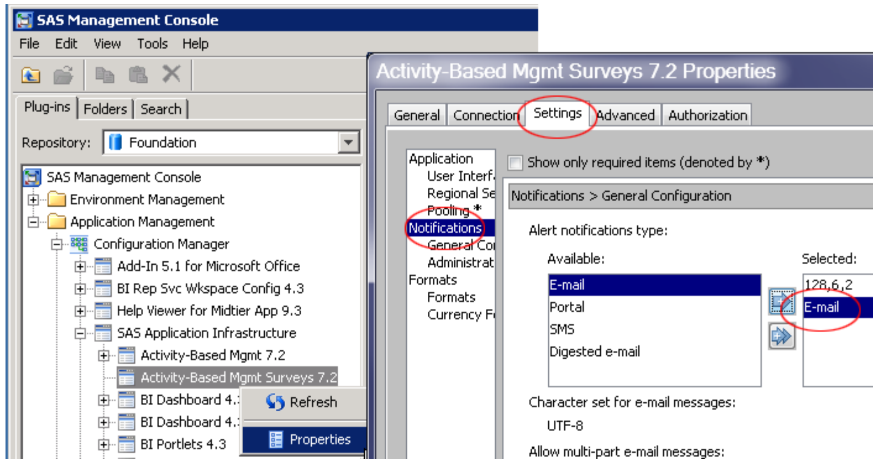
You can send an e-mail when a survey item is reassigned to a different survey taker. Select **Notify users about survey assignment** on the Re-assign Users window. See "Reassign Users" on page 119.



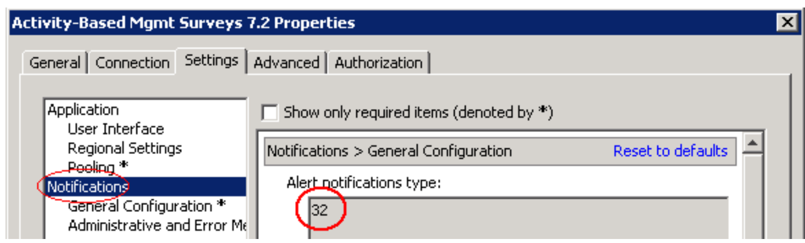
Enable E-mail Notification

To enable e-mail notification for SAS Activity-Based Management Surveys, do the following:

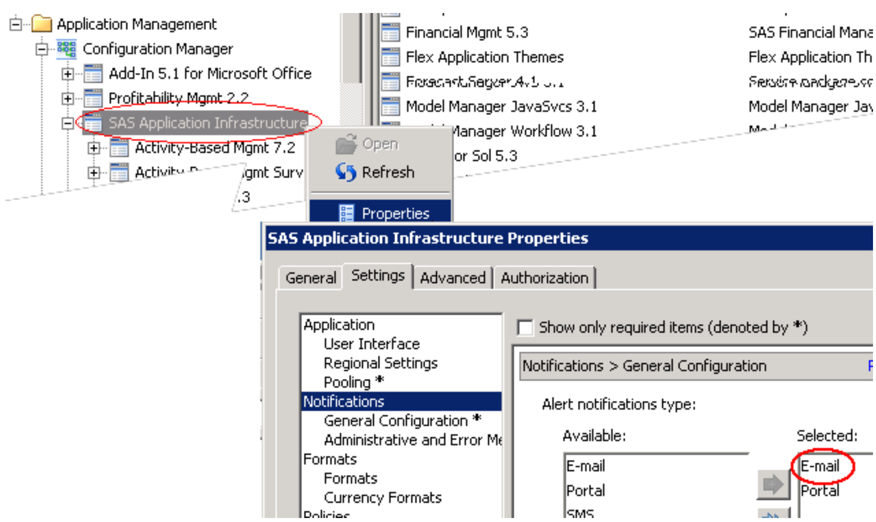
1. Log on to SAS Management Console as an administrator, and access the SAS Activity-Based Management middle-tier server.
2. Click the **Plug-Ins** tab.
3. Expand **Application Management**.
4. Expand **Configuration Manager**.
5. Expand **SAS Application Infrastructure**.
6. Right-click **Activity-Based Mgmt Surveys 7.2** and select **Properties**.
7. Click the **Settings** tab on the Properties window.
8. Select **Notifications**.
9. Ensure that one of the following is true:
   - **E-mail** is selected for Alert Notifications Type, as shown in the following picture.



- **32** is selected for Alert Notifications Type, as shown in the following picture.



Note: 32 indicates that E-mail has been selected at the SAS Application Infrastructure level (as shown in the following picture) so as to apply to all child components, including Activity-Based Management Surveys.

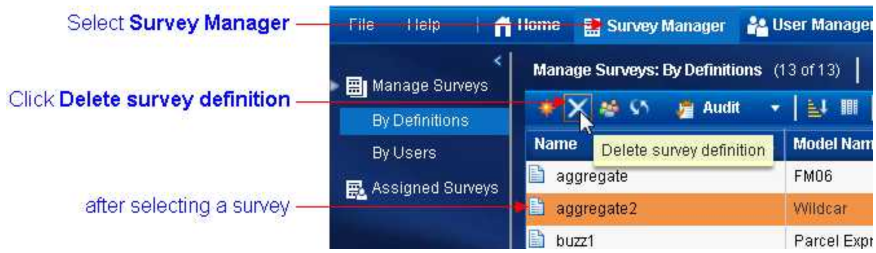


Delete a Survey

To delete a survey:

1. Select **Survey Manager**.
2. Select a survey.
3. Click **Delete survey definition**.

Note: Any data already written to staging tables is not rolled back.



Chapter 10

Survey Preferences

Overview

To select preferences while working as an administrator, select **Files** ⇨ **Survey Preferences**.

You can select two types of preferences:

Application Preferences
These preferences apply to every administrator and to every survey taker.

Model Preferences
These preferences apply to every administrator and to every survey taker for a particular model.

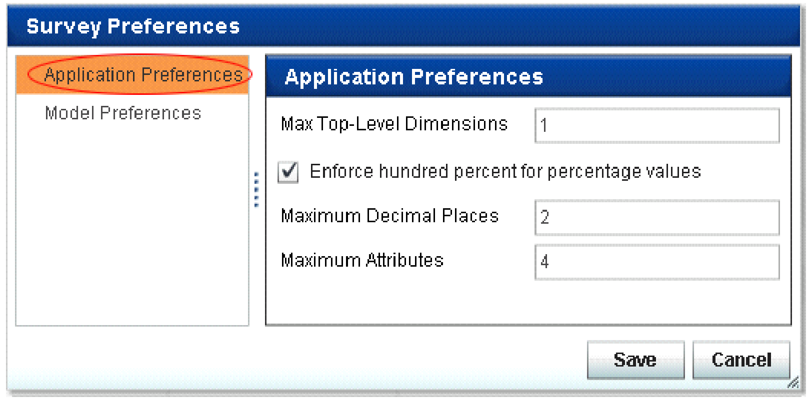
Note: For both types of preferences, you must log off and back on for the preferences to take effect fully.

Both administrators and survey takers can set general preferences for the user interface by selecting **File** ⇨ **Preferences**. These preference options are self-explanatory.

Note: For the global preference, User locale, you must log off and back on for the preference to take effect fully.

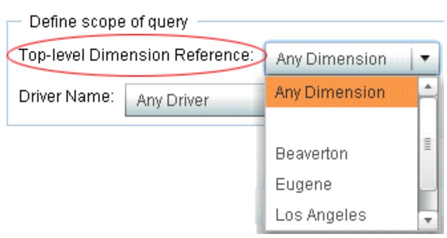
The following topics describe Application and Model preferences in detail.

Application Preferences



You can set the following Application preferences:

Max Top-Level Dimensions
specifies the number of entries in the **Top-level Dimension Reference** drop-down list. This drop-down list is presented when you create a survey.



Enforce hundred percent for percentage values
specifies both of the following:

- When a user takes a driver survey involving a Percentage driver, the user's responses must total 100%.
- When a user takes an aggregated survey, percentage contributions to Driver Quantity Fixed (DQF) must total 100%. See "Create an Aggregated Survey" on page 113.

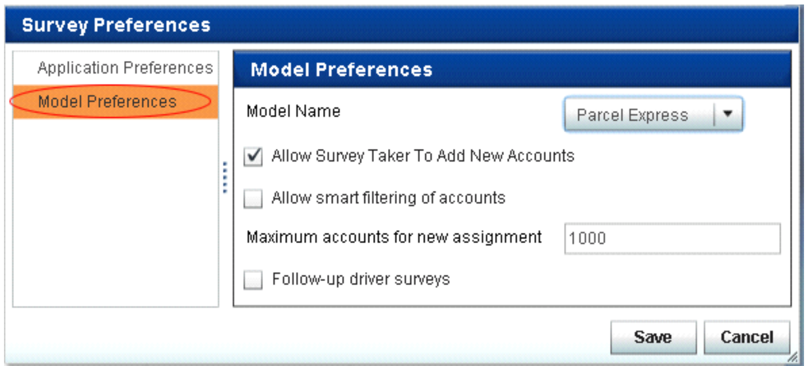
Maximum Decimal Places
specifies the maximum number of decimal places that are accepted from survey taker responses. This option does not prevent a survey taker from entering more decimal places, but it determines how many decimal places are stored in the database.

Maximum Attributes
specifies how many attributes can be added to any particular survey taker for taking an aggregated survey. See "Create an Aggregated Survey" on page 113.

Note: If you have already added attributes to users and then lowered the number of maximum attributes, attributes that exceed the maximum are not automatically removed from users. The preference applies only to the subsequent setting of attributes.

Model Preferences

Overview

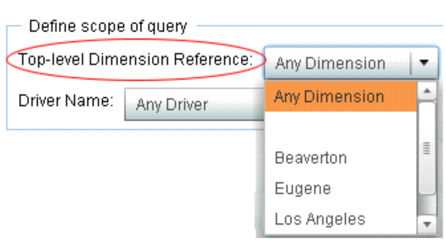


You can set the following Model preferences:

Model Name
specifies the model to which the following preferences apply.

Allow Survey Taker To Add New Accounts
allows survey takers, when taking a driver survey, to create assignments to additional accounts. The survey taker can then take a survey for those new accounts. See "Allow a Survey Taker to Add New Accounts" on page 134.

Allow smart filtering of accounts
Smart filtering affects the drop-down list that is presented to a survey taker who is adding accounts to a survey. When you allow smart filtering, the drop-down list contains only those accounts that are in the top-level dimension.



When you disallow smart filtering (deselect the checkbox), the drop-down list contains accounts that are outside of the top-level dimension.

Maximum accounts for new assignment
specifies the maximum number of assignments to additional accounts that a survey taker can add to a survey.

Follow-up driver surveys
causes the automatic creation and assigning of surveys for Activity Drivers. Whenever you create a Resource Driver survey an Activity Driver survey is automatically created and assigned to the same survey takers as the Resource Driver survey.

Allow a Survey Taker to Add New Accounts

A survey taker who is taking a driver survey can create assignments that don't already exist in the model. The survey taker can then provide a value for the Driver Quantity Fixed (DQF) for each new assignment.

Note: A survey taker can add new assignments provided that an administrator has selected the **Allow Survey Taker to Add New Accounts** model option for the model. See "Model Preferences" on page 133.

Each new assignment must be audited by a survey administrator. After an administrator has approved the addition, the survey values for the assignment are saved to a staging table. See "Audit Surveys" on page 124.

Each new assignment that a survey taker creates, along with its value for DQF, is saved in the Assignment staging table. See "Assignment table" on page 226.

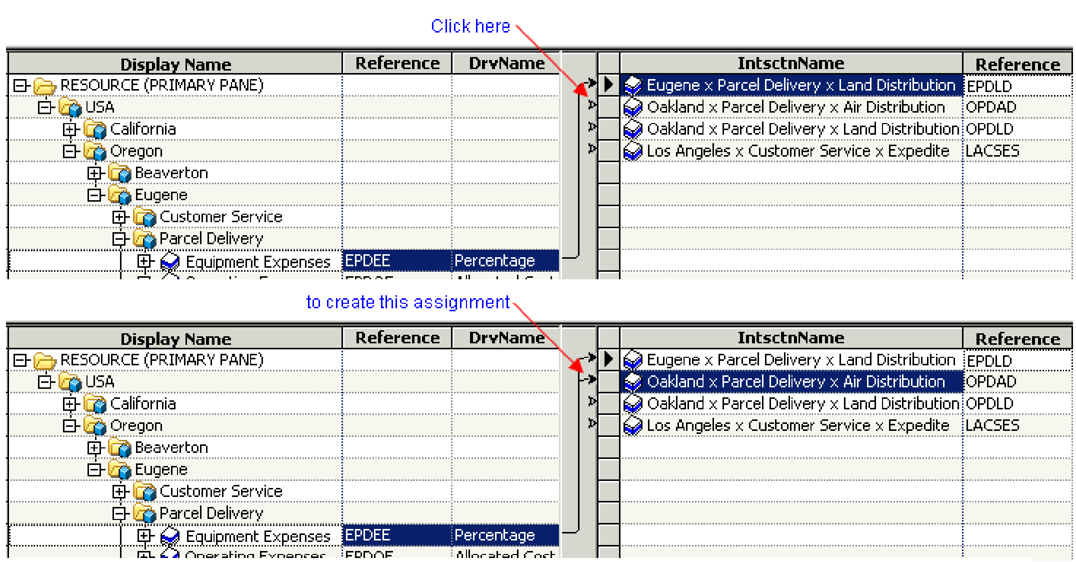
Note: A survey taker can add accounts only in the next module. For a Resource Driver survey, the survey taker can add accounts in the Activity module. For an Activity Driver survey, the survey taker can add accounts in the Cost Object module.

Detailed Example

As previously discussed, to create an assignment using the SAS Activity-Based Management user interface, you

- add a driver to the source account
- add potential destination accounts to the right assignments pane
- click on the destination account

The following picture shows how to create an assignment between Equipment Expenses (EPDEE) and Land Distribution (OPDAD).



As an example, the next picture shows how a survey taker can create an assignment from the EPDEE source account to the OPDAD destination account while taking a Resource Driver survey, as follows:

1. Double-click the source account **Equipment Expenses (EPDEE)** to open it.

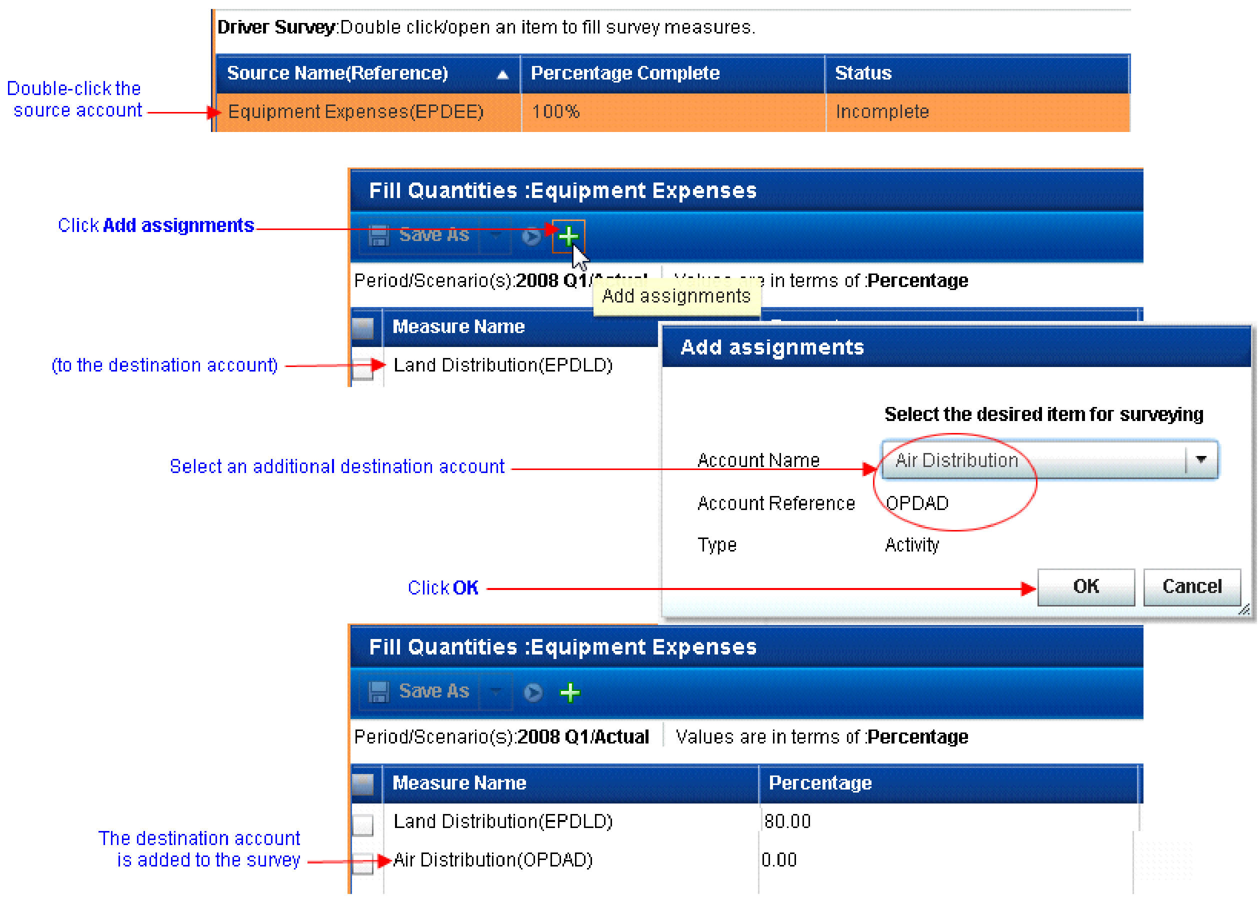
   The existing destination account **Land Distribution (EPDLD)** opens for surveying.

2. Click the **Add assignments** button (+).

   The Add assignments window appears.

3. Select a new (additional) destination account from the drop-down list next to **Account Name**, and then click **OK**.

   Note: The items in the drop-down list are controlled by the **Allow smart filtering of accounts** Model preference. See “Model Preferences” on page 133.



Follow-up Driver Surveys

If you choose this option, then whenever you create a Resource Driver survey the following occurs.

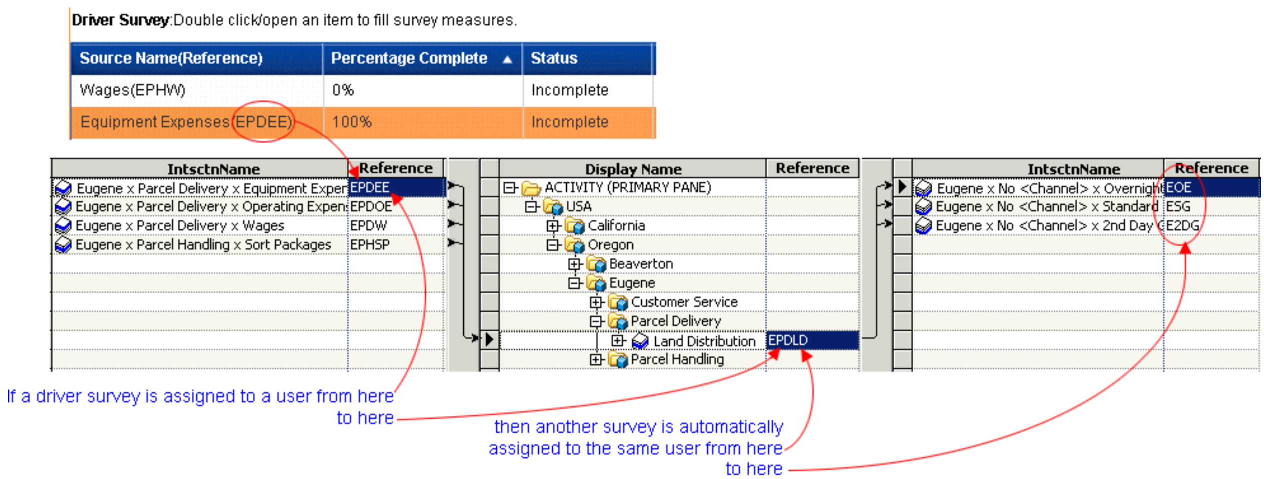
If both of the following conditions hold:

- You assign to a survey taker the survey from a source account in the Resource module to a destination account in the Activity module
- There is an assignment in the Activity model from that destination account to destination accounts in the Cost Object module

then the following occurs:

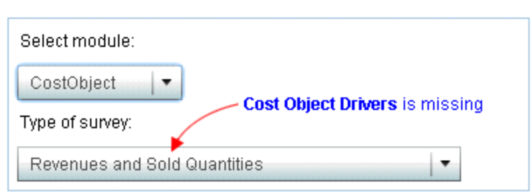
- When the survey is submitted, another survey is automatically created from the Activity module account to the Cost Object accounts
- The new survey is automatically assigned to the same survey taker

This might be clearer in the context of an example. In the following picture, you can see that there is an assignment from EPDEE to EPDLD. And, there is an assignment from EPDLD to three other accounts (EOE, ESG,and E2DG). So, if you create a survey from EPDEE to EWPDLD and assign it to John, then another survey is automatically created from EPDLD to the three other accounts (EOE, ESG, and E2DG). This automatically created survey is assigned to John.

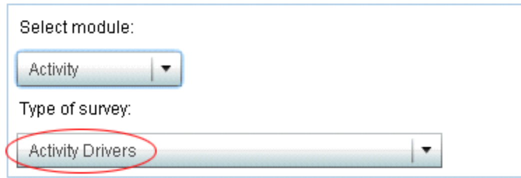


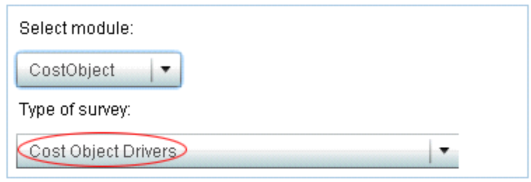
Note: Activity accounts using the Evenly Assigned driver are not included in automatic follow-up because Evenly Assigned drivers do not require user input to determine the allocation of costs.

If you select **Follow-up driver surveys** for a particular model, then when you create either Activity surveys or Cost Object surveys, the option to create driver surveys is not presented to you—because, in that case, those surveys are created automatically.



If you deselect **Follow-up driver surveys** for a particular model, then when you create either Activity surveys or Cost Object surveys, the option to create driver surveys is presented to you—because, in that case, those surveys are not created automatically.


Select module:
Activity
Type of survey:
Activity Drivers


Select module:
CostObject
Type of survey:
Cost Object Drivers

Chapter 11
Taking a Survey

Take a Driver Survey

General Procedure

The following is a list of driver surveys:

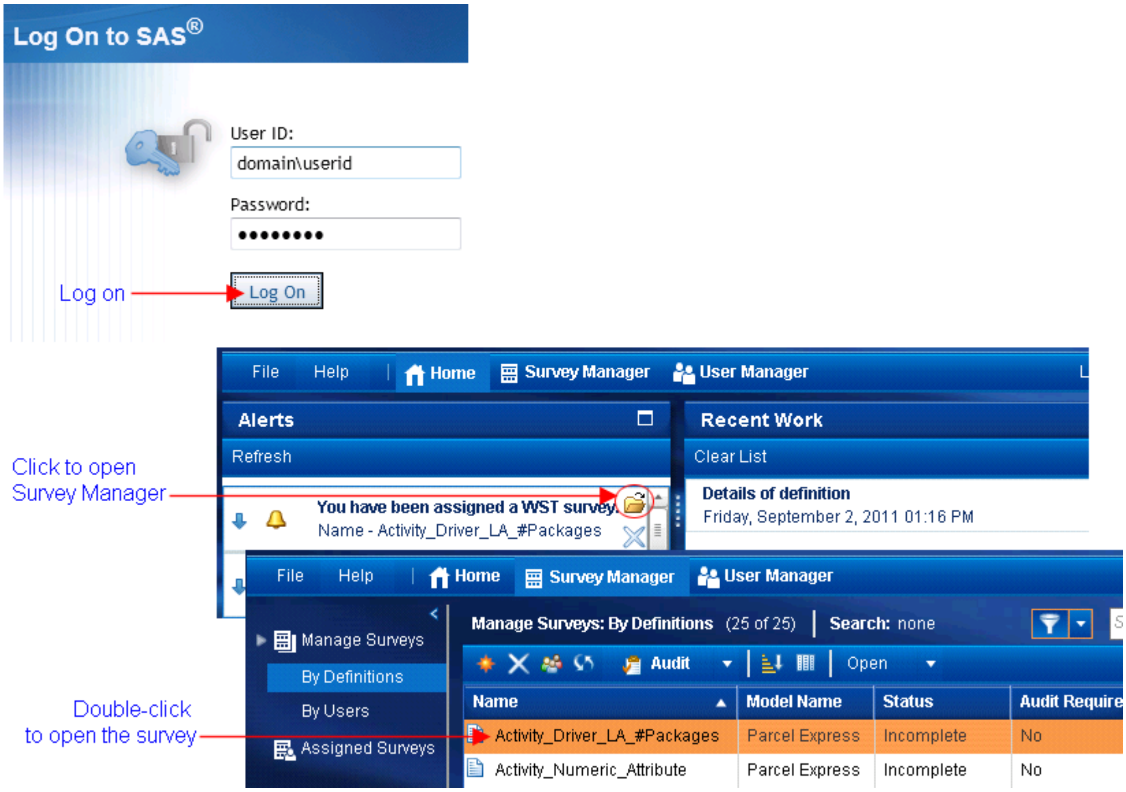
- "Resource Module: Resource Drivers" on page 101
- "Activity Module: Activity Drivers" on page 104
- "Cost Object Module: Cost Object Drivers" on page 107

To take a survey, a user must have the Take Surveys capability. For a discussion of roles and capabilities, see "Creating Roles" on page 4. Also see "Survey Takers" on page 25.

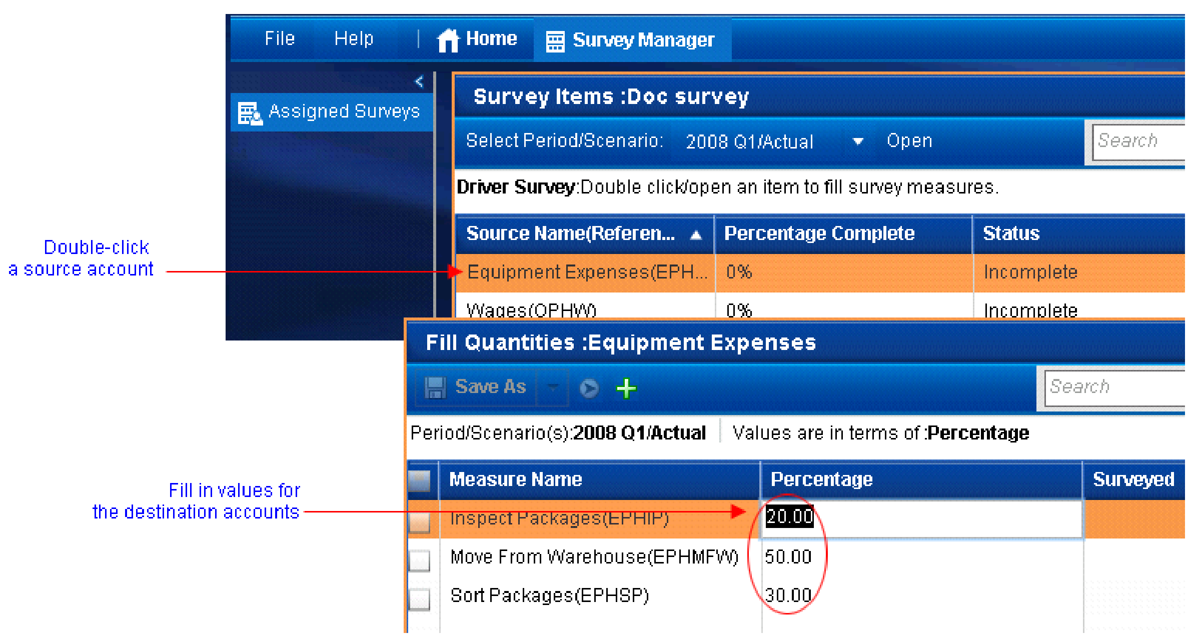
When an administrator assigns a survey, the survey taker receives a notification email. The email contains a link to the survey. After clicking on the link, the survey taker

1. Logs on to SAS Activity-Based Management Surveys.
2. In the Home page, double-clicks the survey notification to open the Survey Manager. The survey to take is highlighted.
3. Double-clicks the highlighted survey to open and take it.

   Note: You can open a survey directly from the Survey Manager.



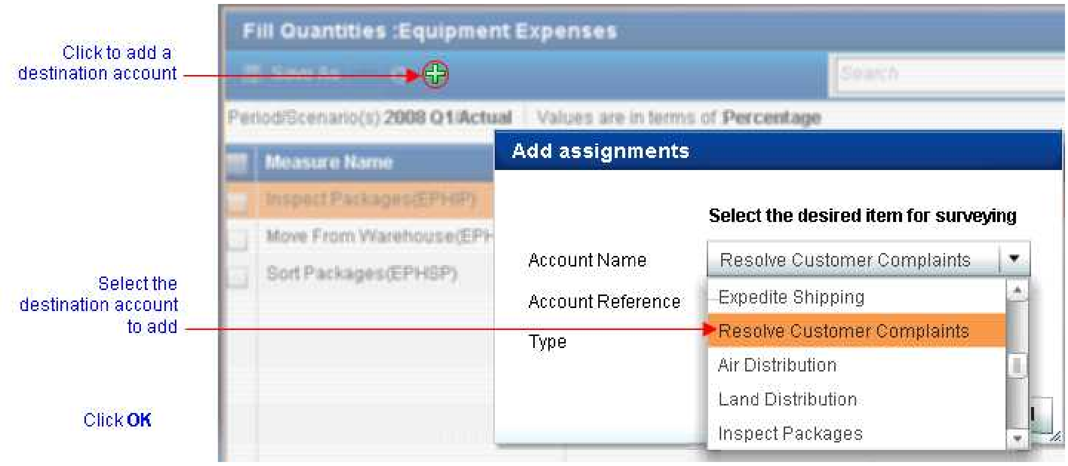
4. Double-click a source account.
5. Fill in values for the destination account.



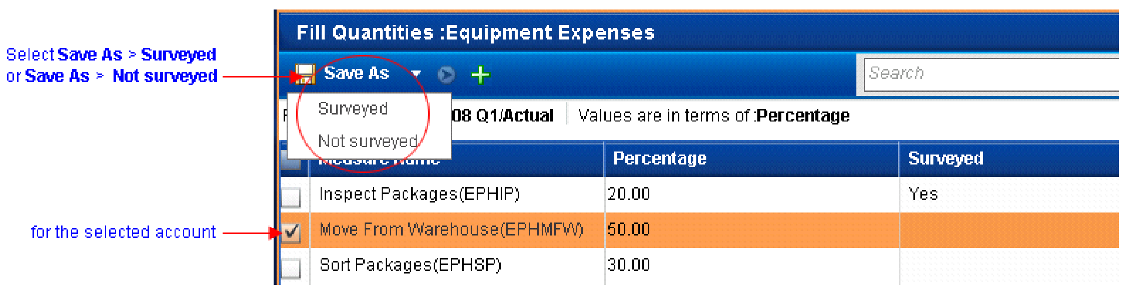
6. If adding new destination accounts is allowed and you want to add a new destination account, then click the plus sign.

   See “Allow a Survey Taker to Add New Accounts” on page 134.

7. To add a new destination account, select the desired account, and then click **OK**.



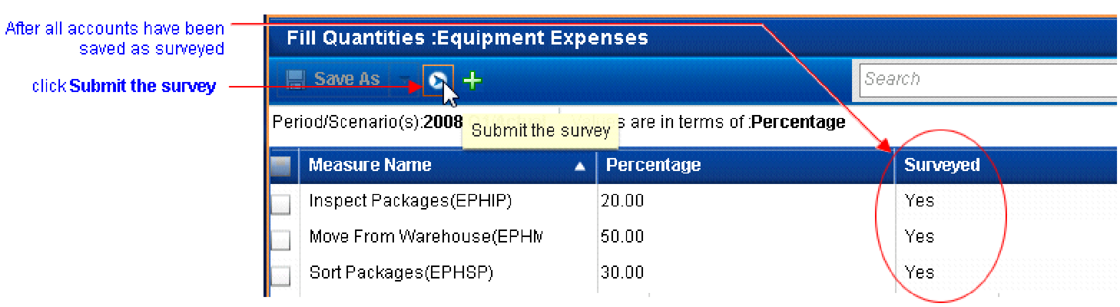
8. For each account, select **Save As** ⇨ **Surveyed** or **Save As** ⇨ **Not surveyed**.



The Survey Manager displays the Percentage Complete—the percentage of survey items that have been saved as surveyed.

Note: When a user selects **Save As** ⇨ **Surveyed**, then values are written to staging tables in the database. When a user selects **Save As** ⇨ **Not surveyed**, then nothing happens to staging tables—in particular, the database is not rolled back.

Once all items have been saved as surveyed, the survey taker can submit the survey.



See "Submit a Survey" on page 145.

Add New Assignments

A survey taker who is taking a driver survey can create assignments that don't already exist in the model. The survey taker can then provide a value for the Driver Quantity Fixed (DQF) for each new assignment.

Note: A survey taker can add new assignments provided that an administrator has selected the **Allow Survey Taker to Add New Accounts** model option for the model. See "Model Preferences" on page 133.

Each new assignment must be audited by a survey administrator. After an administrator has approved the addition, the survey values for the assignment are saved to a staging table. See "Audit Surveys" on page 124.

Each new assignment that a survey taker creates, along with its value for DQF, is saved in the Assignment staging table. See "Assignment table" on page 226.

Note: A survey taker can add accounts only in the next module. For a Resource Driver survey, the survey taker can add accounts in the Activity module. For an Activity Driver survey, the survey taker can add accounts in the Cost Object module.

As an example, the next picture shows how a survey taker can create an assignment from the EPDEE source account to the OPDAD destination account while taking a Resource Driver survey, as follows:

1. Double-click the source account **Equipment Expenses (EPDEE)** to open it.

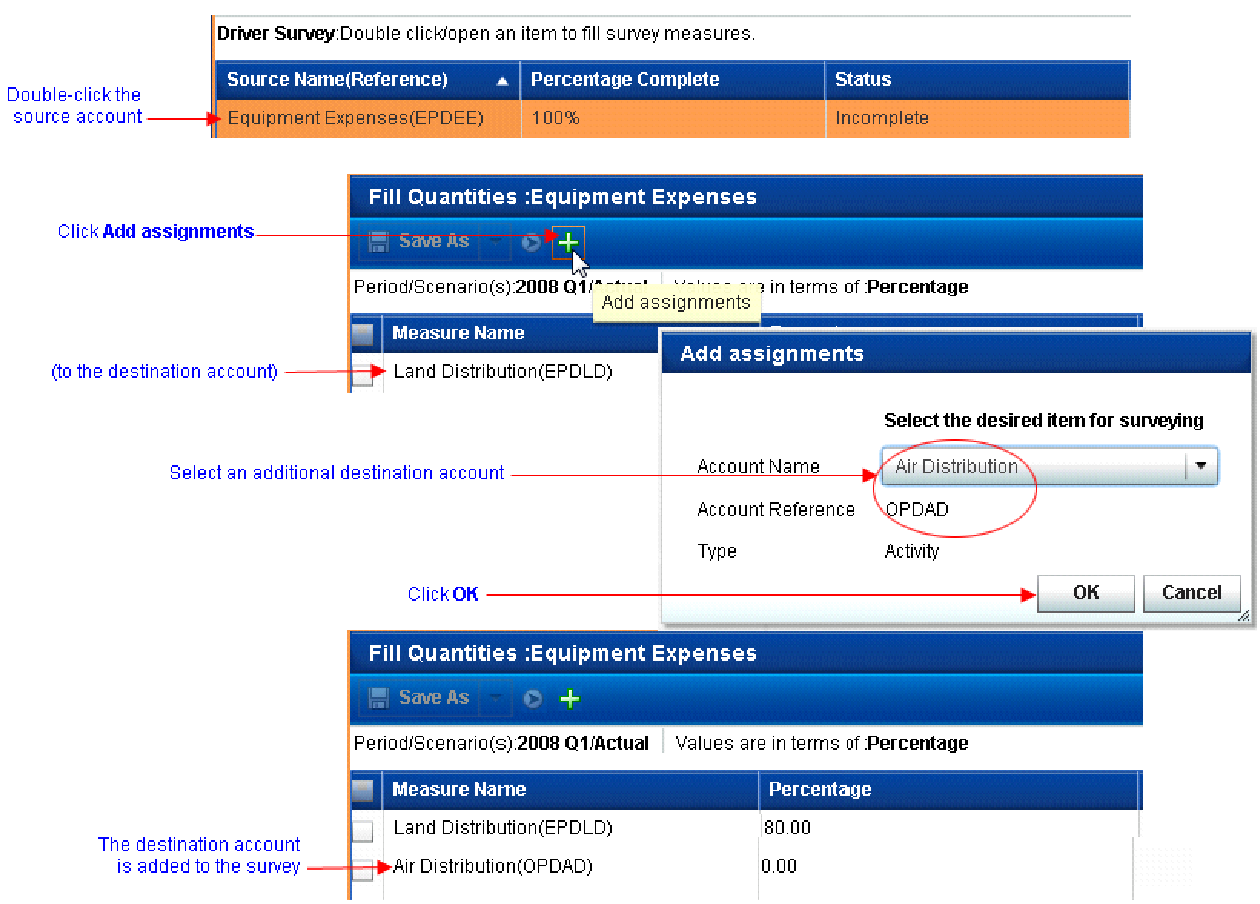
   The existing destination account **Land Distribution (EPDLD)** opens for surveying.

2. Click the **Add assignments** button (+).

   The Add assignments window appears.

3. Select a new (additional) destination account from the drop-down list next to **Account Name**, and then click **OK**.

   Note: The items in the drop-down list are controlled by the **Allow smart filtering of accounts** Model preference. See "Model Preferences" on page 133.



Take a Non-Driver Survey

The following is a list of non-driver surveys:

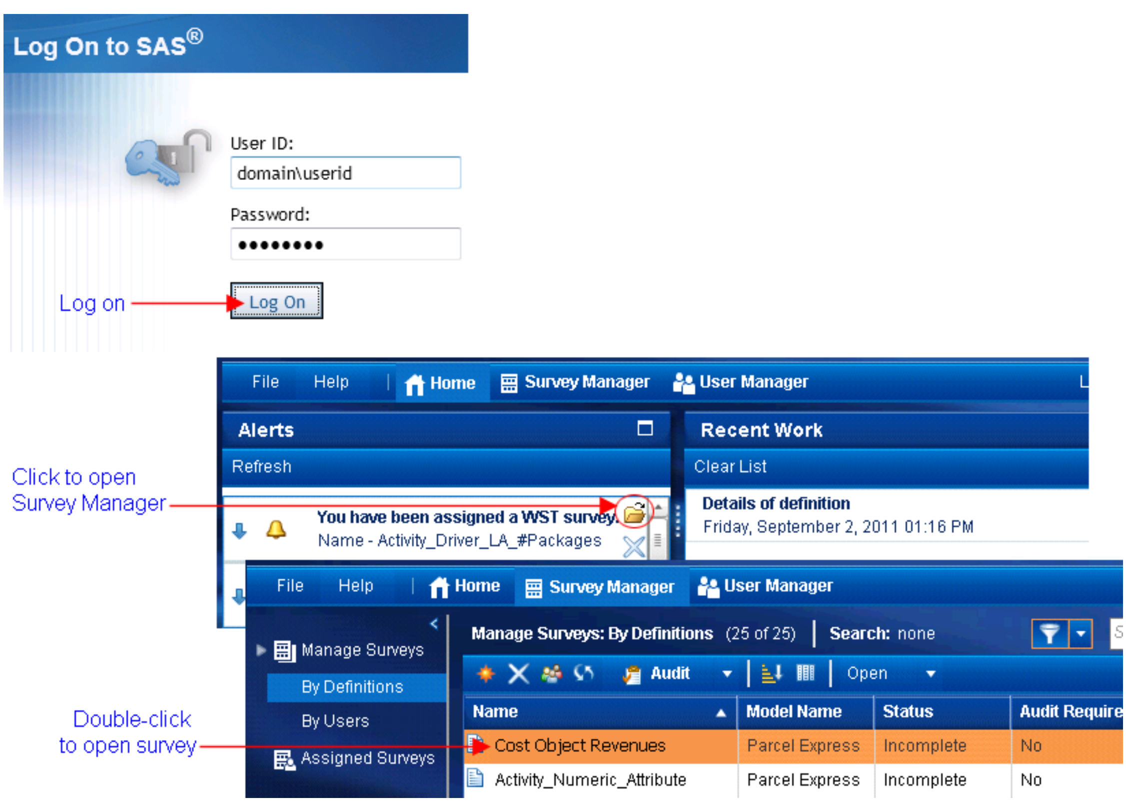
- "External Unit Module: Quantity (account)" on page 99
- "External Unit Module: Unit Costs" on page 100
- "Resource Module: Resource Costs" on page 102
- "Resource Module: Numeric Attributes" on page 103
- "Activity Module: Numeric Attributes" on page 106
- "Cost Object Module: Revenues and Sold Quantities" on page 108
- "Cost Object Module: Output Quantities" on page 109
- "Cost Object Module: Numeric Attributes" on page 110

To take a survey, a user must have the Take Surveys capability. For a discussion of roles and capabilities, see "Creating Roles" on page 4. Also see "Survey Takers" on page 25.

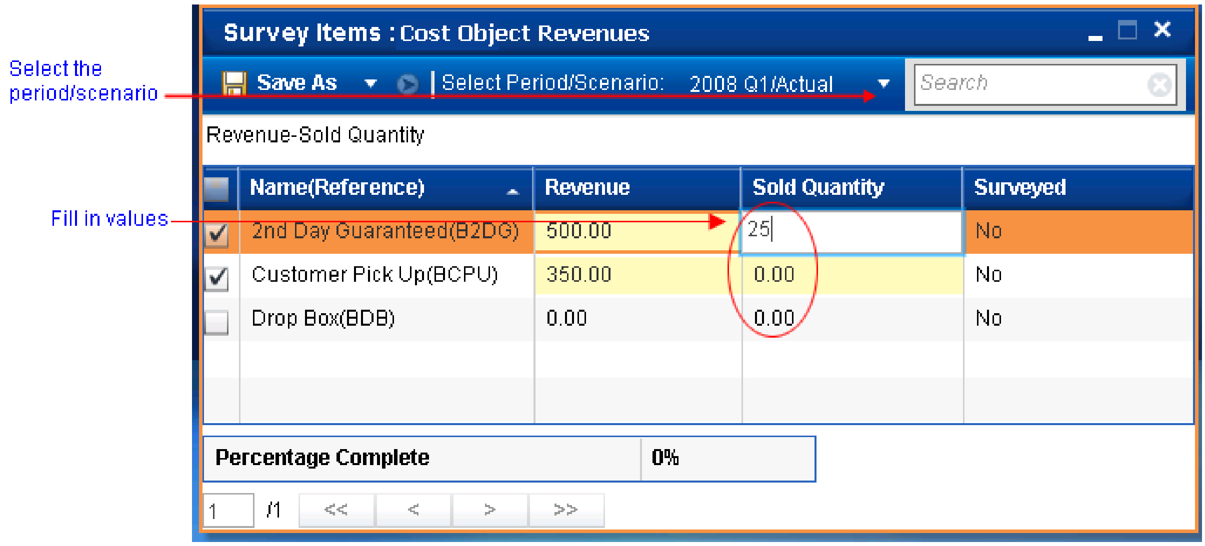
When an administrator assigns a survey, the survey taker receives a notification e-mail. The e-mail contains a link to the survey. After clicking on the link, the survey taker

1. Logs on to SAS Activity-Based Management Surveys.
2. In the Home page, double-clicks the survey notification to open the Survey Manager. The survey to take is highlighted.
3. Double-clicks the highlighted survey to open and take it.

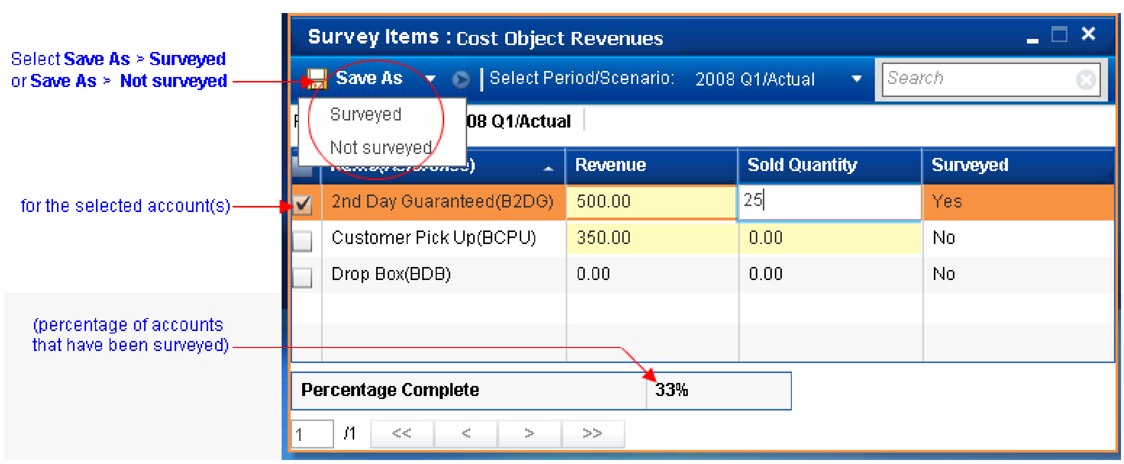
   Note: You can open a survey directly from the Survey Manager.



4. Select the period/scenario for the survey.
5. Fill in values for the survey.



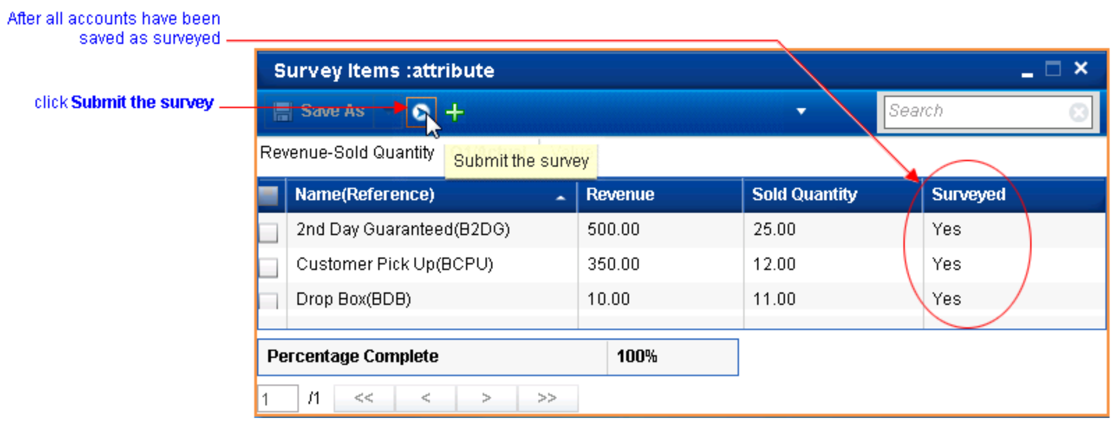
6. For each account, select **Save As** ⇨ **Surveyed** or **Save As** ⇨ **Not surveyed**.



The Survey Manager displays the Percentage Complete—the percentage of survey items that have been saved as surveyed.

Note: When a user selects **Save As** ⇨ **Surveyed**, then values are written to staging tables in the database. When a user selects **Save As** ⇨ **Not surveyed**, then nothing happens to staging tables—in particular, the database is not rolled back.

Once all items have been saved as surveyed, the survey taker can submit the survey.

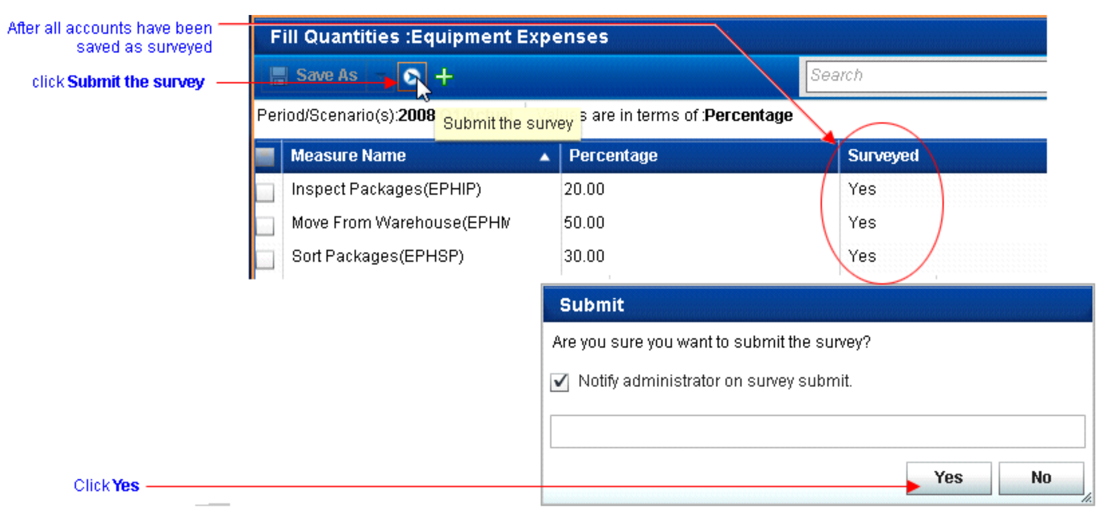


See "Submit a Survey" on page 145.

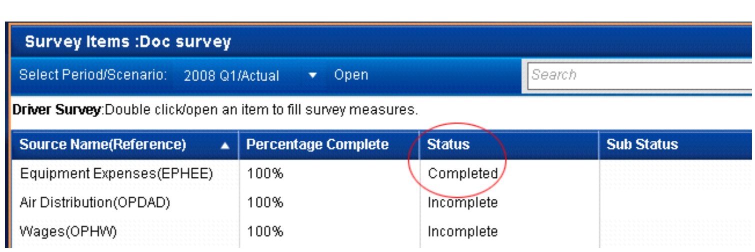
Submit a Survey

The following description applies to any survey. After a survey taker has saved all accounts as surveyed, the survey taker can submit the survey. To submit a survey:

1. Click **Submit the survey**.
    a. Select **Notify administrator on survey submit** to send an alert to the survey administrator.
    b. Click **Yes** to submit the survey.



2. The survey status changes to **Completed**.



Note: Once a survey has been completed, the survey taker can no longer change any responses unless the survey administrator changes the survey back to Incomplete. See "Mark a Survey as Incomplete" on page 123.

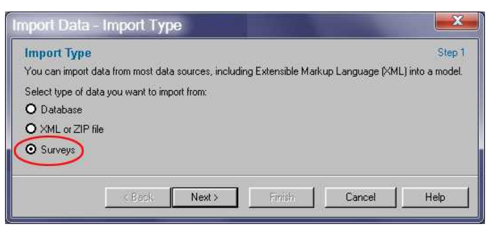
Chapter 12

Importing Survey Data back into the Model

Importing Survey Data

To import survey data, do the following.

1. Select **File** ⇨ **Import** ⇨ **Model Data**.
2. Select **Surveys**, and then click **Next**.



3. On the Import Data – Model window, select an existing model to update with survey data. You cannot create a new model from survey data.

   a. Select an existing model to update with survey data.

      Note: You cannot create a new model from survey data.

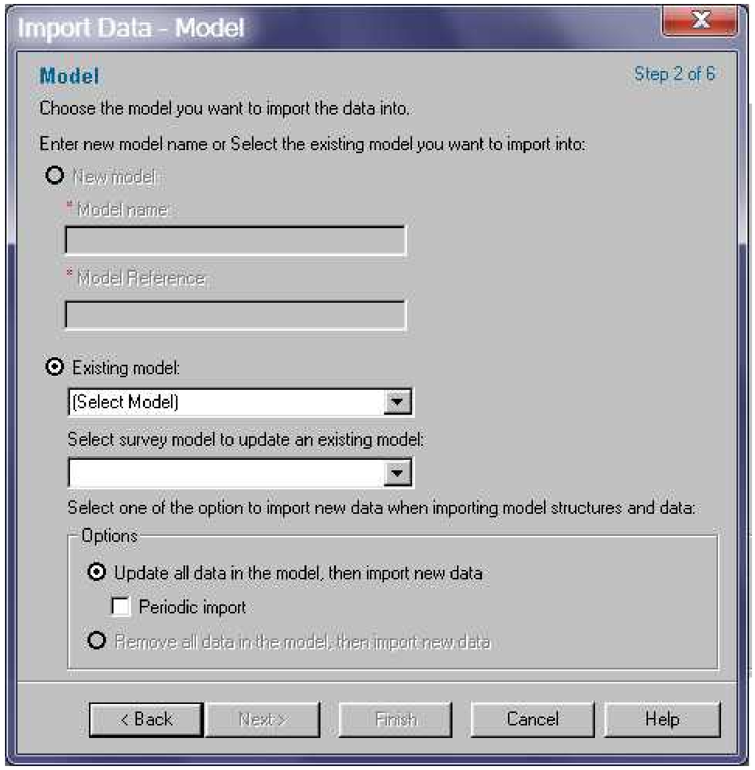
   b. Select the survey data to import.

      Note: Make sure that the survey data is for the correct model. If the data is from a different model than the one from which data was exported, the import can corrupt the existing model. See Step 1 on page 87.

   c. Select whether you want to do periodic import.

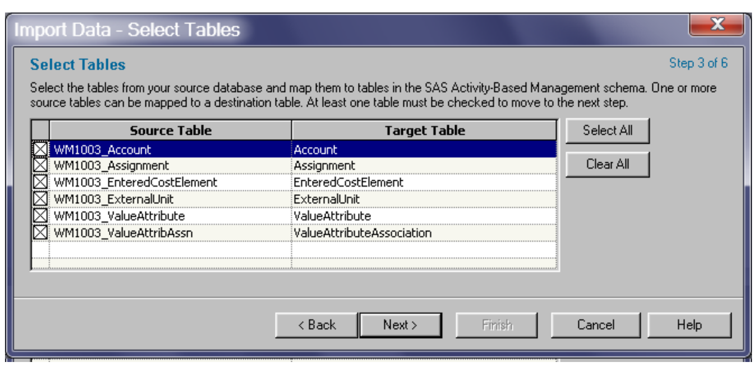
      Periodic import allows you to import only the periods that have changed in a model. For information, see the section on “Incremental Cube Generation” in the SAS Activity-Based Management User’s Guide.

   d. Click **Next**.



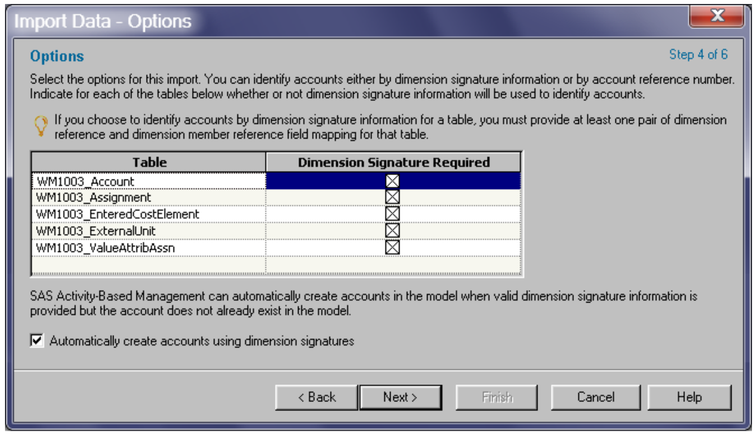
4. On the Import Data – Select Tables window, select tables from the survey data being imported and map them to tables in the model being updated.

   The tables being imported correlate to the tables that were previously exported. See Step 5 on page 88.



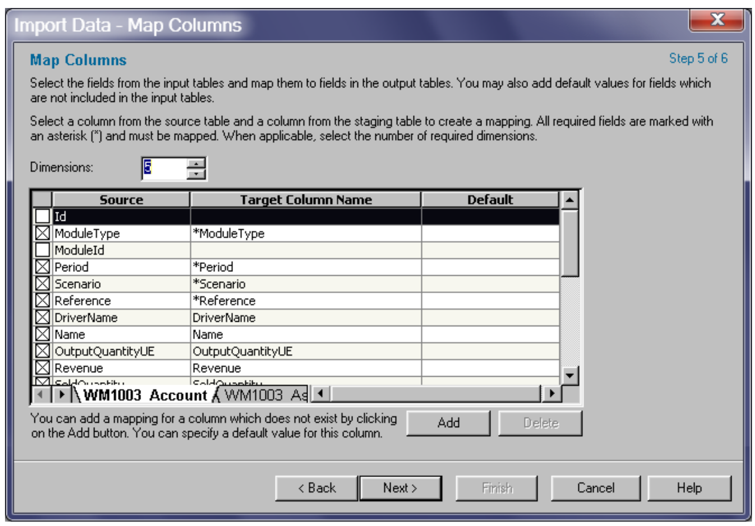
5. On the Import Data – Options window, for each table being imported, select whether you want to identify accounts by their dimension signature.

   Note: The import will work regardless of what you choose, so you can simply click **Next**.

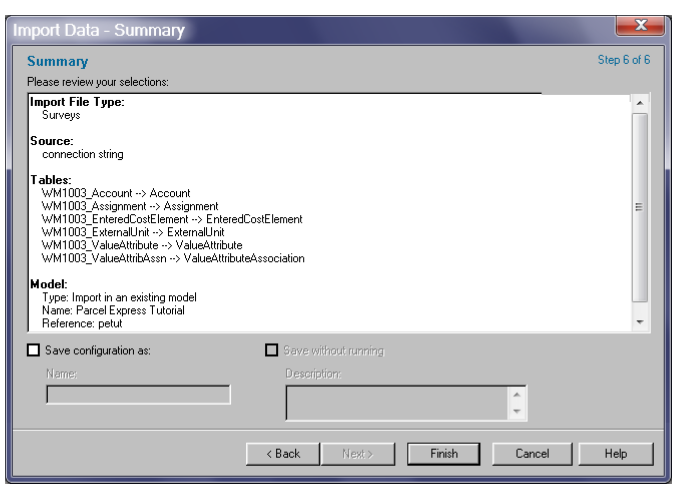


6. On the Import Data – Map Columns window, select the columns to be imported..

   The columns being imported correlate to the columns that were previously exported. See Step 6 on page 88. And see Step 7 on page 89.



7. Review your selections and click **Finish**.



Part 4

Using Staging Tables to Import and Export

Chapter 13

Importing

General Information on Importing a Model from a Database

General steps

The following general steps describe how to import data:

1. Verify that you have permission.
2. Import the data. You can choose to import the entire data set at once, or you can import the data in groups and populate the model in the following general steps:

   Import 1
   The first import step defines the existence of the model and includes the Dimension, DimensionOrder, Period, and Scenario tables. This step is equivalent to the finishing point of the New Model Wizard when you interactively build a model.

   Import 2
   The second import step defines the models content (resources, activities, and cost objects) and includes the Account, DimensionLevel, and DimensionMember tables. This step is equivalent to the finishing point of the New Account Wizard when you interactively create accounts. The Account table includes the revenue and sold quantities.

   Import 3
   The third import step loads costs into the model and includes the EnteredCostElement and ExternalUnit tables. This step is equivalent to the point where you create cost elements in the New Account Wizard when you interactively build a model.

Import 4
The fourth import step flows costs through the model using assignments with quantities for flow calculation. This step includes the Assignment and Driver tables and is equivalent to the point where you interactively create assignments and specify driver quantities in the New Account Wizard.

Import 5
The fifth import step creates attributes for analysis and numerical attributes for performance measures (cost per unit). This step includes the DimensionAttributeAssociation, ValueAttribute, and ValueAttributeAssociation tables. This step is equivalent to creating attributes and attaching attributes to accounts when you interactively build a model.

Import 6
(optional): The sixth import step includes the AssignmentNonUnique, CurrencyRate, PeriodLevel, and ScenarioLevel tables.

3. Calculate costs.
4. Generate cubes.

For a detailed summary about which tables to group in each step and in what order to import tables, see the following Import process summary.

Import process summary

The following table summarizes the process for importing your data in steps:

Table name	Requires dimension signature?	Requires reference numbers?	Has multiple keys?	Import step order
Dimension			No	Import 1
DimensionOrder			No	Import 1
Period			No	Import 1
Scenario			No	Import 1
Account*	Yes	Yes	Yes	Import 2
DimensionLevel			No	Import 2
DimensionMember			No	Import 2
EnteredCostElement		Yes	Yes	Import 3
ExternalUnit	Yes	Yes	Yes	Import 3
Assignment		Yes	Yes	Import 4
Driver			No	Import 4

Table name	Requires dimension signature?	Requires reference numbers?	Has multiple keys?	Import step order
DimensionAttributeAssociation	Yes	Yes	Yes	Import 5
ValueAttribute			No	Import 5
ValueAttributeAssociation	Yes	Yes	Yes	Import 5
AssignmentNonUnique		Yes	Yes	Optional
CurrencyRate			No	Optional
PeriodLevel			No	Optional
ScenarioLevel			No	Optional

Note: The Account table must contain both a dimension signature and a reference number, while the other tables listed with "Yes" in both columns must have either a dimension signature or a reference number.

Importing data from a database

Overview

To import from a database, you can use the wizard, or you can write a program.

For information on connecting to a database, see Chapter 15, “Connecting to a Database,” on page 189 .

Make sure to observe the restrictions on table names in a database for all staging tables.

The data schema

The database to be imported must match the data schema. For information on the data schema, see Chapter 16, “Staging-Table Schemas,” on page 205.

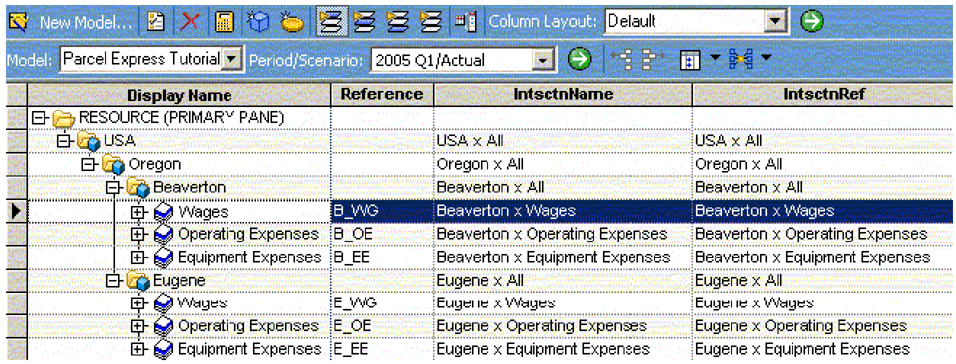
All tables and fields in the data schema (except tables that are designated for export only) must exist in the database. When the database is imported, SAS Activity-Based Management attempts to convert all values to a reasonable format type. For example, if imported dates are in the Microsoft SQL Server varchar format, the dates are converted to binary dates that are compatible with the SAS Activity-Based Management database. SAS Activity-Based Management attempts to convert all numeric values.

Understanding keys

The SAS Activity-Based Management model is based on a dual-key concept. To define any account in the model, you can describe it based on its dimension signature key or its reference key.

Dimension signature key

The dimension signature consists of a dimension reference and a dimension member reference for each dimension used to define an account. In the following example, the Resource module has been defined based on two dimensions (Region and General Ledger Account). The individual accounts are defined as intersections of these two dimensions. So, the Wages account (highlighted) consists of an intersection of the Region=Beaverton and the General Ledger Account=Wages. This account definition can be displayed in the grid with the intersection name (the column IntsctnName) or the intersection reference (the column IntsctnRef) properties.



The dimension signature for this account is based on the dimensions used and the dimension members used, so the dimension signature for this account requires multiple values as shown here:

Property	Example
DimRef1	Region
DimMemberRef1	Beaverton
DimRef2	General Ledger Account
DimMemberRef2	Wages

Reference key

In the previous example, the reference for the account is B_WG, and this single value uniquely identifies the account. (Guidelines for using dimension signatures and references)

Creating sample database tables

To understand how to create database tables, you should interactively import a sample model. The sample model demonstrates the dimensions and dimension order for each module, the default period/scenario association, and anticipated periods and scenarios. Use the sample model as a source for the model export to a database for general use.

Create the sample database tables by performing the following general steps:

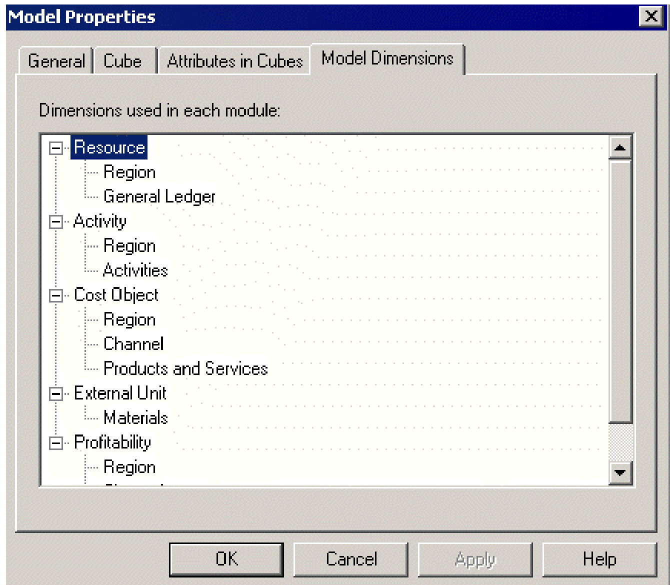
1. Import a sample model in XML format. The sample models are located in the following directory:

   ```
   C:\Program Files\<Client_Installation_Location>
   \SAS\Activity-Based Management Solution\Client\Samples\Models\Native
   ```

2. Create a target database to export your sample database.
3. Archive the model.

   This creates a set of database tables.

4. Review the contents of the model.



If the model that you want to import has the same number of dimensions in each module as the sample model shown, you can use the sample model directly. The External Units module consists of one dimension; the Resource module consists of two dimensions; the Activity module consists of two dimensions; and the Cost Object module consists of three dimensions. After you export the sample database, modify the contents of the tables to reflect your own data in the appropriate dimension signature columns.

If the model that you want to import has a different number of dimensions in each module as the sample model, then the sample can still provide a good starting point for creating a staging table template. But, you must customize all of the tables that require the dimension signature (the Account, Assignment, EnteredCostElement, ExternalUnit, DimensionAttributeAssociation, and ValueAttributeAssocation tables) to include all of the dimensions that you used in the model that you want to import.

Number of dimensions in the imported model	**Difference in the number of dimensions in the sample model**
External Units module: two dimensions	Add one dimension
Resource module: one dimension	Remove one dimension
Activity module: three dimensions	Add one dimension
Cost Object module: five dimensions	Add two dimensions

Exporting to a database creates all of the required tables with most of the required columns. The required columns in some of the tables depend on the number of dimensions in a model. Because each dimension signature consists of a pair of columns, any table with dimension signature columns will add columns as the number of dimensions in the model increases. Although you can manually create the tables, exporting to a database reduces the risk of omitting required tables, omitting required columns within tables, and creating typographical errors. And, it gives you a viable sample to follow in developing your own staging table content and when building a SAS Activity-Based Management model.

Database table relationships, dependencies, and data values

There are no database-enforced relationships between any of the SAS Activity-Based Management tables, and there are no dependencies between tables. You can specify any values in the tables that you need. However, while the data is being imported, SAS Activity-Based Management checks the validity of the values and rejects any invalid records. You will receive messages that indicate any errors.

If you import all the tables at one time, the wizard will import the tables in the correct order, so errors are minimized. However, if you import individual tables, then you must ensure that the tables are imported in the correct order, as noted in the Importing data: General steps Help section.

Following are a few examples of common errors made while importing tables:

- A record in the DimensionMember table does not correspond to a record in the Dimension table. You must import the Dimension table before you import the DimensionMember table.
- A record in the Account table does not correspond to a record in the DimensionMember table. You must import the DimensionMember table before you import the Account table.
- A record in the Assignment table (where the source account or the destination account is) is not in the Account table. You must import the Account table before you import the Assignment table. If you select the option to create a new Account through the Assignments dimension signature, you do not need to import the Account table first.

Guidelines for using dimension signatures and references

When importing data for assignments, cost elements, dimension attribute associations, and value attribute associations, you can choose to import with either a dimension signature or reference key. As you define the extraction and transformation processes to create the staging tables for SAS Activity-Based Management, you should be aware of the advantages and disadvantages of using one key over another.

If you are importing model data using references rather than dimension signatures, you must define both the reference and the dimension signature in the Account table. The Account table can be used as a mapping index to match the imported data in staging tables (Assignment, EnteredCostElement, DimensionAttributeAssociation, and the ValueAttributeAssociation tables) to their respective dimension signature as defined in the model. The advantage of using references rather than dimension signatures is that the import tables can have significantly fewer columns. For example, in a two dimension model, the reference would be a single column to import and the dimension signature would be four columns to import. The total impact of this change to the Assignment table would be six fewer columns required.

If you are importing model data using dimension signatures, you must include the dimension reference and dimension member reference for each dimension, which can be tedious to maintain. For example, in a two dimension model, the required keys include four columns for the two dimensions. However, there is one advantage of importing data using dimension signatures: You can automatically create new accounts that appear in the transaction tables (Assignment, EnteredCostElement, DimensionAttributeAssociation, and the ValueAttributeAssociation tables). The dimension signature method provides a distinct advantage over the reference key method because the reference key method skips over any new accounts found in the transaction tables and fails to import them.

Period/scenario associations

If you use the wizard to import data, you can import multiple period/scenario associations. If you write a program to import data, you can import more than one period/scenario association at a time. Period/scenario associations are imported if a model element (such as an account or a cost element) contains data for the period/scenario association.

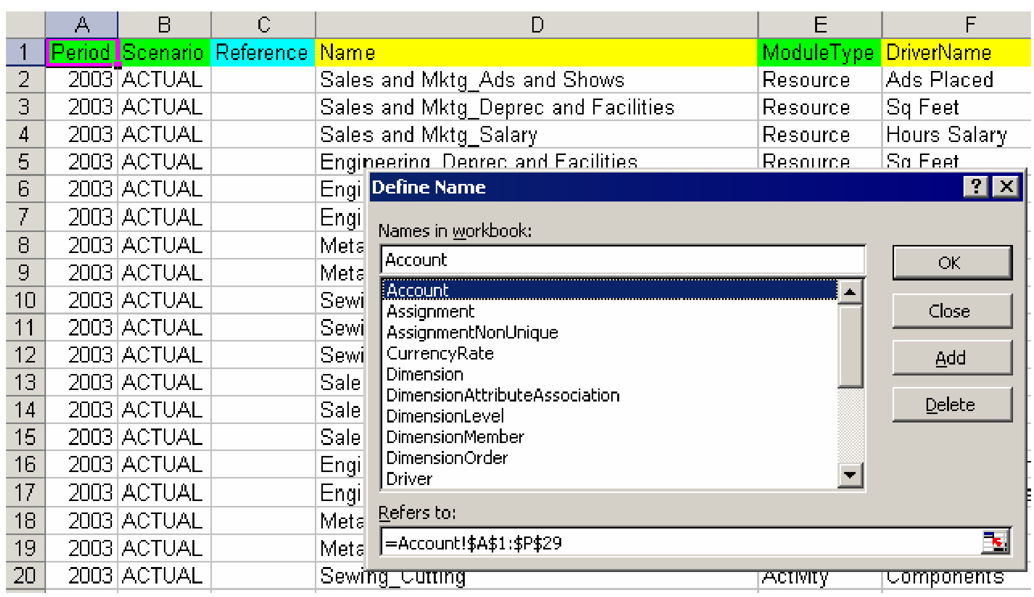
Data from multiple databases

You can import data from multiple databases by first creating a database view. The view specifies which data you want to import from multiple tables in multiple databases. Then, you can use the wizard.

Alternatively, you can use the SAS Activity-Based Management Web Services Integration API to write a program that uses an XML import configuration to specify each database in a separate StagingArea element.

Data from a Microsoft Excel workbook

If you want to import data from a Microsoft Excel workbook, you must create named ranges for the different sets of data. To create a named range in Microsoft Excel, highlight the data and select Insert > Name > Define. Multiple named ranges must be defined for each required stage table inside a single Microsoft Excel spreadsheet, as shown in the following example:



Period/scenario associations

Period/scenario associations are ignored unless there is a model element (such as an account or a cost element) that contains data for the period/scenario association.

Importing data with your own program

To write a program for importing data, use the SAS Activity-Based Management Web Services Integration API. You must also create XML that maps the structures and the periodic data in the database or in the XML file to the SAS Activity-Based Management model database.

Using the Import Data Wizard to Import from a Database

Importing the Model

Before attempting to import data from a database, see Chapter 15, “Connecting to a Database,” on page 189, which provides information about connecting to the most common databases.

Note: You can import a model without first opening a model.

1. Create a database or a database view that matches the data schema.

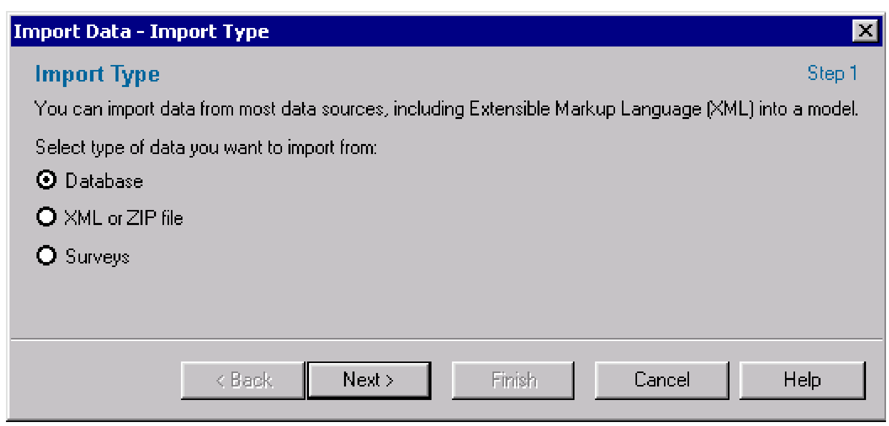
   For information about the data schema, see . on page 205

2. In SAS Activity-Based Management, verify or create any required period/scenario associations.
3. Select **File** ⇨ **Import** ⇨ **Model Data**.

   The Import Data Wizard opens.

Step 1 of the Wizard

1. Select the **Database** option.

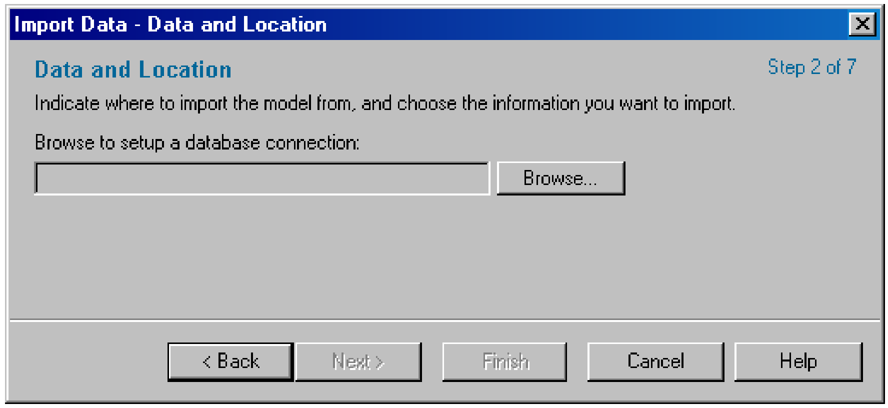


2. Click **Next**.

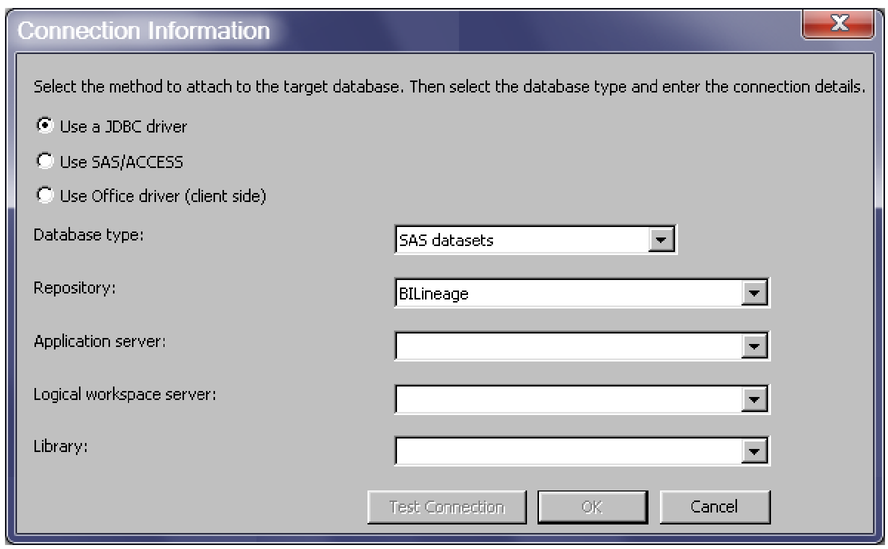
Step 2 of the Wizard

1. Click the 

button on the Wizard page.



The Connection Information window opens.

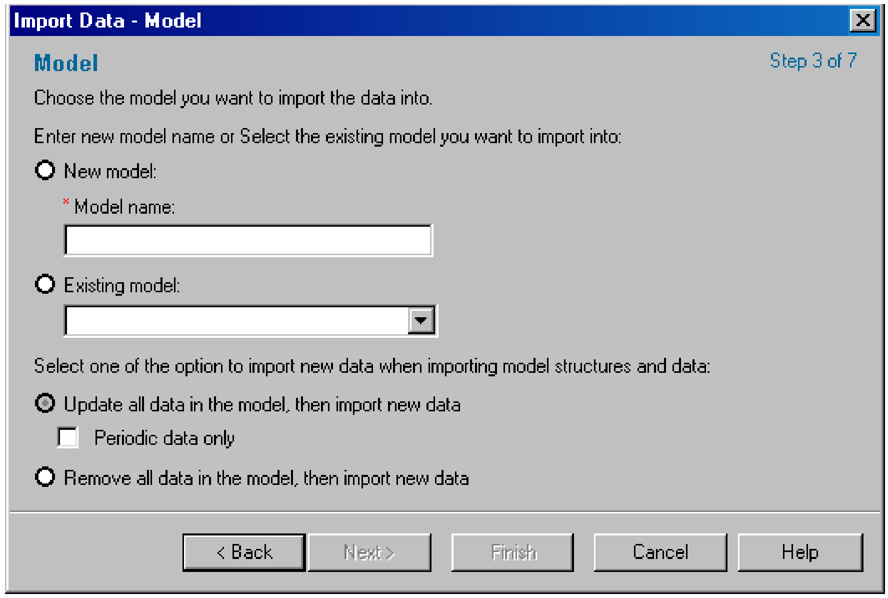


For information on using this window, see Chapter 15, “Connecting to a Database,” on page 189.

2. Click **Next**.

Step 3 of the Wizard

1. If you want to import the database into a new model, do the following:



   a. Select the **New model** option.

   b. Type the **Model name**.

2. If you want to import the database into an existing model, do the following:

   a. Select the **Existing model** option.

   b. From the drop-down list, select a model.

   c. To append the imported data to the existing model data, select the **Update all data in the model, then import new data** option.

   d. Select **Periodic data only** to import only the staging tables that contain periodic data. See "Periodic Import" on page 167.

   e. To remove all existing data, select the **Remove all data in the model, then import new data** option.

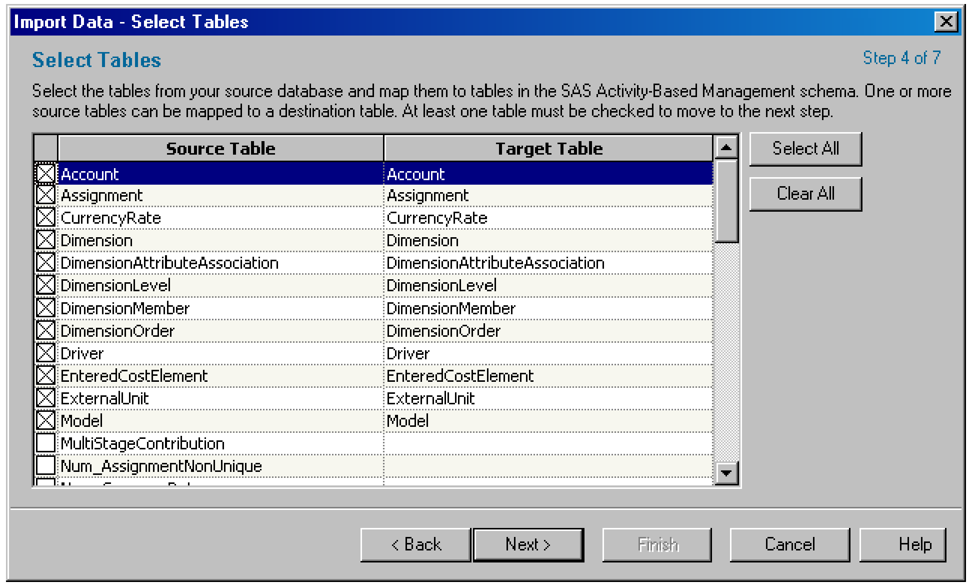
3. Click **Next**.

Step 4 of the Wizard

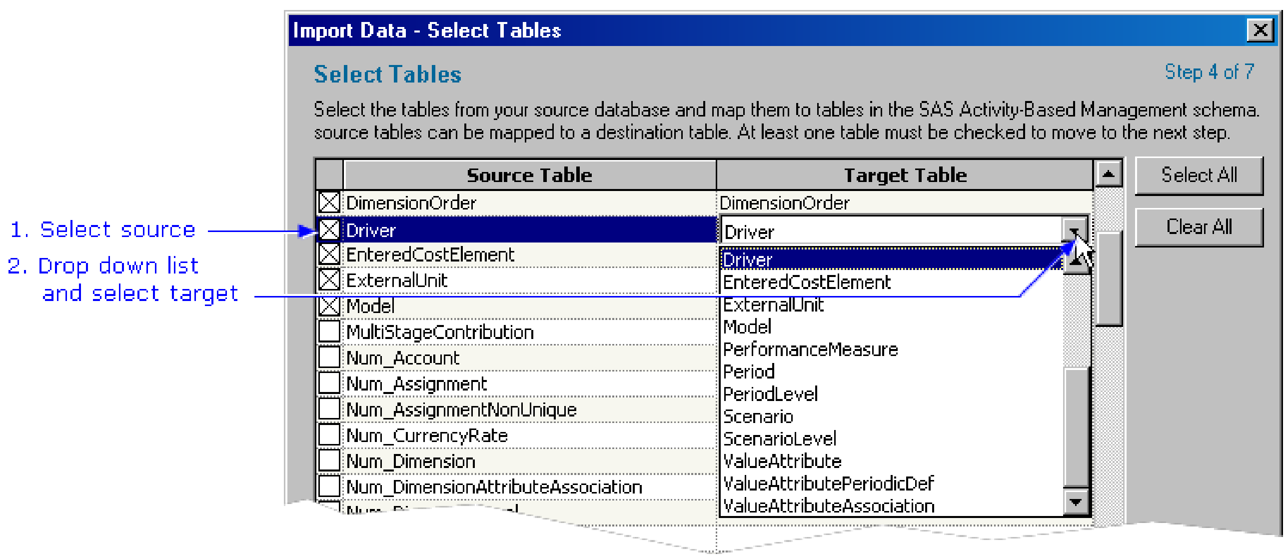
In this step, you specify which tables to import. And, for each table that you import, you specify to what SAS Activity-Based Management table it maps.

1. To select a table to import, select the check box to the left of the table name in the **Source Table** column.

   Note: If you selected **Periodic data only** in the previous step, then only tables that contain periodic information are displayed for selection.



2. To map a **Source Table**, click in the **Target Table** column to the right of the source table, and select a SAS Activity-Based Management table from the drop-down list.



Note: If you have named the source tables with the same names that are used in SAS Activity-Based Management, the wizard automatically creates the mappings and selects the tables to be imported. If you have not used the same names, then you must specify which source table maps to which SAS Activity-Based Management table.

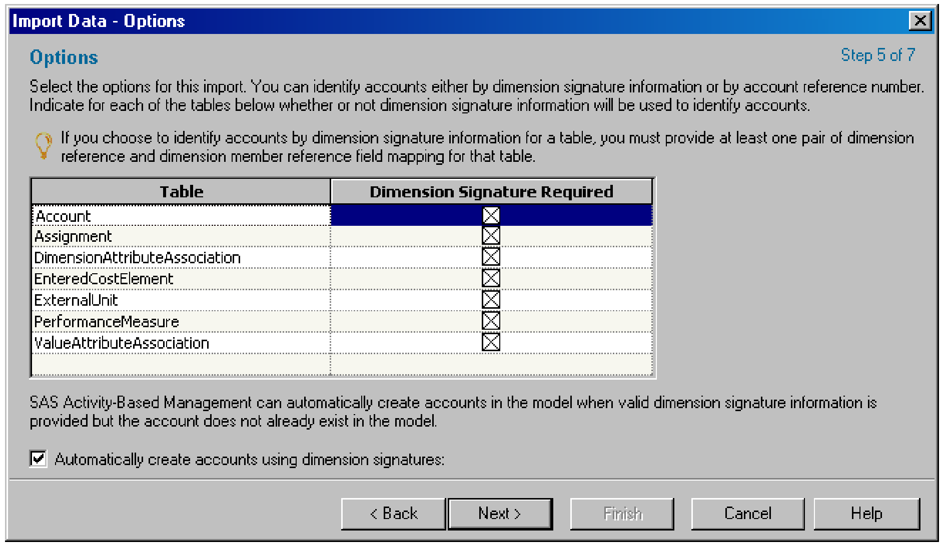
3. Repeat steps 1 and 2 to map every table that you want to import.

4. Click **Next**.

Step 5 of the Wizard

In this step, you specify how accounts are identified in source tables that contain accounts. You can identify each account by a dimension signature or by reference number. You can choose to have the wizard create accounts for all valid dimension signatures, even if accounts have not been created for those intersections.

1. To identify accounts by the dimension signature, select the check box in the **Dimension Signature Required** column to the right of each source **Table**.



   If you clear the check box, accounts will be identified by their references.

2. To have the wizard automatically create accounts for all valid dimension signatures, select the **Automatically create accounts using dimension signatures** option.

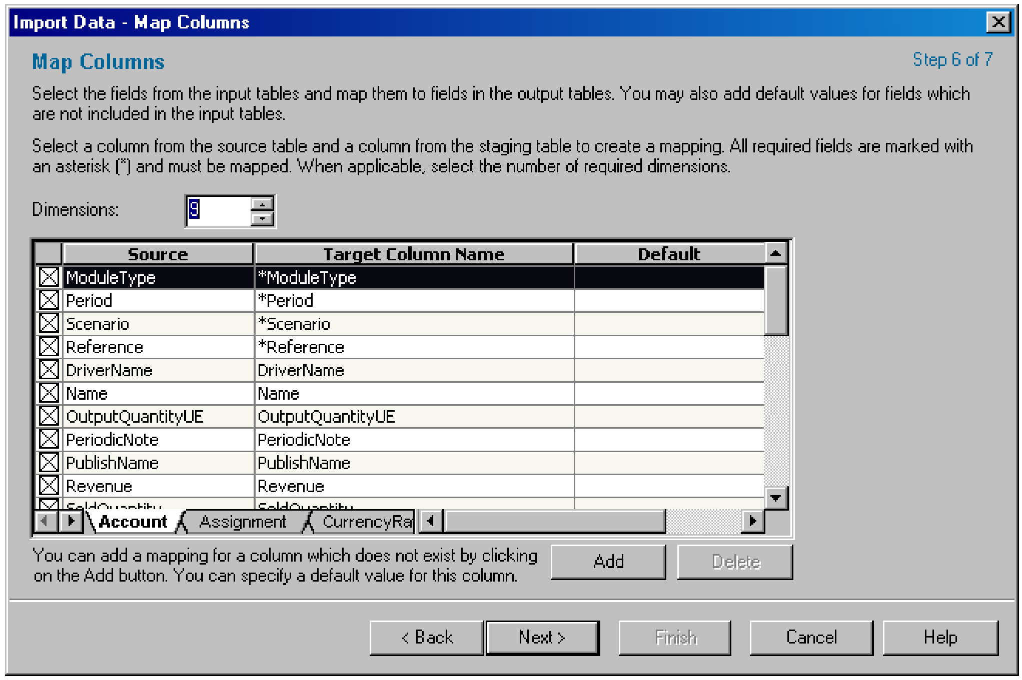
3. Click **Next**.

Step 6 of the Wizard

In this step, you specify which table columns to import. And, for each table column that you import, you specify to which table column in SAS Activity-Based Management it maps.

1. To select a table column to import, select the check box to the left of the table name in the **Source** column.

   You can select as many table columns as needed, but you must import those columns that are required. Required columns are denoted by an asterisk (*).



Note: If you have named the source table columns with the same column names that are used in SAS Activity-Based Management, the wizard automatically creates the mappings and selects the columns to be imported. If you have not used the same names, then you must specify which source table column maps to which SAS Activity-Based Management table column.

2. If you chose in the previous step to identify accounts by their dimension signatures, specify the number of **Dimensions** that are contained in the source table.

   The **Dimensions** option appears only when you choose to identify accounts by their dimension signatures. You must specify the number of dimensions so that the interface displays the correct column names that you must map. For each dimension in the source table, there must be two columns that can be mapped to the SAS Activity-Based Management table columns; these two columns are named DimRef<number> and DimMemberRef<number>. For example, a source table that contains two dimensions must contain columns that must be mapped to DimRef1, DimMemberRef1, DimRef2, and DimMemberRef2.

3. To map a **Source** table column, click in the **Target Column Name** column to the right of the **Source** table column, and select a SAS Activity-Based Management table column from the drop-down list.

   The drop-down list contains the values DimRef<number> and DimMemberRef<number> only if you chose to identify accounts by their dimension signatures, which is the only situation in which these values are needed.

4. If a source table does not contain a column that you want to create in the model, and you want to assign a value to this column, do the following:

   a. Click **Add**.

      A new row appears.

   b. Click in the **Source** column, and select a column name from the drop-down list.

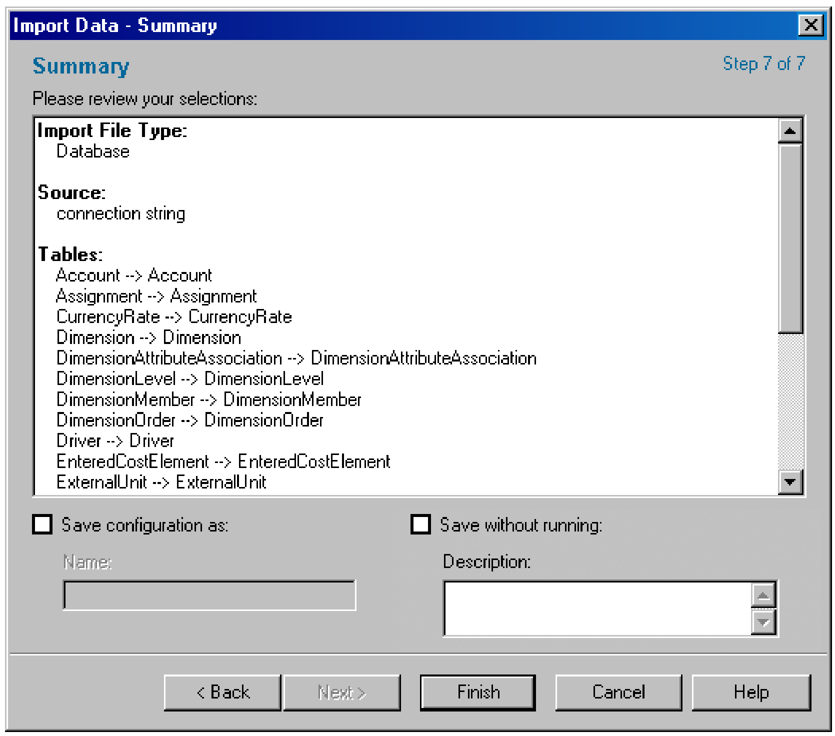
   c. Type a **Default** value.

      For example, if the source table does not contain a column to map to the Period column in SAS Activity-Based Management, and you want to create a period named MyPeriod in the model, then add a column named <None>. Map this new column to the **Target Column Name** Period, and specify the **Default** value as MyPeriod.

5. Repeat steps 1 through 4 for every table column that you want to import.
6. Click the appropriate tab at the bottom of the grid to map the columns for another table.
7. Click **Next**.

Step 7 of the Wizard

1. Review the import summary.



2. If you need to change any information, click **Back** until you reach the step that you need to change in the wizard.

   All of the information that you have specified is saved. Click **Next** to advance through the wizard.
3. To save the import configuration so that the import can be easily run again, do the following:
   a. Select the **Save configuration as** option.
   b. Type the **Name**.
   c. Type the **Description**.
4. Select **Save without running** to save the import configuration with performing the import.
5. Click **Finish**.

Periodic Import

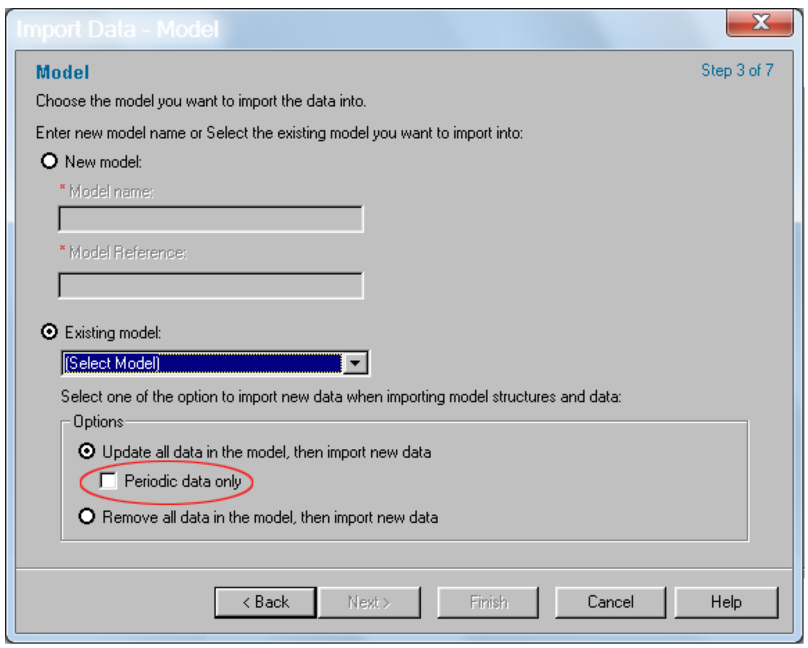
When you generate a cube, for every period (period and scenario association) that is to be included in an existing cube, if:

- the cube already contains that period, and
- the period has not been modified since the cube was last generated

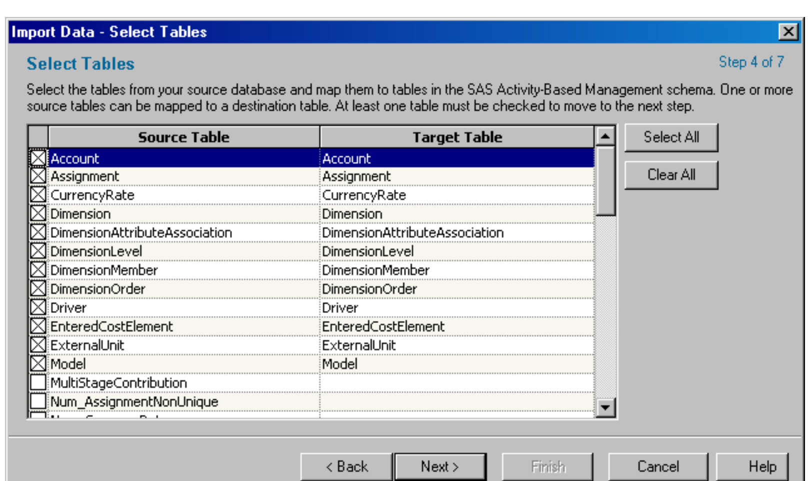
then the period is not regenerated.

This means that cube generation is faster because periods that have already been generated are not regenerated.

In order to support incremental cube generation, SAS Activity-Based Management provides a **Periodic data only** option in the Import Wizard that allows you to import only the periods that have changed in a model—for example, the new periods.



If you select **Periodic data only**, then only those staging tables that contain periodic data are displayed in the Import Wizard for you to select for importing.



Staging tables are distinguished by whether they contain periodic or structural data. Periodic data is model data which is stored separately for each period/scenario association. Structural data is model data which is independent of any period/scenario association. It is data that is common to all period/scenario associations.

The following staging tables contain **periodic data**:

- Account
- Assignment
- CurrencyRate
- ExternalUnit
- EnteredCostElement
- PerformanceMeasure
- ValueAttributeAssociation
- ValueAttributePeridicDef
- DimensionalAttributeAssociation

The following staging tables contain **structural data**:

- Dimension
- DimensionMember
- DimensionLevel
- DimensionOrder
- Driver
- Model
- ValueAttributes
- Period
- PeriodLevel
- Scenario
- Scenariolevel

When you generate a cube for a model that has been generated before, SAS Activity-Based Management determines whether the entire cube must be regenerated or whether only the new or modified periods need to be generated. You do not have to specify, when you generate a cube, whether you want incremental generation or not. SAS Activity-Based Management makes the determination for you.

Note: Cubes in SAS OLAP that are generated using either the NO_NWAY option or the NONUPDATEABLE option are not eligible for incremental generation.

Chapter 14
Exporting

Exporting Model Data to a Database

Overview

You can export model data to a database or to an XML file. If you want to export only a portion of the model data, you must export to a database. If you export to an XML file, all model data is exported. If you want to export all of the model data, you can export to a database or to an XML file.

The following table lists some of the reasons why you might export model data. For each reason, the table shows whether you would export to a database or to an XML file:

Reason for exporting	Export destination
To export model data and to manipulate the data, and then to import the data back into the model or into another model	Database
To archive a model	Database XML file (requires less disk space)
To export some of the items in the model, but to exclude other items	Database

Note: When a table is exported to a database, SAS Activity-Based Management attempts to match the destination table's column names with the source table's column names as closely as possible, so that they are compatible with SAS. A name change is necessary when a database has certain naming limitations (such as a limited number of characters per field); those name changes are noted in the Field Name column of each table.

When exporting a table to a database, observe the following restrictions on the table name:

SQL Server:

- Table name must begin with a letter.
- Table name should not start with special characters like $, @, #, %, |, !
- Special characters like %, -, |, ! are not allowed in the table name.
- Table name must be less than or equal to 128 characters.

Oracle:

- Table name must begin with a letter.
- Table name should not start with special characters like $, _, #, %,-,|, !
- Special character like @, %, - are not allowed in the table name.
- Table name must be less than or equal to 30 characters.

MySQL:

- Special characters -,%, @, |, ! are not allowed in the table name.
- Table name must be less than or equal to 64 characters.

Microsoft Access:

- Table name must be less than or equal to 64 characters.
- Special characters like $, %, #, !, - are not allowed in the table name.

Microsoft Excel:

- Table name must be less than or equal to 32 characters.
- Special characters like $, %, #, !, - are not allowed in the table name.

Using the Export Wizard to Export Data

Overview

When you use the Export wizard to export model data, you can choose to export to a database or to an XML file. If you export to a database, you can select individual database tables and properties to export. If you export to an XML file, you must export all of the model data.

Note: If the database or the XML file already exists, it is overwritten.

Note: You can perform this task without first opening a model.

1. Verify that the model and an empty database that matches the data schema (if you export to a database) are ready.
2. Select **File** ⇨ **Export** ⇨ **Model Data**.

   The Export Wizard appears.
3. Follow the directions in the wizard. For detailed instructions, see "Using the Export Wizard" on page 175.

Archiving a Model

When you archive a model, important model data is preserved so that the model can be restored to a saved state. Although not all model data is preserved, user-entered data and unique data are saved. Model data that is not saved is regenerated by SAS Activity-Based Management when the model is restored and calculated.

You might want to archive a model for the following reasons:

- to create a backup
- to save a version before making major changes
- to restore a model after upgrading a SAS Activity-Based Management server
- to transport a model between SAS Activity-Based Management servers

Determining the Tables to Export for Business Analysis

When you export model data for business analysis, you must determine which tables to export with the appropriate calculated values. You can export the following SAS Activity-Based Management standard staging tables with calculated values.

Account table

Exporting the Account table with the calculated values enables you to perform additional analysis on any account, which is useful for static, calculated values analysis, but is not useful for cost-flow analysis. The types of fields in an exported Account table include:

- Definitional

  Keys: dimension signature and reference

  Model, module type, period, scenario, driver names, model name, unit of measure, periodic note

- Entered values: OutputQuantityUE, Revenue, SoldQty, TDQUE
- Calculated values

  Cost values: AllocatedCost, AssignedCost, AssignedIdleCost, AssignedNonReciprocalCost, AssignedReciprocalCost, DrivableCost, DrivenCost, DriverRate, EnteredCost, IdleCost, ReceivedAllocatedCost, ReceivedAssignmentCost, ReceivedBocCost, ReceivedCost, ReceivedDrivenCost, ReceivedNonReciprocalCost, ReceivedReciprocalCost, UnassignedCost, UsedCost

  Driver data: AssignedIdleQuantity, DrivenQuantity, IdlePercentage, IdleQuantity, OutputQuantity, TDQ, TDQBasic, TDQCalculated, UnassignedQuantity, UsedQuantity

  Profitability analysis (uses both entered values and calculated values): Cost, Profit, Revenue, SoldQuantity, UnitCost, UnitProfit, UnitRevenue

- Attributes (specific to the model design)

  Dimensional attributes - used for grouping

  Numerical attributes - entered values

  Calculated attributes - performance measures

Assignment table

Exporting the Assignment table with the calculated values enables you to trace specific costs as they flow through a model. The Assignment table provides the cost flow and driver-quantity flows between each source account and destination account. It contains the content from the Single-stage Contributions OLAP cube fact table. The types of fields in an exported Assignment table include:

- Definitional

Keys (for both the source account and destination account): dimension signature and reference

Destination module type, driver name, model name, period, scenario, source module type

- Driver analysis: DriverQuantityBasic, DriverQuantityCalculated, DriverQuantityFixed, DriverQuantityVariable, DriverWeightFixed, DriverWeightVariable, IdleDriverQuantity, IdleDriverQuantityUE
- Cost flow: allocated cost, cost, idle cost, source cost
- Attributes (specific to the model design)

  Dimensional attributes - used for grouping

  Numerical attributes - entered values

  Calculated attributes - performance measures

Multi-stage contributions cube fact table (with calculated values)

The types of fields in an exported Multi-stage Contributionscube include:

- Definitional

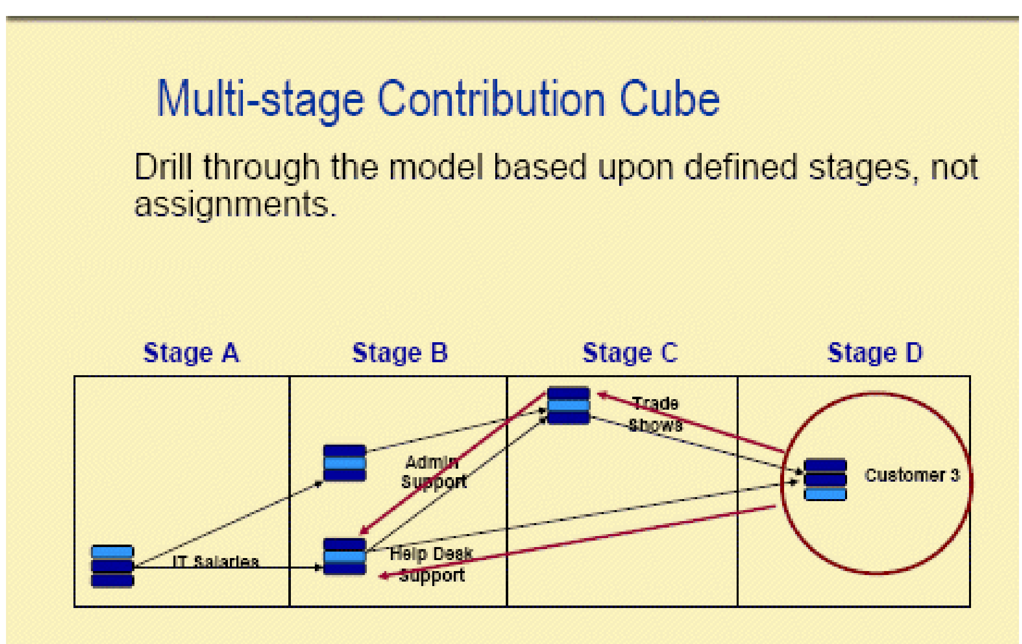
  Keys for each account (resources, activities, cost objects, external units): dimension signature and reference

  Activity module type, Cost Object module type, model name, period, Resource module type, scenario
- Entered values: OutputQuantityUE, Revenue, SoldQty, TDQUE
- Calculated values: Cost, OutputQuantity
- Attributes specific to the model design (resources, activities, cost objects, external units)

  Dimensional attributes - used for grouping

  Numerical attributes - entered values

  Calculated attributes performance measures



The cube fact table holds all of the calculations for cost flow through the model. The Multi-stage Contributions cube holds cost flow from each step through the model. Using the Multi-stage Contributions cube, an analysis can be created to trace a single, final cost object, through each contributing cost, to the activities back to the original resource.

In the previous example, all stages (A through D) are available for analysis in the Multi-stage Contributions cube. (About fact tables)

Resource Contributions cube

The types of fields in an exported Resource Contributions cube include:

- Definitional

  Keys for each beginning or final account (resources, cost objects): dimension signature and reference

  Destination module type, model name, period, scenario, source module type, source reference

- Entered values: DestinationSoldQuantity
- Calculated values: ContribCost, ContribPcnt, DestinationCost, DestinationOutputQuantity
- Attributes specific to the model design for each beginning or final account (resources, cost objects)

  Dimensional attributes used for grouping

  Numerical attributes entered values

  Calculated attributes performance measures

Report table exports

The types of reports that you can generate include:

- Reports exported from SAS Activity-Based Management: You can choose to export the standard staging tables or you can export the standard fields and contents, which are created in the standard reports that are installed with SAS Activity-Based Management.
- Reports on imported data content directly inside SAS Enterprise Guide: Using the SAS Activity-Based Management Add-In for SAS Enterprise Guide, the SAS Activity-Based Management reports can be run directly from within SAS Enterprise Guide. This produces a SAS table that can be used in further analysis.

The exported report tables are consistent with the standard report templates that are formatted and shipped with SAS Activity-Based Management.

Working with Tables, Dimensions, Properties, and Attribute Values

For tables, dimensions, properties, and attribute values, you can export all items or specific items. You can change the name of each exported item from its default name. For maximum flexibility with tables, you can export the same table to multiple export tables. For example, you can export the Account table to the tables named ResourceAccounts and ActivityAccounts.

TIP The Export wizard shows you which database fields are required for re-importing model data.

Exporting, Filtering, and Limiting Calculated Results

Following are the general steps that you perform when exporting model data:

- Export only certain period/scenario associations.

  You can export one or more period/scenario associations.

- Export only certain tables.

  You can select the specific SAS Activity-Based Management staging table you want to export. You can choose to change the names of the tables that you are exporting, which is useful when you are exporting multiple tables of the same type for distribution across a large audience (actual account and planning account).

- Choose the specific fields to be included in the dimension signature in all of the tables to be exported.

  The required fields for the dimension signature include a dimensional reference and the dimensional member reference. These fields are required if you want to re-import the exported data into SAS Activity-Based Management. However, for readability and integration with other systems (data warehouses), you might find it helpful to include the dimension name, the dimension member name, the dimension level, and the dimension level name.

- Export only certain fields in the tables.

  You can select specific fields within each table to be exported. The default (archive) selections of fields do not include any calculated values, so be sure to carefully select the calculated values you need to export for further analysis or to import into another system (data warehouse). For each field you export, you can change the names of the fields, which is particularly useful when creating a SAS Activity-Based Management system to import into another system (which might have predefined fields). You can change the numbers in a dimension signature to a more useful notation of the organizational structure.

- Export only certain dimensions in specific tables.

  You can eliminate unnecessary fields in the exported data. This is useful in the Multi-stage Contributions cube. The default behavior is to include all of the possible dimensions in each stage of the export, but in most SAS Activity-Based Management models, only a limited number of dimensions actually apply to a given stage in the model. By eliminating unnecessary fields, you can significantly decrease the size of the exported data.

- Export only certain members in a dimension in specific tables.

  You can define a specific point in the dimensional hierarchy to include in the exported data. This filter method is useful for creating specific exports for an organizational structure (specific departments, specific product lines, or specific customer types).

- Export multiple tables of a specific type -- add table and field -- filtering for content.

  When performing business analysis, you might want to export a single model into multiple tables, which is useful when providing specific results tables to specific departments. The ability to export a single SAS Activity-Based Management staging table into multiple database tables might be useful when splitting the actual costs and budget costs. To export a single table into multiple tables, you need to add a table and select multiple versions of the same staging table type, and map the versions to different destination table names. Then, apply a filter to limit the results going to each destination table.

Filtering Data

Use the following methods to filter the data that you want to export:

- select parts of a dimension
- specify comparisons for the values of attributes
- specify comparison operators for the values of fields in a table

If you filter by a table field, you do not need to export the field.

The following table lists the comparison operators:

Operator	Description	Field type
LIKE	Wildcard Use the percentage symbol (%) to specify any amount of text, including spaces. For example, Name LIKE fiscal% will match fiscal, fiscally, and fiscal year 2004.	Text
	Use an underscore (_) to specify a single character. For example, Name LIKE account_ will match account1, account2, and accountX.	Text
	If you omit both % and _, the comparison is the same as when you use the operator =. For example, Name LIKE fiscal is equivalent to Name = fiscal.	Text
=	Equal	Text or numeric
<>	Not equal	Text or numeric
<	Less than	Numeric
>	Greater than	Numeric
<=	Less than or equal to	Numeric
>=	Greater than or equal to	Numeric

Writing Your Own Program to Export Data

You can use the SAS Activity-Based Management Web Services Integration API function ExportModelData, combined with an XML export configuration, to append data to an existing database or to overwrite an existing database. When SAS Activity-Based Management appends data, it leaves existing data in the tables; new tables, new fields, and new data are added if they do not exist. For more information, see the section on "Using the API" in this document.

You can export one set of data at a time on each server. After an export finishes, another export can start.

Using the Export Wizard

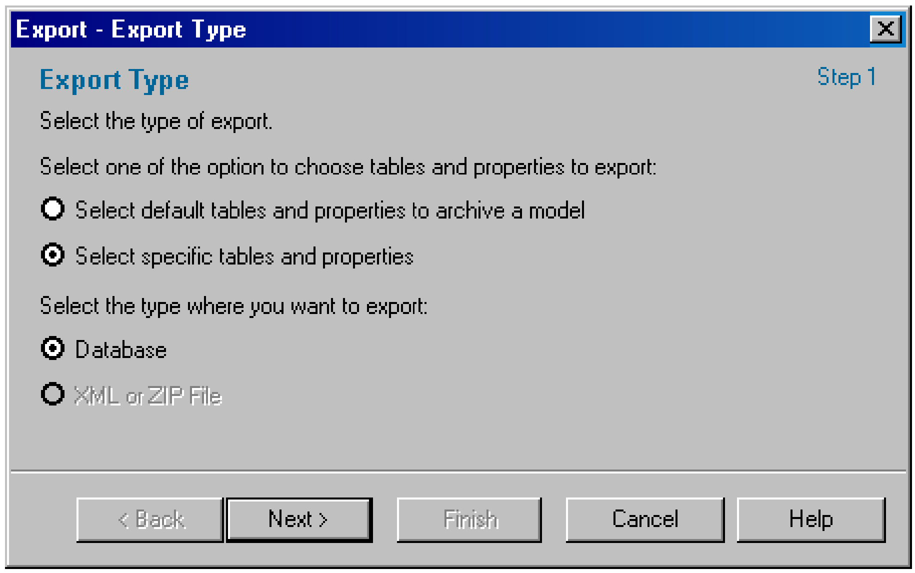
Before attempting to export data to a database, see Chapter 15, "Connecting to a Database," on page 189 which provides information about preparing to connect to the most common databases.

Note: You can perform this task without first opening a model.

1. Verify that the model is ready.
2. Select **File ⇨ Export Model Data**.

   The Export Wizard appears.

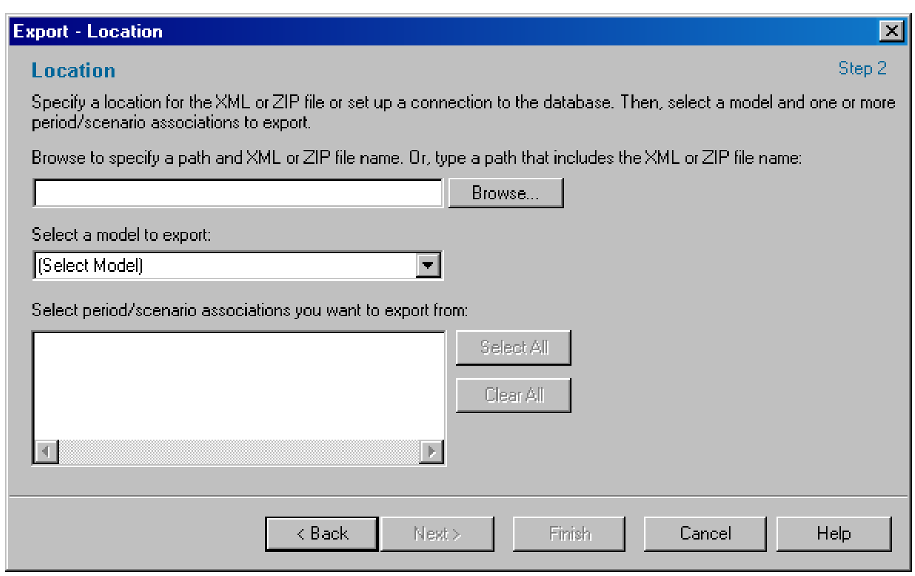
Step 1 of the Wizard



1. Select the **Select specific tables and properties** option.

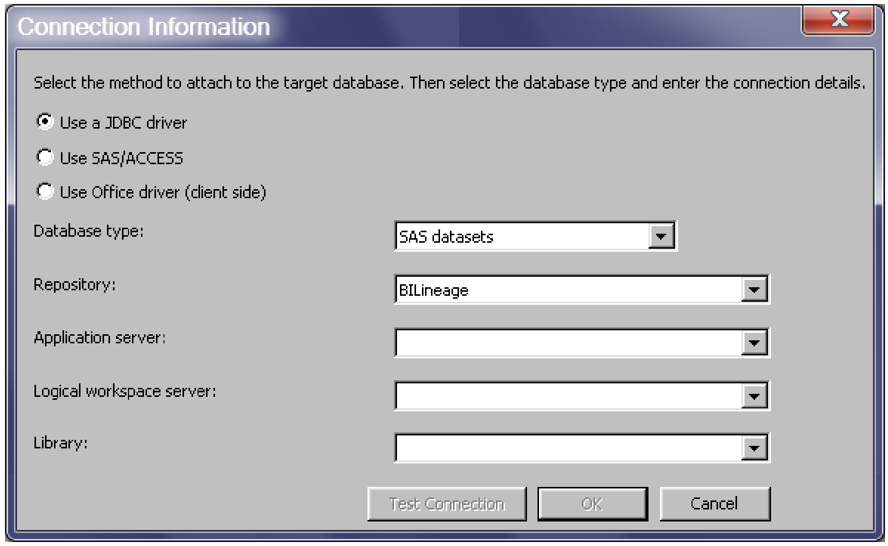
   Notice that the only export type available is **Database**.
2. Click **Next**.

Step 2 of the Wizard



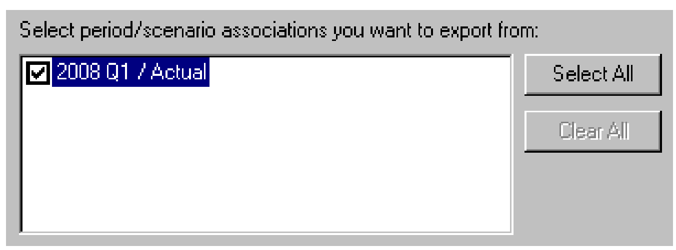
1. Click Browse... .

   The Connection Information window opens.



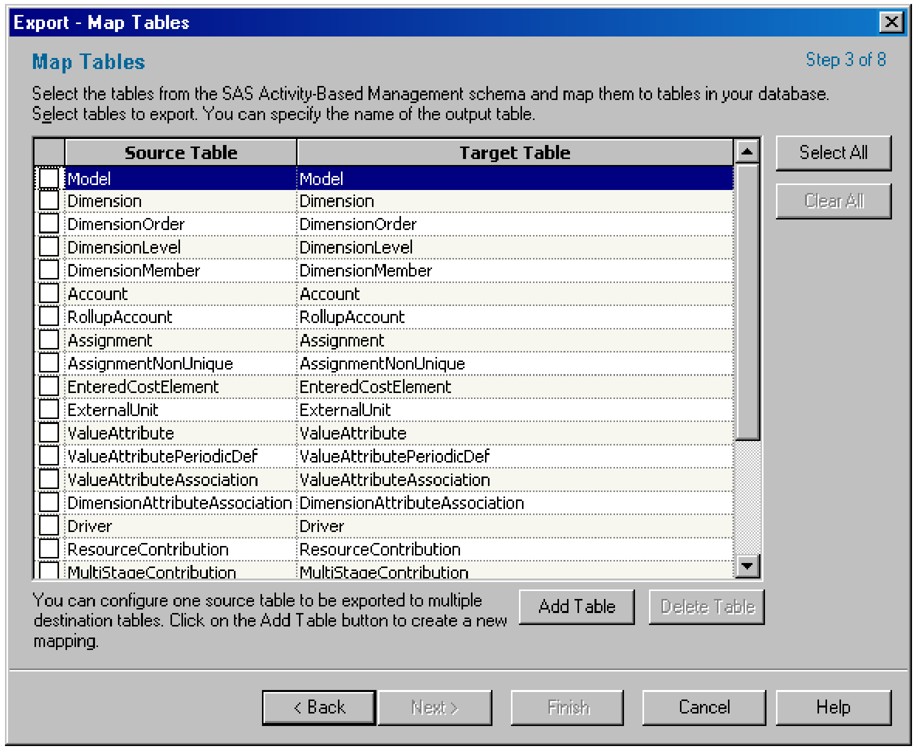
For information on using this window, see Chapter 15, “Connecting to a Database,” on page 189.

2. From the **Select a model to export** drop-down list, select a model.
3. From the **Select period/scenario associations you want to export from** list, select the check box next to one or more period/scenario associations.



4. Click **Next**.

Step 3 of the Wizard



1. To select a table to export, select the check box to the left of the table in the **Source Table** column.

   You can select as many tables as needed.

2. To change the name of an exported table, click in the **Target Table** column to the right of a source table and type a new name.

3. To map a source table to more than one target table, do the following:

   a. Click **Add Table**.

      A new row is added with default information.

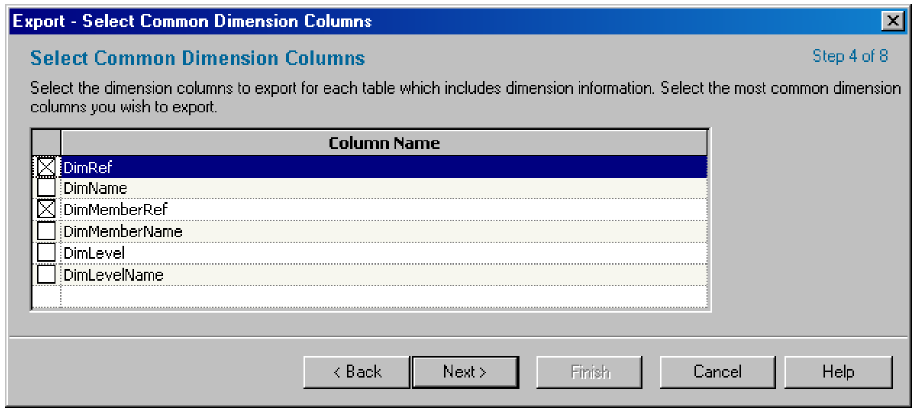
   b. Click in the **Source Table** column, and select a SAS Activity-Based Management table from the drop-down list.

4. Repeat steps 1 and 2 for every table that you want to export.

   Next, you will select dimensions to export for each table that contains dimension information.

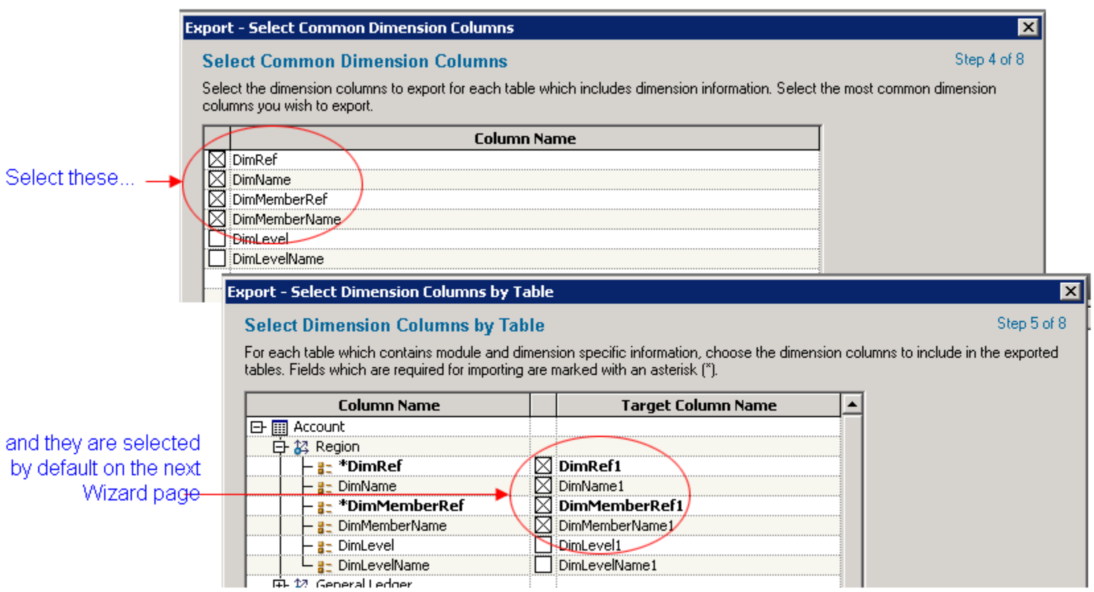
5. Click **Next**.

Step 4 of the Wizard



1. To select dimensions within tables to export, select the check box to the left of the dimension in the **Column Name** column.

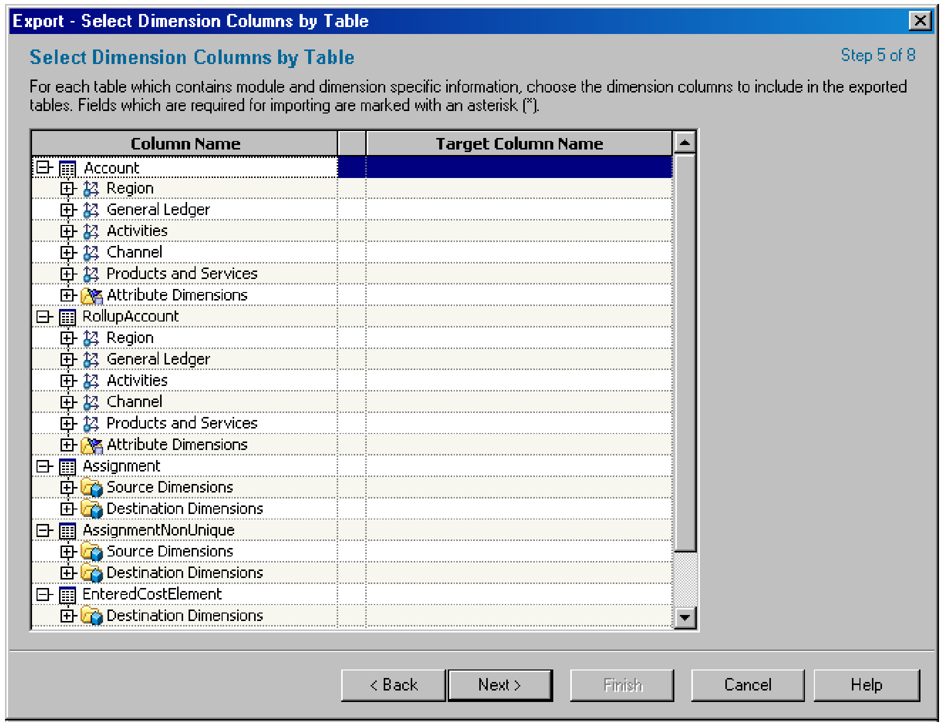
   Columns that you select are selected by default on the next Export Wizard screen. You can, however, change your selection on the next screen. That is, you can deselect a field that you had selected, or select a field that you had deselected.



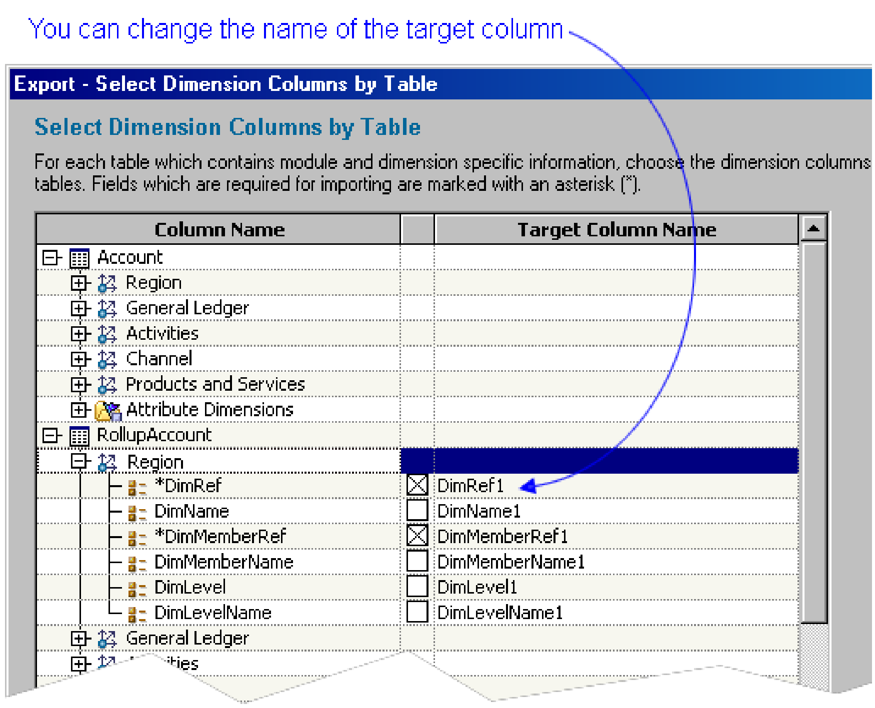
   Next, you will select dimensions and module information to export for each table that contains dimension information or module information.

2. Click **Next**.

Step 5 of the Wizard



1. Expand each table to review which columns are automatically selected to export.



2. To select a column to export, select the check box to the left of the **Target Column Name**.

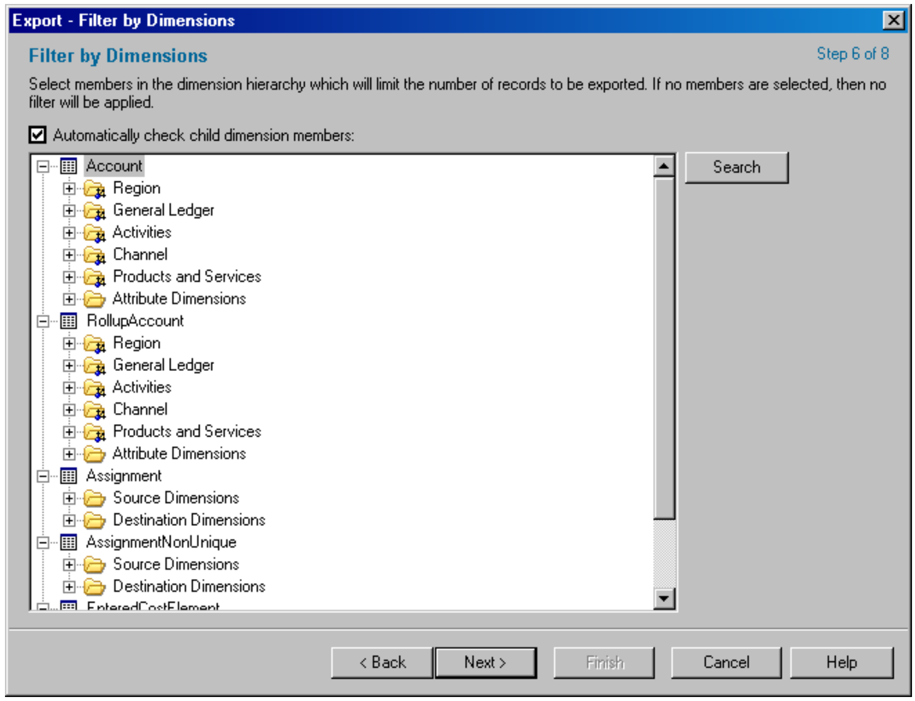
   Notice that columns marked with an asterisk (*) must be exported.

3. To change the name of an exported column, click in the **Target Column Name** column to the right of a column name, and type a new name.

Next, you will select dimension members to export. By default, all data will be exported. By selecting dimension members, you can limit the amount of data that is exported.

4. Click **Next**.

Step 6 of the Wizard

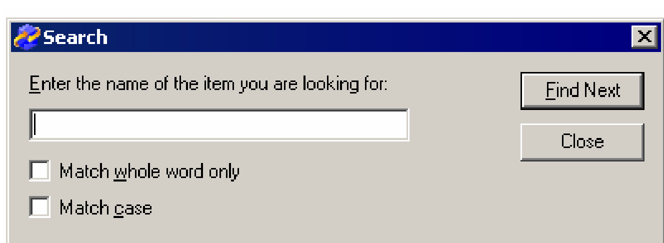


1. Expand each table to view the dimension members.

2. If you want to select all dimension members when a dimension is selected, select the **Automatically check child dimension members** option.
3. To select a dimension member, select the check box to the left of the dimension member.
4. To search for a dimension member, do the following:

   a. Click **Search**.

   The Search dialog box appears.



b. In the **Enter the name of the item you are looking for** box, type the name of the item.

   You can search for a table, a dimension, or a dimension member.

c. Select the **Match whole word only** option and the **Match case** option.

d. Click **Find Next**.

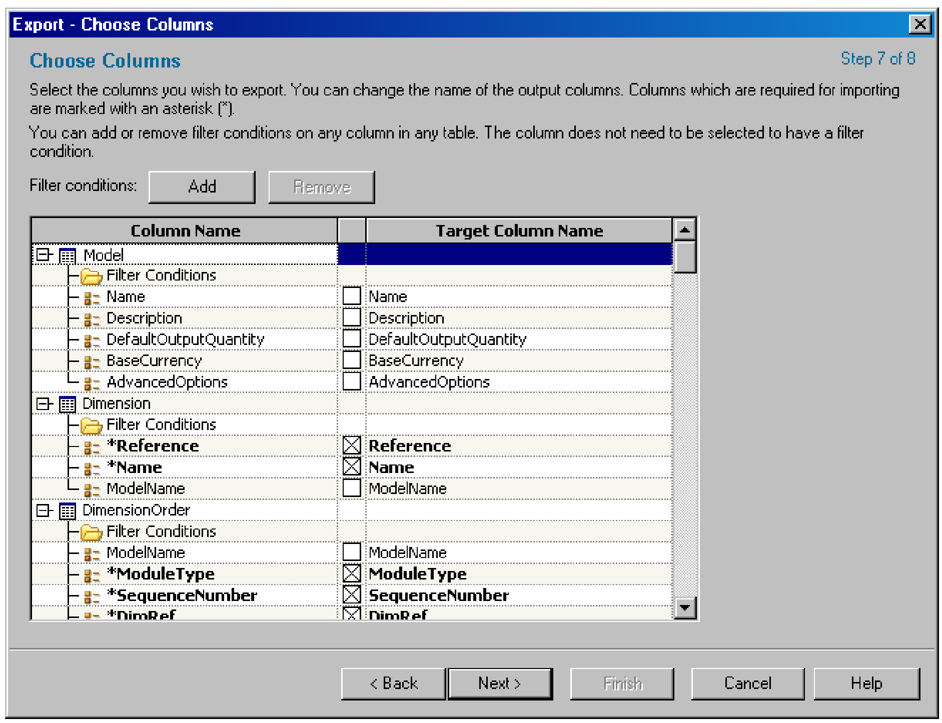
The next occurrence of the item is found.

5. Repeat steps 1 through 3 for each dimension member that you want to export.

   Next, you will select columns to export. You can limit which data to export by creating filters.

6. Click **Next**.

Step 7 of the Wizard



1. To select a column, select the check box to the left of the target column name.

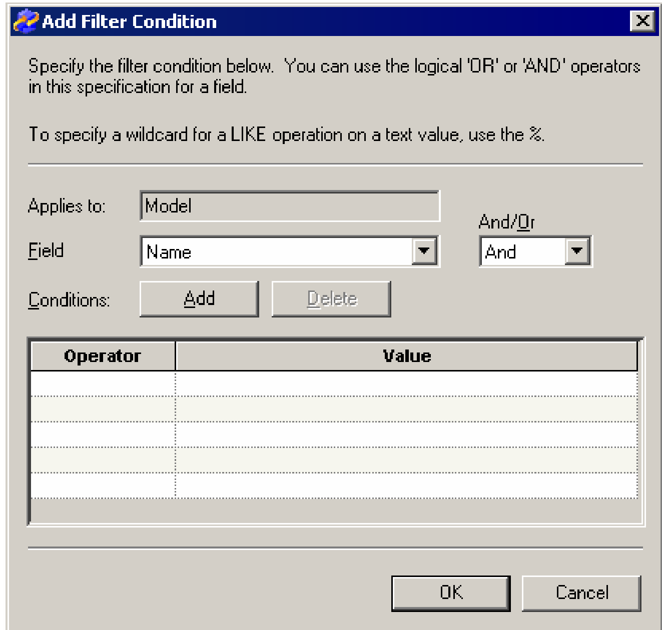
   Notice that columns marked with an asterisk (*) must be exported.

2. To change the name of an exported column, click in the **Target Column Name**column to the right of a column name, and type a new name.

3. To add a filter condition to a table, do the following:

   a. Click **Add**.

      The Add Filter Condition dialog box appears.



b. Select a database **Field** on which to base the filter condition.

c. From the **And/Or** drop-down list, select a logical operator.

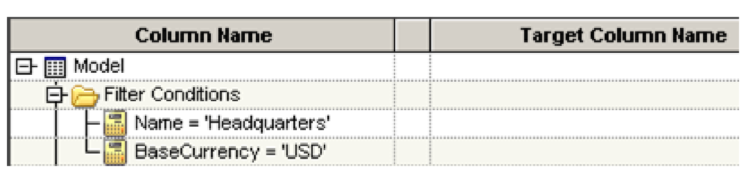
d. Click **Add**.

The field appears in the list.

e. Select an **Operator** from the drop-down list.

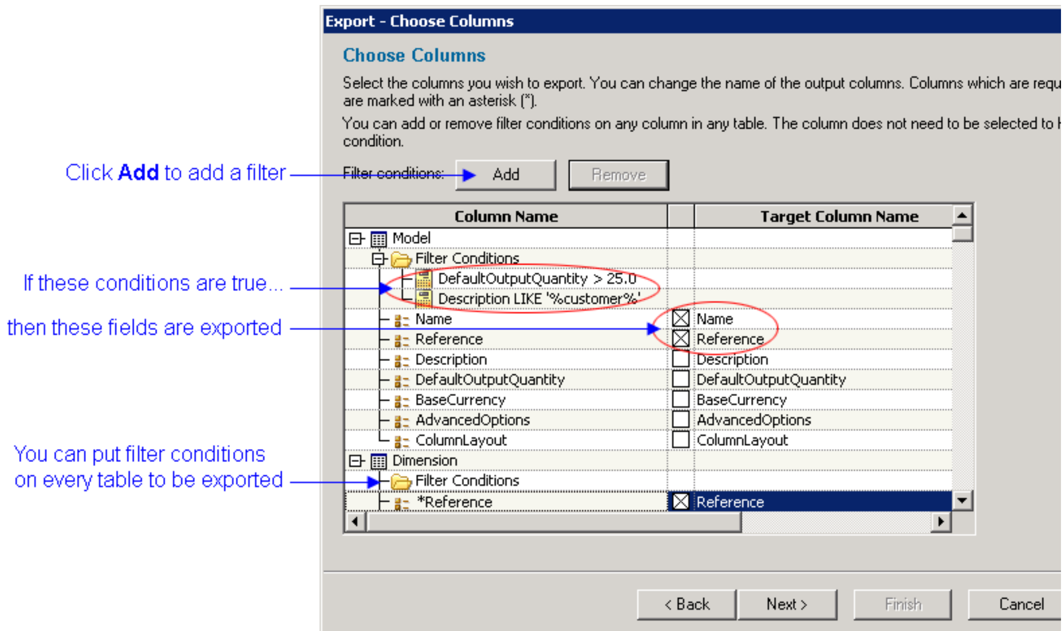
f. Type a **Value** for the operator.

You can create as many filter conditions as needed, but each filter condition can specify only one field. For example, if you need to limit the export to a model named Headquarters with a base currency of United States Dollars, you would have to open the Add Filter Condition dialog box twice. The first time, you would need to create the filter condition for the model name Headquarters. The second time, you would need to create the filter condition for the base currency United States Dollars. The completed filter conditions would look like the following:



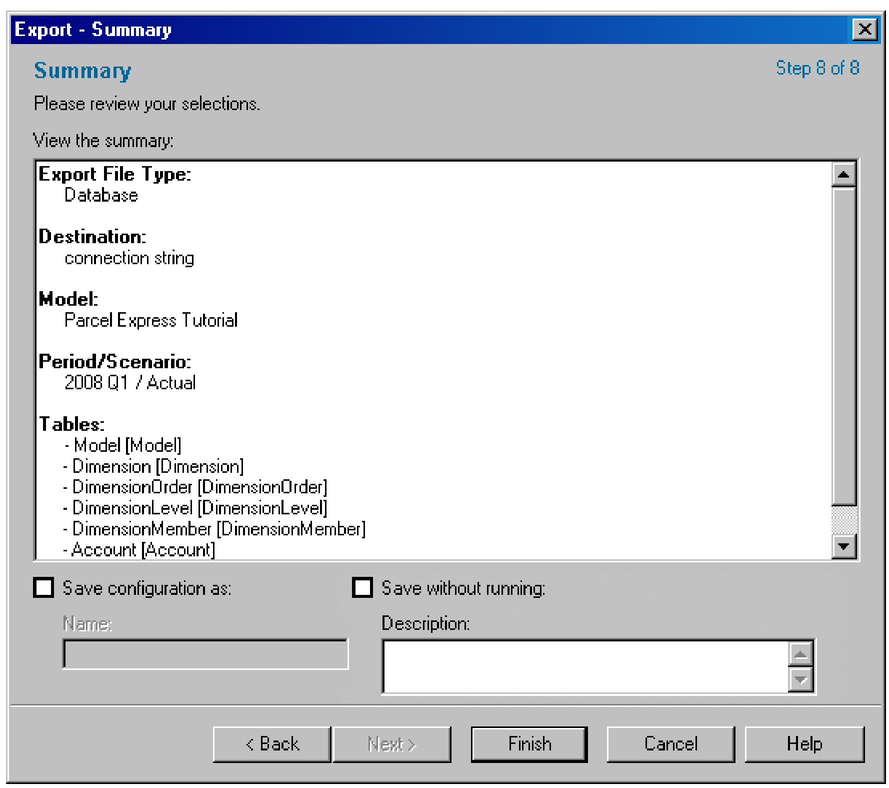
Selected columns are exported only if they pass the filter. In other words, for a column to be exported, it must both:

- be selected
- pass whatever filters exist for the table



4. Click **Next**.

Step 8 of the Wizard



1. Review the export summary.
2. If you need to change any information, click **Back** until you reach the step that you need to change in the wizard.

   All of the information that you have specified is saved. Click **Next** to advance through the wizard.

3. To save the export configuration so that the export can be easily run again, do the following:
   a. Select the **Save configuration as** option.
   b. Type the **Name**.
   c. Type the **Description**.
4. Select **Save without running** to save the export configuration with performing the export.
5. Click **Finish**.

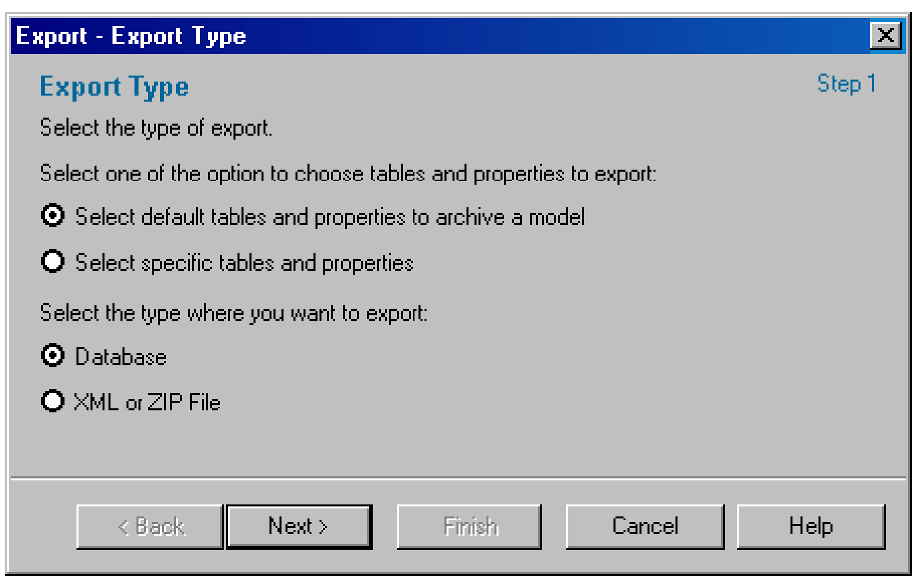
Archive a Model to a Database with the Export Wizard

Before attempting to archive a model to a database, see Chapter 15, "Connecting to a Database," on page 189 which provides information about preparing to connect to the most common databases.

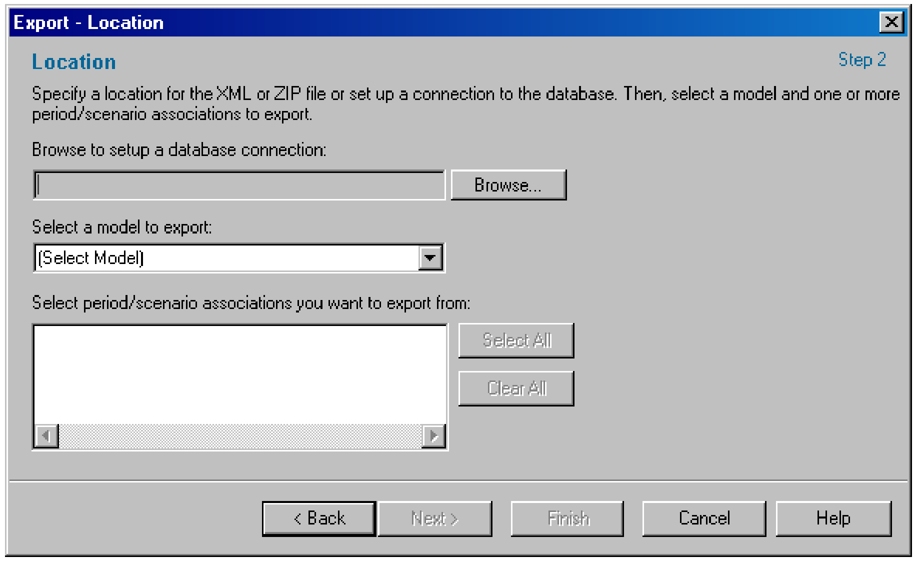
Note: You can perform this task without first opening a model.

1. Verify that the model and an empty database are ready.
2. Select **File** ⇨ **Export Model Data**.

   The Export Wizard appears.



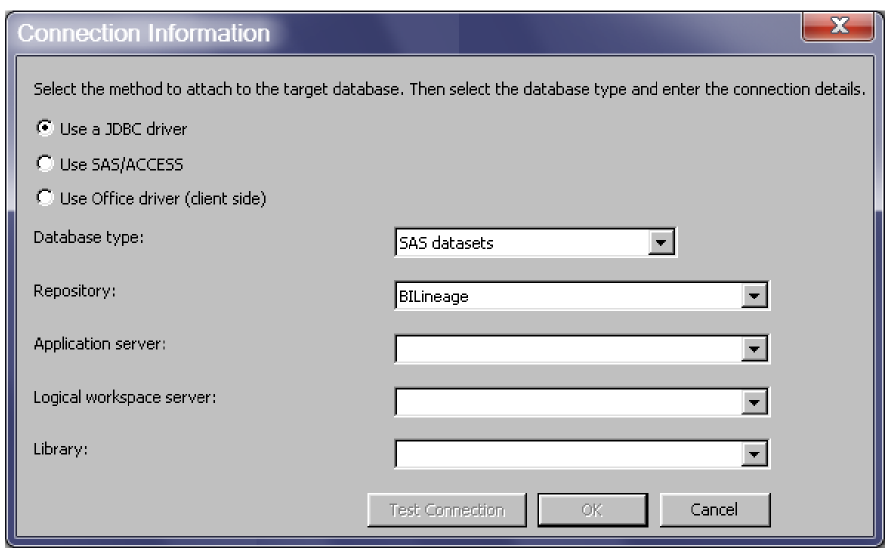
3. Select the **Select default tables and properties to archive a model** option.
4. Select the **Database** option.
5. Click **Next**.



6. Click 

.

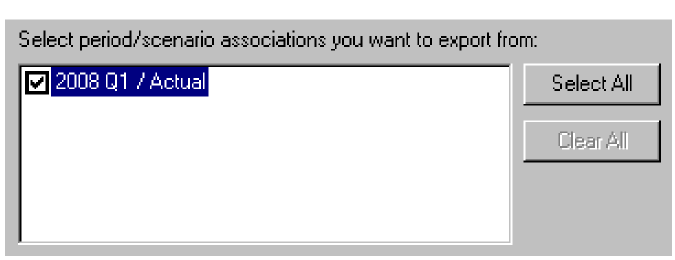
   The Connection Information window opens.



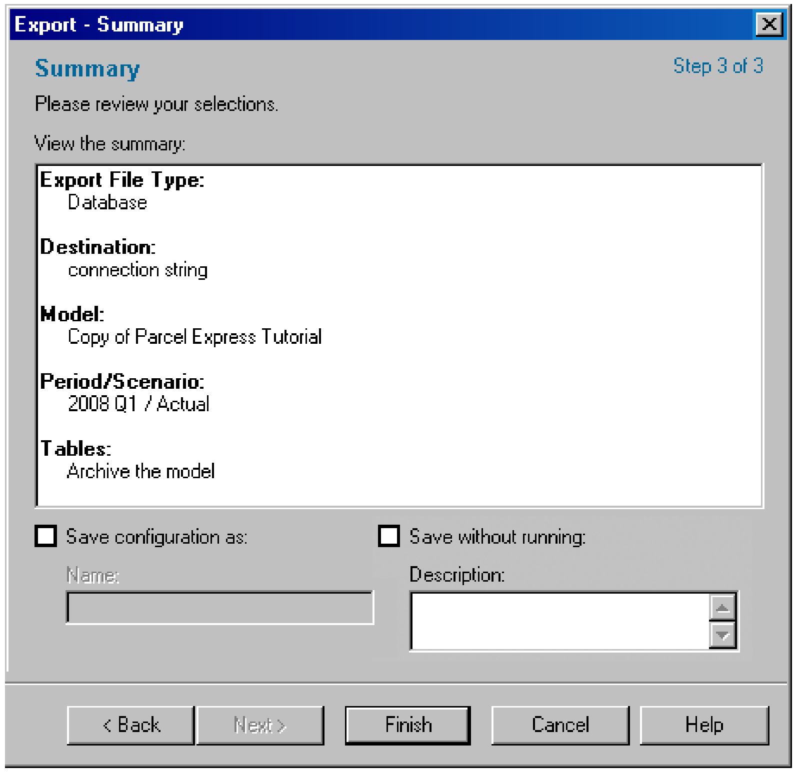
   For information on using this window, see Chapter 15, “Connecting to a Database,” on page 189.

7. From the **Select a model to export** drop-down list, select a model.

8. From the **Select period/scenario associations you want to export from** list, select the check box next to one or more period/scenario associations.



9. Click **Next**.



10. Review the export summary.
11. If you need to change any information, click **Back** until you reach the step that you need to change in the wizard.

    All of the information that you have specified is saved. Click **Next** to advance through the wizard.
12. To save the export configuration so that the export can be easily run again, do the following:
    a. Select the **Save configuration as** option.
    b. Type the **Name**.
    c. Type the **Description**.
13. Select **Save without running** to save the export configuration with performing the export.
14. Click **Finish**.

Chapter 15

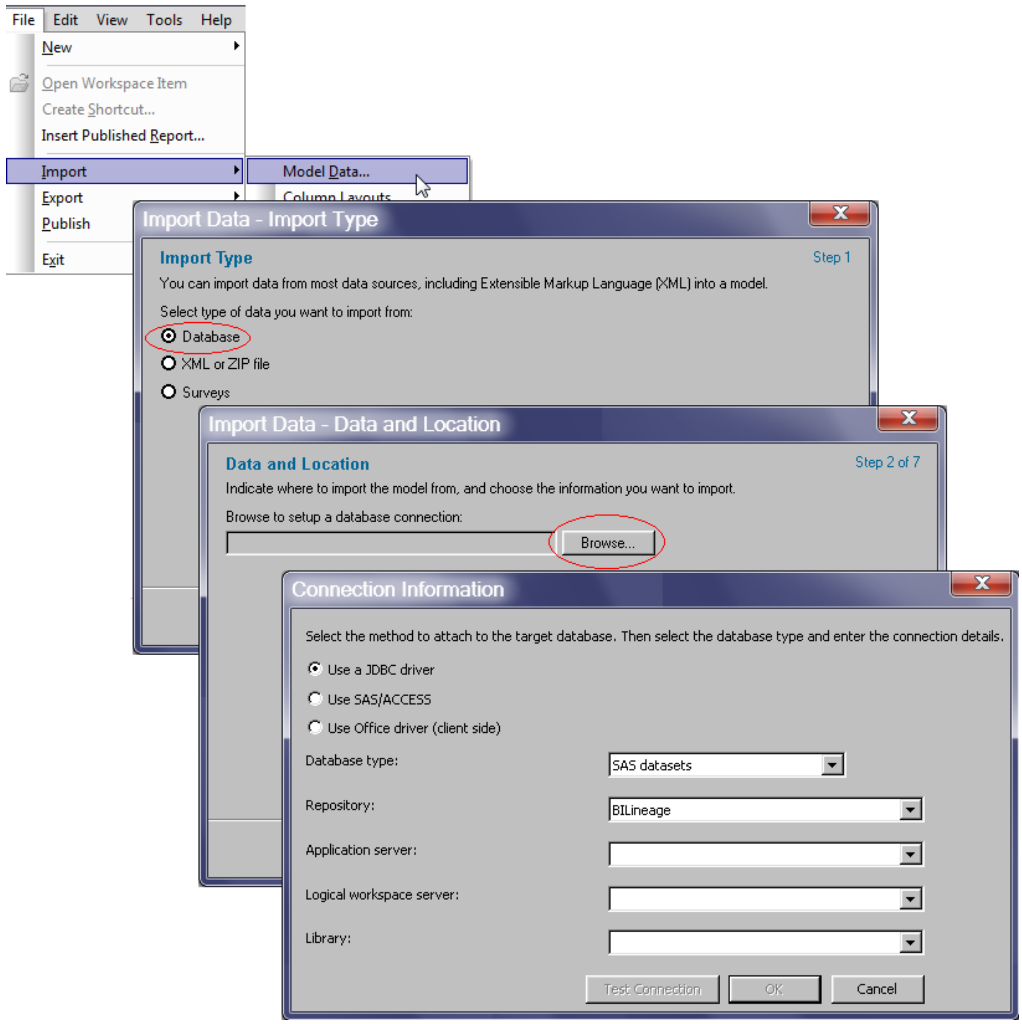
Connecting to a Database

How to Access the Connection Information Dialog

To access the Connection Information dialog:

1. From the **File** menu, select either to import or to export model data:
   - **File ⇨ Import ⇨ Model Data**
   - **File ⇨ Export ⇨ Model Data**
2. Select **Database**.
3. Click **Browse**.

   The Connection Information dialog opens.



About Connecting to a Database

In order to import or export from a database, you must connect to the database. You can connect using either:

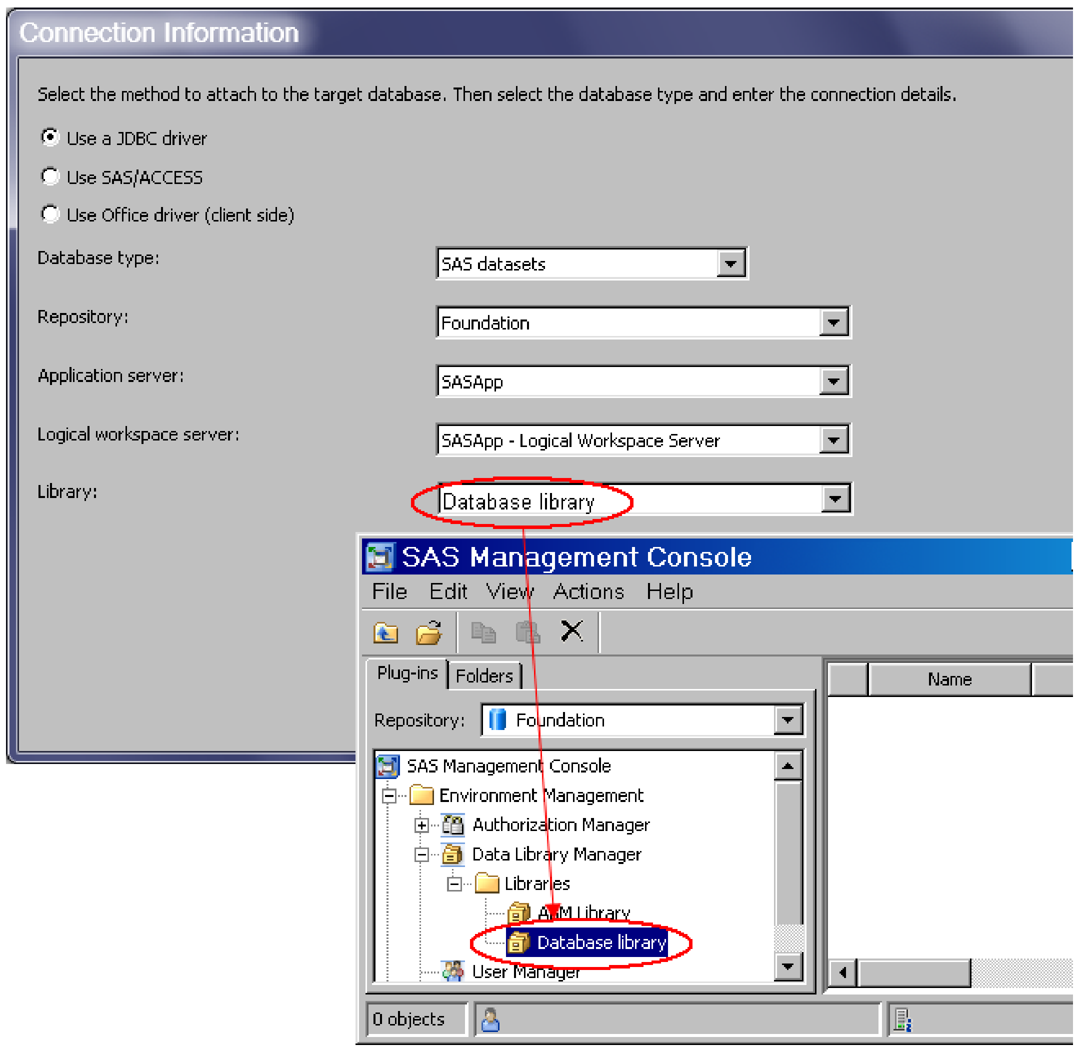
- JDBC driver
  - "JDBC Driver with SAS Datasets" on page 191
  - "JDBC Driver with Microsoft SQL Server" on page 193
  - "JDBC Driver with Oracle" on page 194
  - "JDBC Driver with MySQL" on page 195
  - "JDBC Driver with Another Database" on page 195
- "SAS/ACCESS" on page 196
- "Office Driver (client side)" on page 197

Following are details concerning these two methods of connecting to a database.

Using A JDBC Driver

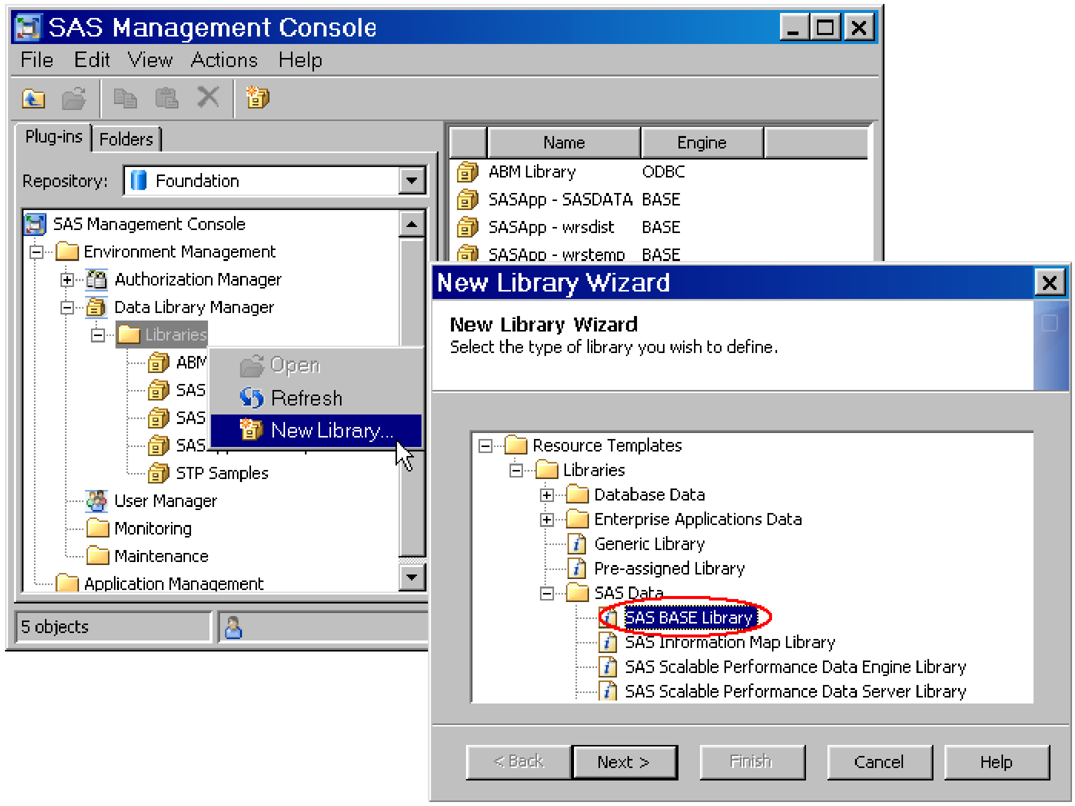
JDBC Driver with SAS Datasets

Before importing from or exporting to SAS datasets, you must create a SAS Base library.



To create the library:

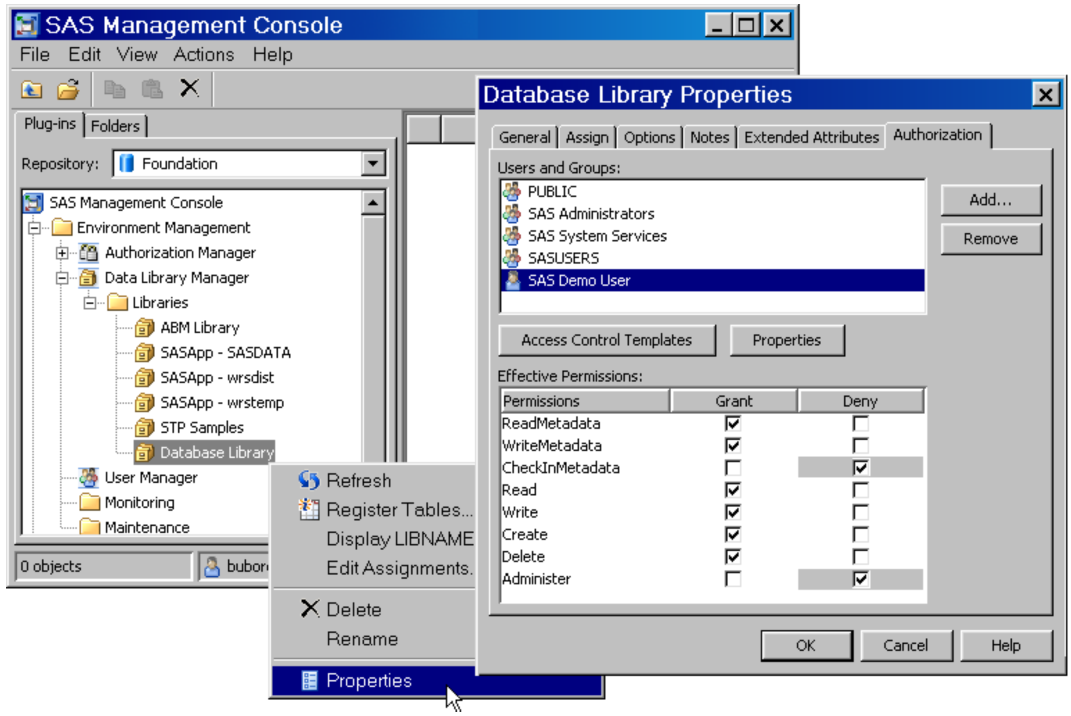
1. Open SAS Management Console connected to your Metadata Server.
2. On the **Plug-ins** tab, right-click **Libraries** (under **Environment Management** ⇨ **Data Library Manager**) and select **New Library**.
3. Select **SAS Base Library**.



The SAS Activity-Based Management user who wants to import from or export to SAS datasets must have the appropriate permissions for the SAS Base Library that you created.

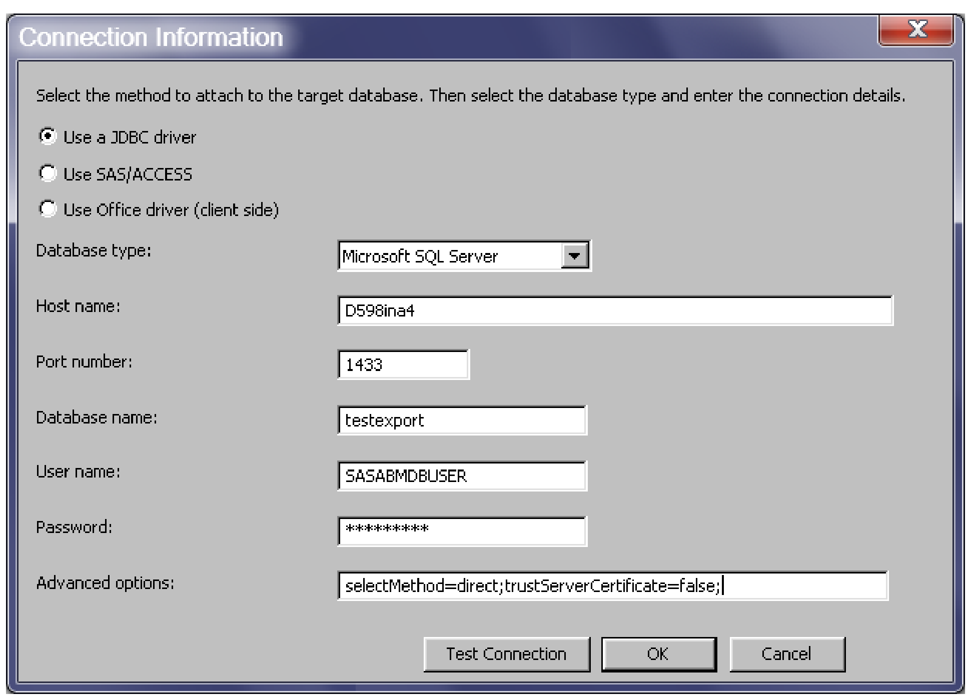
To set the library permissions:

1. Right-click the SAS Base Library that you created, and then select **Properties**.
2. Select the user of SAS Activity-Based Management who will be importing from or exporting to SAS datasets.
3. Set the permissions as shown in the following picture.



JDBC Driver with Microsoft SQL Server

The following picture shows an example of connecting to a Microsoft SQL Server database.



The following are some considerations to keep in mind when connecting to a Microsoft SQL Server database:

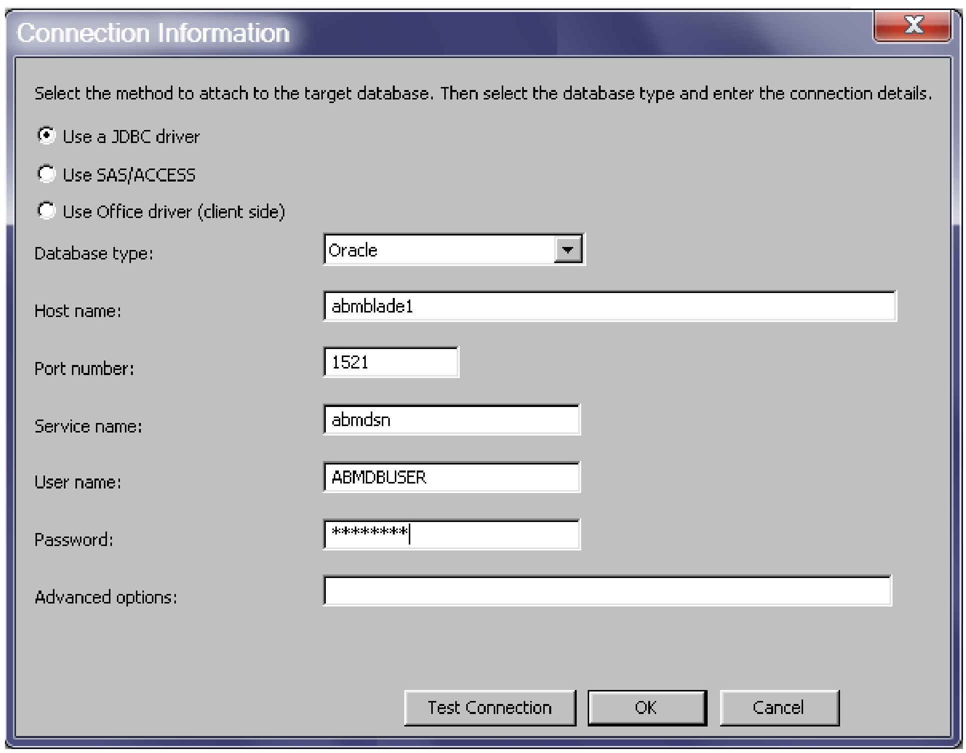
- **Host name** is the MachineName where the SAS Activity-Based Management database is installed.
- **Port** is the TCP port where the SAS Activity-Based Management database instance is running.

  If the database is on a Named Instance, then go to the SQL Server Configuration manager to check the TCP port on which it is running. Named Instances run on a dynamic TCP port.
- If you specify advanced options, separate multiple options with a semicolon. For example:

```
selectMethod=direct;trustServerCertificate=false;
```

JDBC Driver with Oracle

The following picture shows an example of connecting to an Oracle database.



The following are some considerations to keep in mind when connecting to an Oracle database:

- **Host name** is the MachineName where the Oracle client is installed.
- **Port** is the TCP port where the SAS Activity-Based Management database instance is running.
- You can find the values of **Host name** and **Service name** in the file **`tnsnames.ora`**.

```
ABMDSN2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = abmblade1)(PORT = 1521))
    )
```

```
      (CONNECT_DATA =
        (SERVICE_NAME = abmdsn)
      )
    )
```

- Open the **`sqlnet.ora`** file and verify that it has EZCONNECT specified under NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT).

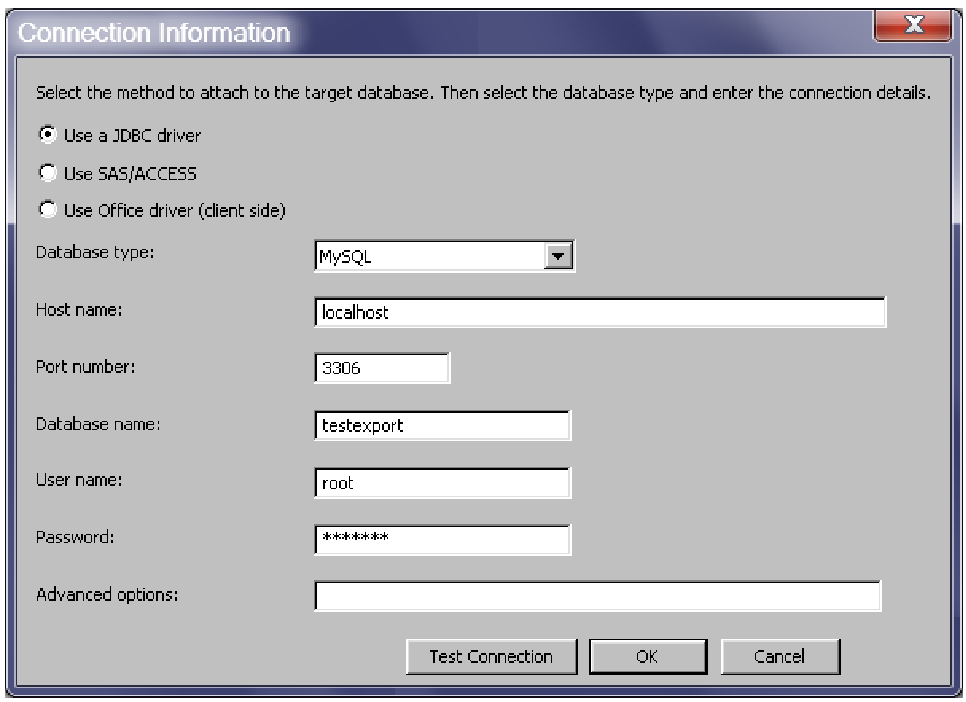
  This file is at the same location as **`tnsnames.ora`**. For example:

  ```
  C:\Oracle\product\11.1.0\client_1\network\admin\sqlnet.ora
  ```

  .

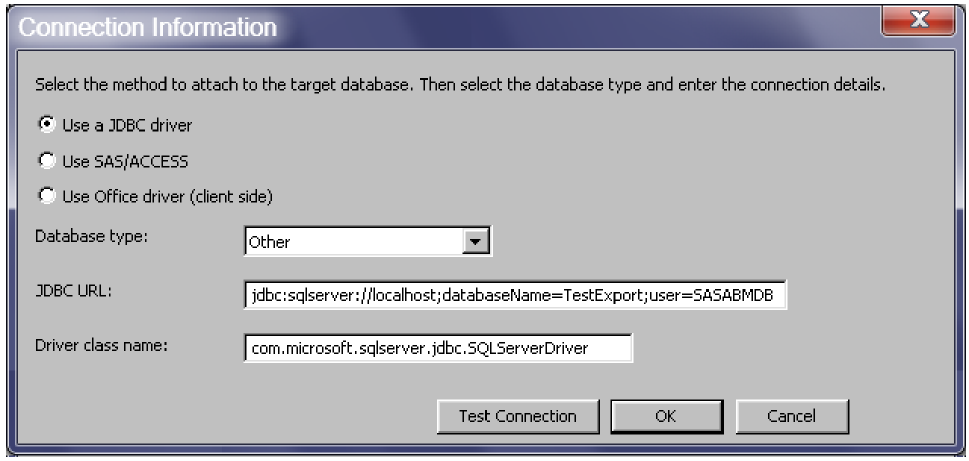
JDBC Driver with MySQL

The following picture shows an example of connecting to a MySQL database



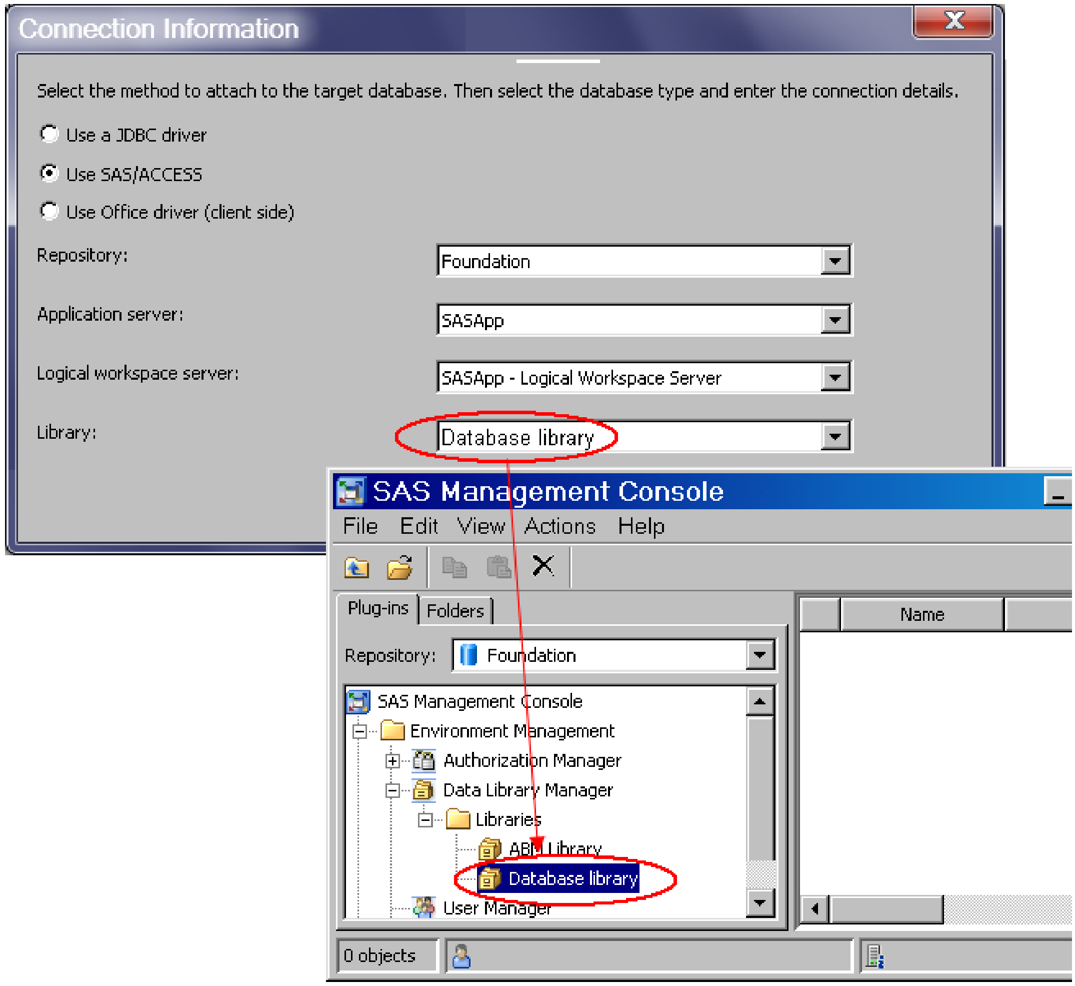
JDBC Driver with Another Database

In order to use a JDBC driver to connect to any other database, you must enter a JDBC connection string to connect to the target database and a JDBC driver class name. (When you connect to one of the previously mentioned databases, SAS Activity-Based Management takes care of creating the JDBC connection string and specifying the driver class name).



SAS/ACCESS

Before using SAS/ACCESS, you must first use SAS Management Console to create a library that points to the target database. The library should have all the required READ/WRITE permissions for the user of SAS Activity-Based Management who will need to connect to the database to import or export.



Notes:

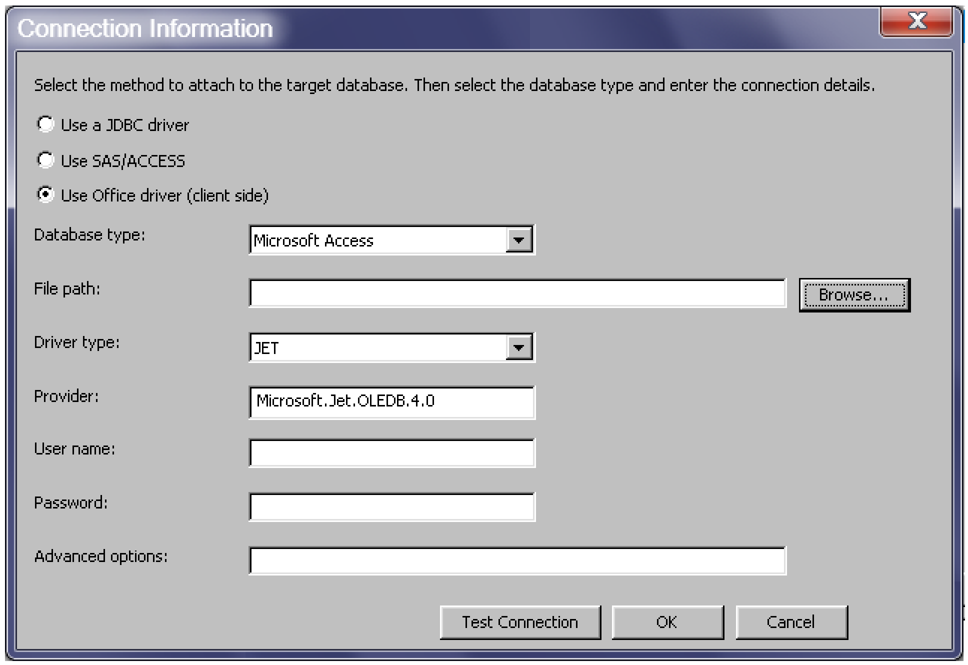
- When you use SAS Management Console to create a library pointing to any database on your Metadata Server, you must specify the maximum text length as an advanced option. The maximum text length should be longer than the longest column in your target database table used for import or export. See "Specifying the Maximum Text Length" on page 200.
- When you use SAS Management Console to create a library for Microsoft Excel on your Metadata Server, you must specify SCAN_TEXT=NO as an advanced option. This option is only required for Microsoft Excel. See "Specifying SCAN_TEXT=NO as Advanced Option" on page 204.
- When you create a library to export to Microsoft Excel, always use a clean Excel file. The system cannot remove the existing data from Excel before doing the export.

Office Driver (client side)

You can import from or export to Microsoft Access and Microsoft Excel using a client-side driver. This has the advantage that you can now import from or export to client installations of Microsoft Office even if your SAS Activity-Based Management server is on a UNIX system where the Microsoft Office drivers would not work.

Microsoft Access

The following picture shows an example of connecting to Microsoft Access. The file path that you enter should have READ/WRITE permission.

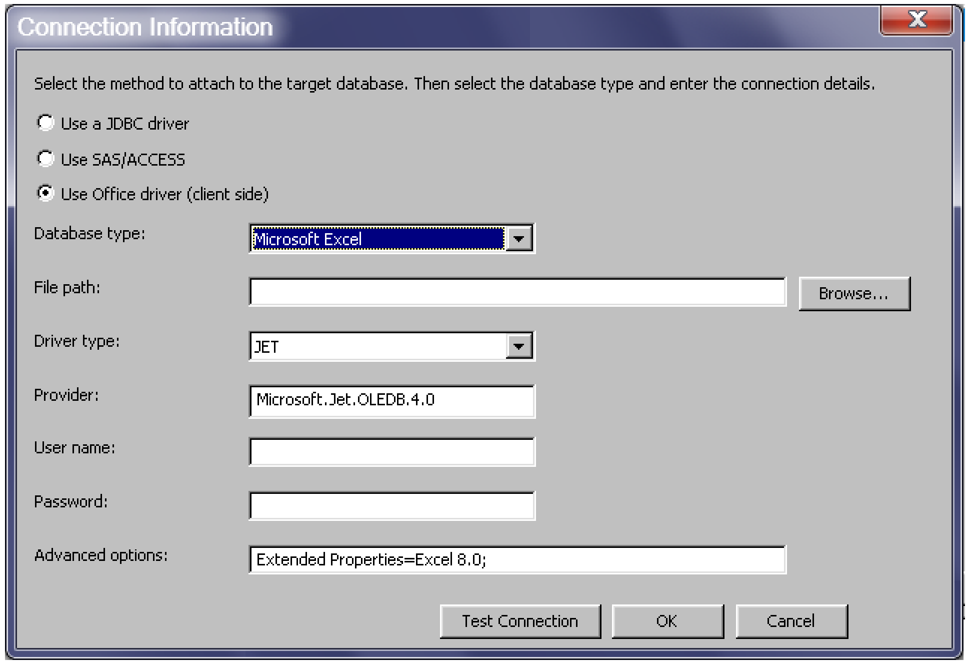


Notes:

- The JDBC driver for Microsoft Access must use a bridge to ODBC. This means that you must install the appropriate ODBC driver on the SAS Activity-Based Management server machine. See "Installing the ODBC Driver for Microsoft Access and Excel" on page 201.
- The directory that you specify for importing from or exporting to must have the proper permissions. See "Permissions for Microsoft Access and Microsoft Excel" on page 202.
- If SAS Activity-Based Management client and server are on same machine, then you can use a local path. For example: `c:\accessexport\test.mdb`. If they are on different machine, then you should use the network path. For example: `\\d12345\accessexport\test.mdb`.

Microsoft Excel

The following picture shows an example of connecting to Microsoft Excel. The file path that you enter should have READ/WRITE permission.



Notes:

- The JDBC driver for Microsoft Excel must use a bridge to ODBC. This means that you must install the appropriate ODBC driver on the SAS Activity-Based Management server machine. See "Installing the ODBC Driver for Microsoft Access and Excel" on page 201.

- The directory that you specify for importing from or exporting to must have the proper permissions. See "Permissions for Microsoft Access and Microsoft Excel" on page 202.

- If SAS Activity-Based Management client and server are on same machine, then you can use a local path. For example:

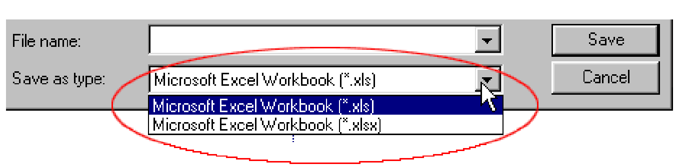
  ```
  c:\excelexport\test.xls
  ```

  . If they are on different machine, then you should use the network path. For example:

  ```
  \\d12345\excelexport\test.xls
  ```

  .

- You can save the Excel file as either *.xls(2003 format) or *.xlsx (2007 format).



- When you export to Microsoft Excel, always use a clean Excel file. The system cannot remove existing data from Excel before doing the export.

Connection Options

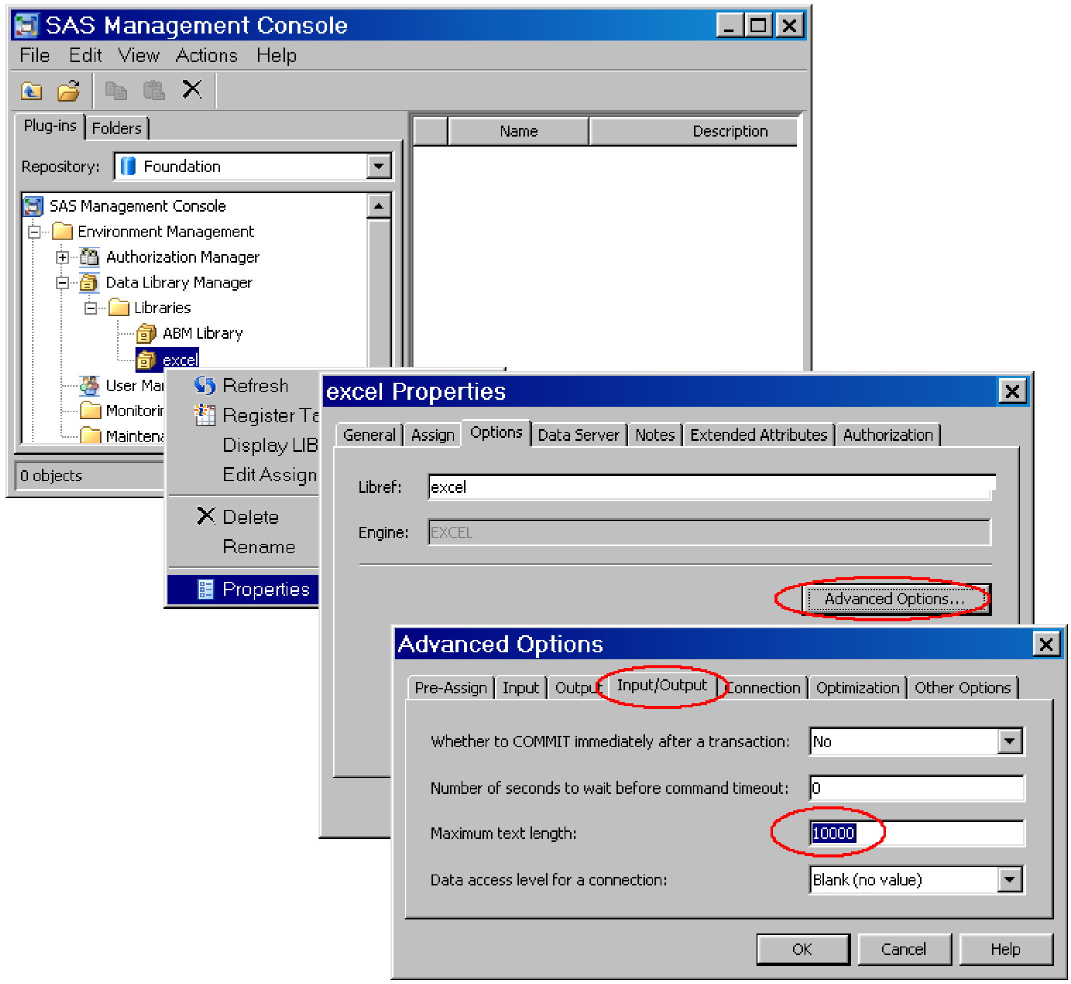
Specifying the Maximum Text Length

When you use SAS Management Console to create a library pointing to any database on your Metadata Server, you must specify the maximum text length as an advanced option. The maximum text length should be longer than the longest column in your target database table used for import and export. Otherwise the system will trim longer columns and you will lose data.

To set the advanced option:

1. In the **Plug-ins** tab of SAS Management Console, right-click the library, and then select **Properties**.
2. Click the **Options** tab of the Properties window.
3. Click **Advanced Options**.
4. Click the **Input/Output** tab.
5. Set the Maximum text length to be longer than any column in your table.

   Note: The following picture shows setting the maximum text length for an Excel library. The steps are the same for a different database. Also note that the length shown in the picture is just an example. Make sure that the length you specify is greater than the longest column in your table.



Installing the ODBC Driver for Microsoft Access and Excel

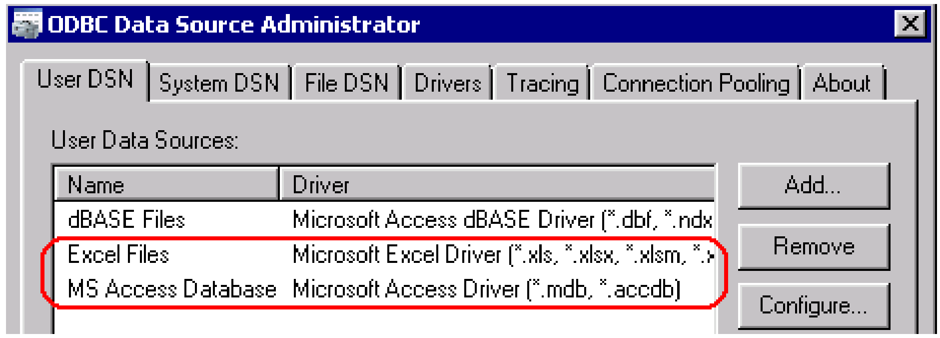
The JDBC driver for Microsoft Access and Microsoft Excel must use a bridge to ODBC. You must install the appropriate ODBC driver on the SAS Activity-Based Management server machine to import from Access or Excel.

The following providers are supported:

- Microsoft.ACE.OLEDB.12.0 for use with either Microsoft Office 2007 or Microsoft Office 2010
- Microsoft.Jet.OLEDB.4.0 for use with JET

To display a list of ODBC drivers that are already installed, do the following:

1. Select **Start** ⇨ **Control Panel** on the SAS Activity-Based Management client machine.
2. Select **Administrative Tools**.
3. Select **Data Sources (ODBC)**.
4. Click the **Drivers** tab.
5. Verify that drivers for Microsoft Access and Microsoft Excel are listed as shown.



If the drivers are not installed, you can install them by running AccessDatabaseEngine.exe, which you can obtain from the Microsoft Download Center at **`http://www.microsoft.com/downloads/`**. Because the SAS Activity-Based Management client is a 32–bit application and can work only with 32–bit drivers, there are four cases to consider when downloading the drivers:

You have not installed Microsoft Office
: You can download the 32–bit drivers for Office 2010. On the Microsoft Download Center, search for "Microsoft Access Database Engine 2010 Redistributable" and select the 32–bit verson.

You have installed Microsoft Office 2007, 32–bit
: If you do not already have the drivers, you can download them. On the Microsoft Download Center, search for "2007 Office System Driver: Data Connectivity Components". Or search for "Microsoft Access Database Engine 2010 Redistributable" and select the 32–bit version (the 32–bit driver for Office 2010 works with Office 2007).

You have installed Microsoft Office 2010, 32–bit
: If you do not already have the 32–bit drivers, you can download them. On the Microsoft Download Center, search for "Microsoft Access Database Engine 2010 Redistributable" and select the 32–bit version.

You have installed Microsoft Office 2010, 64–bit
: If you have installed the 64–bit drivers, then you need to install the 32–bit drivers. However, Microsoft does not allow you to install 32–bit drivers for Office 2010 along with 64–bit drivers. Therefore, you must install the 32–bit drivers from Microsoft Office 2007 instead. On the Microsoft Download Center, search for "2007 Office System Driver: Data Connectivity Components".

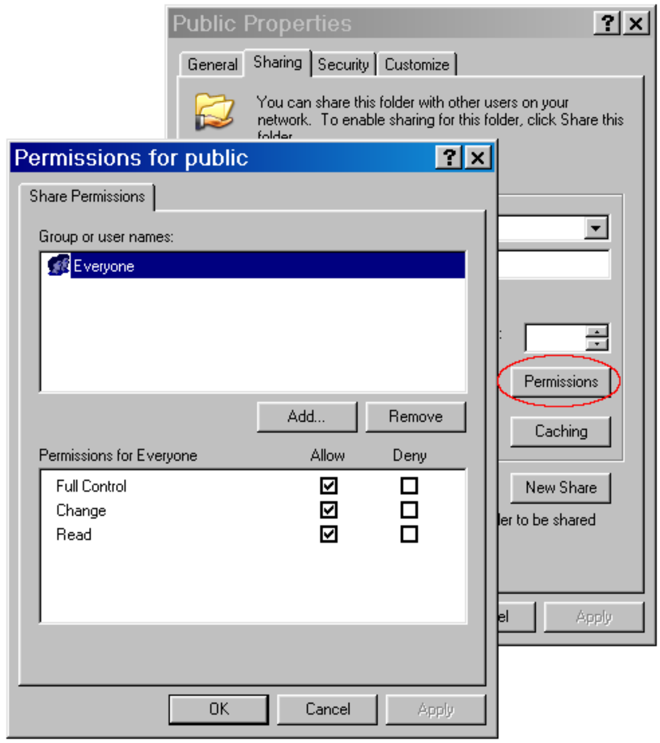
: *Note:* The 32–bit drivers for Microsoft Office 2007 do not replace the 64–bit drivers for Microsoft Office 2010, so the 64–bit drivers remain available for use with applications other than SAS Activity-Based Management.

Permissions for Microsoft Access and Microsoft Excel

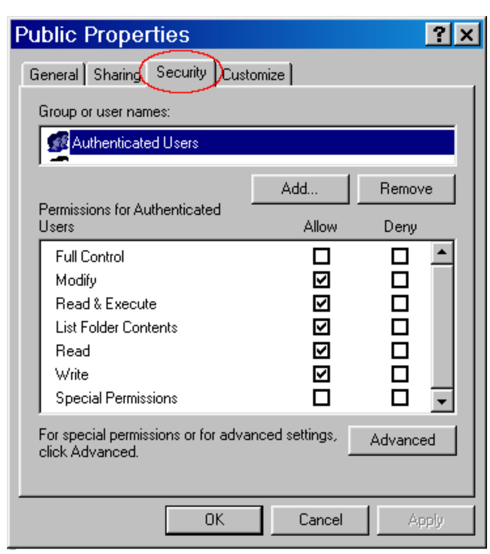
When you import from Microsoft Access or Microsoft Excel or export to either one, the directory that you import from or export to must have the proper permissions. To set the permissions, do the following:

1. In Windows Explorer, right-click the directory and select **Properties**.
2. Click the **Sharing** tab and select **Share this folder**.
3. Click **Permissions** and allow the following permissions:
   - Full Control

- Change
- Read



4. Click the **Security** tab and allow the following permissions:
   - Modify
   - Read & Execute
   - List Folder Contents
   - Read
   - Write

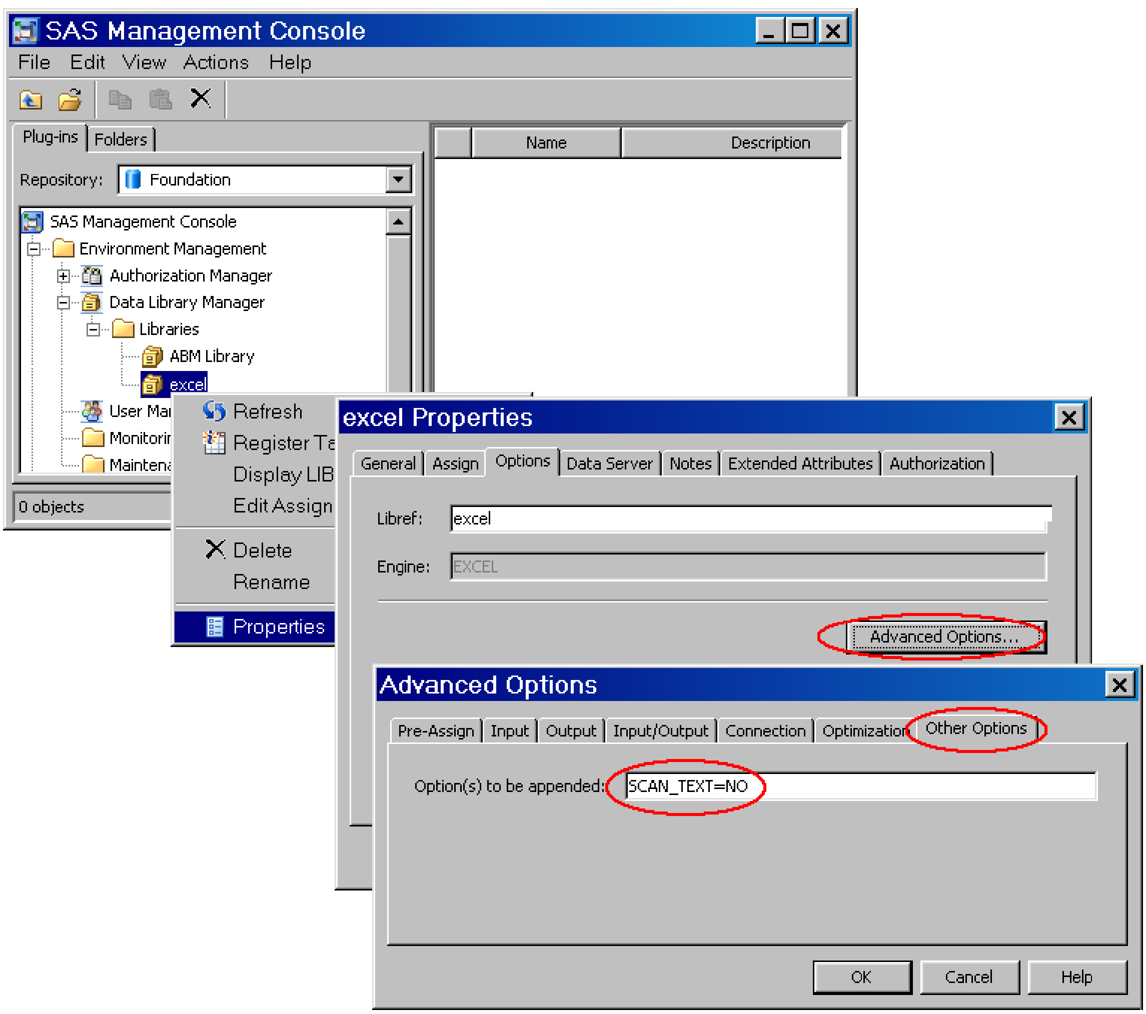


Specifying SCAN_TEXT=NO as Advanced Option

When you use SAS Management Console to create a library for Microsoft Excel on your Metadata Server, you need to specify SCAN_TEXT=NO as an advanced option.

To set the advanced option:

1. In the **Plug-ins** tab of SAS Management Console, right-click the **Excel** library, and then select **Properties**.
2. Click the **Options** tab of the Properties window.
3. Click **Advanced Options**.
4. Click the **Other Options** tab.
5. In the **Option(s) to be appended** field, type SCAN_TEXT=NO.



Chapter 16

Staging-Table Schemas

Introduction

The following staging tables define the SAS Activity-Based Management data schema. Each table corresponds to a specific structural or periodic aspect of a model, such as period , scenario, dimension, and so on. Use this data schema to create a database that will hold the data that you want to import into a model. This data schema is also used by SAS Activity-Based Management to export models to XML files.

Staging Tables for Importing a New Model

When you create a new model by importing data , you must import at least the following tables:

- Dimension
- DimensionMember
- DimensionLevel
- DimensionOrder

Periodic Data versus Structural Data

Staging tables are distinguished by whether they contain periodic or structural data. Periodic data is model data which is stored separately for each period/scenario association. Structural data is the model data which is independent of any period/ scenario association. It is data which is common to all period/scenario associations.

Periodic data

The following staging tables contain periodic data:

- Account
- Assignment
- CurrencyRate
- DimMemberDimAttrAssociation
- DimMemberValueAttrAssociation
- ExternalUnit
- EnteredCostElement
- PerformanceMeasure
- ValueAttributeAssociation
- ValueAttributePeridicDef
- DimensionalAttributeAssociation

When importing data for an already-existig model, you can speed the import by importing only periodic data (see “Periodic Import” on page 167.).

Structural data

The following staging tables contain structural data:

- Dimension
- DimensionMember
- DimensionLevel
- DimensionOrder
- Driver
- Model
- ValueAttributes
- Period
- PeriodLevel
- Scenario
- Scenariolevel

General notes

Remember the following information when referring to these tables:

- The tables are organized based on the general order in which models are created during import; however, the actual order of model creation is not explicitly implied by the table sequence.
- All Reference and Name field string comparisons are not case sensitive.
- The identifier for periods and scenarios is Period.Reference and Scenario.Reference.
- In XML, a database null value is represented by $NULL$. Null values and empty strings are handled in the same way. Null numbers are typically imported to a model as is.
- The Reference and Name fields for hierarchical items (such as Period, Scenario, and DimMember) must comply with the naming conventions (see the online help).
- The fields that are noted as Required for archive are the fields that are exported by default.
- When a table is exported to a database, most of the names of the exported fields are the same as the names listed in this section. However, some names are changed when exported; those changes are noted in the Field Name column of each table. SAS Activity-Based Management attempts to match the destination column names with the source column names as closely as possible, so this name change is necessary only when a database imposes certain naming limitations (such as a limited number of characters).

Data Types

The SAS Activity-Based Management data schema uses the following data types:

Alphanumeric
: is a text value that translates to values such as varchar, nvarchar, char, or nchar.

Date
: is a regular binary date value that must be recognized by ADO.NET.

DriverSignature
: is a driver type that is identified by a series of ModuleType and DriverName pairs.

Float
: is an 8-byte floating point number.

Integer
: is a 4-byte signed integer.

Memo
: is a large text-based value that translates to the text or ntext data types in Microsoft SQL Server.

Boolean
: is a Boolean value. The actual value may be a value other than True/False, such as Yes/No, bit, or a Char field. SAS Activity-Based Management will attempt to interpret Boolean values in other formats.

DimensionSignature
: is a series of one or more pairs of the following dimension and dimension member reference information for each dimension, where (n) represents the dimension level:

```
DimRef(n)
DimMemberRef(n)
```

The number of pairs that is used for importing is the maximum number of structural dimensions that is used in the modules being imported. For example, if the Resource module uses two dimensions and the Cost Object module uses three dimensions, the number of DimRef and DimMemberRef pairs must be three. The following example is from a model that tracks the food preparation costs for an airline.

```
DimRef1="Customer"
DimMemberRef1="WF Portland to CHI"
DimRef2="Products"
DimMemberRef2="Omelette"
```

The dimensions that are included in the Dimension Signature data type are only structural dimensions. Dimension attributes are imported and exported using the DimensionAttributeAssociation table.

ExportDimensionSignature

A series of sets of the following dimension information for each dimension, where (n) represents the dimension level:

```
DimName(n)
DimRef(n)
DimMemberName(n)
DimMemberRef(n)
DimLevel(n)
DimLevelName(n)
```

The following example is from a model that tracks the food preparation costs for an airline:

```
DimName1="Customer"
DimRef1="Customer"
DimMemberName1="WF Portland to CHI"
DimMemberRef1="WF Portland to CHI"
DimLevel1="1"
DimLevelName1="Customer L1"
DimName2="Products"
DimRef2="Products"
DimMemberName2="Omelette"
DimMemberRef2="Omelette"
DimLevel2="1"
DimLevelName2="Products L1"
```

Currency Codes

The following table lists the three-letter currency codes that are used by the import/export data schema. Currency codes are used in the following items:

- The Model table
- The CurrencyRate table
- The XML calculate configuration schema

Currency	Code
Afghani	AFA
Albania Lek	ALL
Algerian Dinar	DZD

Currency	Code
Argentine Peso	ARS
Ariary	MGA
Armenian Dram	AMD
Aruban Guilder	AWG
Australian Dollar	AUD
Azerbaijanian Manat	AZM
Bahamian Dollar	BSD
Bahraini Dinar	BHD
Baht	THB
Balboa	PAB
Barbados Dollar	BBD
Belarus Ruble	BYR
Belgian Franc	BEF
Belize Dollar	BZD
Bermudian Dollar (Bermuda Dollar)	BMD
Bolivar	VEB
Boliviano	BOB
Brazilian Real	BRL
Brunei Dollar	BND
Burundi Franc	BIF
Canadian Dollar	CAD
Cape Verde Escudo	CVE
Cayman Islands Dollar	KYD
Cedi	GHC
CFA Franc BCEAO	XOF
CFA Franc BEAC	XAF
CFP Franc	XPF
Chilean Peso	CLP
Colombian Peso	COP

Currency	Code
Comoro Franc	KMF
Cordoba Oro	NIO
Costa Rican Colon	CRC
Cuban Peso	CUP
Cyprus Pound	CYP
Czech Koruna	CZK
Dalasi	GMD
Danish Krone	DKK
Denar	MKD
Deutsche Mark	DEM
Dinars	CSD
Djibouti Franc	DJF
Dobra	STD
Dominican Peso	DOP
Dong	VND
Drachma	GRD
East Caribbean Dollar	XCD
Egyptian Pound	EGP
Ekwele	CQE
El Salvador Colon	SVC
Ethiopian Birr	ETB
euro	EUR
Falkland Pound	FKP
Fiji Dollar	FJD
Forint	HUF
French Franc	FRF
Gibraltar Pound	GIP
Gourde	HTG
Guarani	PYG

Currency	Code
Guernsey Pounds	GGP
Guinea Franc	GNF
Guinea-Bissau Peso	GWP
Guyana Dollar	GYD
Hong Kong Dollar	HKD
Iceland Krona	ISK
Indian Rupee	INR
Iranian Rial	IRR
Iraqi Dinar	IQD
Irish Punt	IEP
Isle of Man Pounds	IMP
Italian Lira	ITL
Jamaican Dollar	JMD
Jersey Pounds	JEP
Jordanian Dinar	JOD
Kenyan Shilling	KES
Kina	PGK
Kip	LAK
Kroon	EEK
Kuna	HRK
Kuwaiti Dinar	KWD
Kwacha	MWK
Kwacha	ZMK
Kwanza	AOA
Kyat	MMK
Lari	GEL
Latvian Lats	LVL
Lebanese Pound	LBP
Lempira	HNL

Currency	Code
Leone	SLL
Leu	ROL
Leva	BGN
Liberian Dollar	LRD
Libyan Dinar	LYD
Lilangeni	SZL
Lithuanian Litas	LTL
Loti	LSL
Luigini	SPL
Luxembourg Franc	LUF
Malagasy Franc	MGF
Malaysian Ringgit	MYR
Mali franc	MLF
Maltese Lira	MTL
Manat	TMM
Markka	FIM
Mauritius Rupee	MUR
Metical	MZM
Mexican Peso	MXN
Moldovan Leu	MDL
Moroccan Dirham	MAD
Naira	NGN
Nakfa	ERN
Namibia Dollar	NAD
Nepalese Rupee	NPR
Netherlands	ANG
Netherlands Guilder	NLG
New Dinar	YUM
New Israeli Sheqel	ILS

Currency	Code
New Kwanza	AON
New Lei	RON
New Taiwan Dollar	TWD
New Zaire	ZRN
New Zealand Dollar	NZD
Ngultrum	BTN
North Korean Won	KPW
Norwegian Krone	NOK
Nuevo Sol	PEN
Ouguiya	MRO
Pa'anga	TOP
Pakistan Rupee	PKR
Pataca	MOP
Peso Uruguayo	UYU
Philippine Peso	PHP
Portuguese Escudo	PTE
Pound Sterling	GBP
Pula	BWP
Qatari Rial	QAR
Quetzal	GTQ
Rand	ZAR
Rial Omani	OMR
Riel	KHR
Rufiyaa	MVR
Rupiah	IDR
Russian Ruble	RUB
Rwanda Franc	RWF
Saudi Riyal	SAR
Schilling	ATS

Currency	Code
Seychelles Rupee	SCR
Singapore Dollar	SGD
Slovak Koruna	SKK
Solomon Islands Dollar	SBD
Som	KGS
Somali Shilling	SOS
Somoni	TJS
South Korean Won	KRW
Spanish Peseta	ESP
Sri Lanka Rupee	LKR
St Helena Pound	SHP
Sucre	ECS
Sudanese Dinar	SDD
Surinam Guilder	SRG
Suriname Dollars	SRD
Swedish Krona	SEK
Swiss Franc	CHF
Syrian Pound	SYP
Tajik Ruble	TJR
Taka	BDT
Tala	WST
Tanzanian Shilling	TZS
Tenge	KZT
Timor Escudo	TPE
Tolar	SIT
Trinidad and Tobago Dollar	TTD
Tugrik	MNT
Tunisian Dinar	TND
Turkish Lira	TRL

Currency	Code
UAE Dirham	AED
Uganda Shilling	UGX
Ukrainian Hrivna	UAH
US Dollar	USD
Uzbekistan Sum	UZS
Vatu	VUV
Yemeni Rial	YER
Yen	JPY
Yuan Renminbi	CNY
Zimbabwe Dollar	ZWD
Zloty	PLN

Table legend

All tables in the data schema contain the following columns:

This column	Holds this information
Field Name	Represents a value in a model. An external field may be mapped to this name.
Type of Data	Specifies the type of data stored in the field. See “Data Types” on page 208.
Length	Defines the maximum length of the field. memo is for long text values that can be virtually unlimited; it translates to the Microsoft SQL Server text and ntext types.
Key	Specifies that the field uniquely identifies an item. These values are required whenever inserting, updating, or deleting an item. In some tables, there is more than one way to identify an item. In those cases, there is a key1 and key2. When identifying the item, you must specify all key1 fields or all key2 fields.
Import/Add	Identifies whether the field is required or optional for import. N/A means the field is not valid for import. Import allows the insertion of new items and the updating or merging of existing items. For updating or merging, only key fields are required. In some tables there is a Required1 and Required2. The user must provide all Required1 fields or all Required2 fields together with all Required fields for inserting a new item.

This column	Holds this information
Archive	Indicates whether the field is optional or required for archiving a model by exporting it to staging tables. (See "Archiving a Model" on page 170.)
Delete	Indicates whether the field is required for deleting a row.

Notes on specific fields, if any, follow the table.

Naming Conventions

General naming conventions

The name of any item must conform to the following rules:

- Names cannot contain this character: |
- Any item, such as a dimension, a driver, an attribute, and so on, that might become a dimension in a cube cannot have the reserved names All or None.
- Names are case insensitive. For example, the name My Model is the same as my model and mY mODEL.

In addition to the general naming conventions, there are more restrictive naming conventions for the following items.

Attribute naming conventions

In addition to the general naming conventions, attribute names must conform to these rules:

- Attribute names must be unique within a parent.
- Attribute names may contain up to 64 alphanumeric characters. However, attribute names longer than 50 characters are truncated to 50 characters when a cube is generated for Microsoft Analysis Services. SAS OLAP allows all 64 characters for attribute names.
- An attribute name cannot be the name of a numeric property.
- Attribute names may contain the following characters, even though these characters are not valid in cubes:

  ```
  . , ; ' ` : ? * & % $ ! - + = ( ) [ ] { } / /
  ```

  Each of these characters is replaced with an underscore (_) when a cube is generated.

For additional considerations concerning stage attributes, see "Stage Attributes" in the *SAS Activity-Based Management User's Guide*, available from the Help menu.

Dimension naming conventions

In addition to the general naming conventions, dimension names must conform to these rules:

- Dimension names may contain up to 64 alphanumeric characters. However, dimension names longer than 32 characters are truncated to 32 characters when a cube is generated for Microsoft Analysis Services. SAS OLAP allows all 64 characters for dimension names.

- Dimension names must be unique within all dimensions and dimension attributes.
- Dimension names must be unique within a parent.
- Dimension names may contain these characters, even though these characters are not valid in cubes:

  ```
  . , ; ' ` : ? * & % $ ! - + = ( ) [ ] { } / /
  ```

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Dimension level naming conventions

In addition to the general naming conventions, dimension level names must conform to these rules:

- Dimension level names may contain up to 64 alphanumeric characters. However, dimension level names longer than 50 characters are truncated to 50 characters when a cube is generated for Microsoft Analysis Services. SAS OLAP allows all 64 characters for dimension level names.
- Dimension level names must begin with an alphabetic character.
- Dimension level names cannot contain these characters:

  ```
  / / |
  ```

- Dimension level names may contain these characters, even though these characters are not valid in cubes:

  ```
  . [ ]
  ```

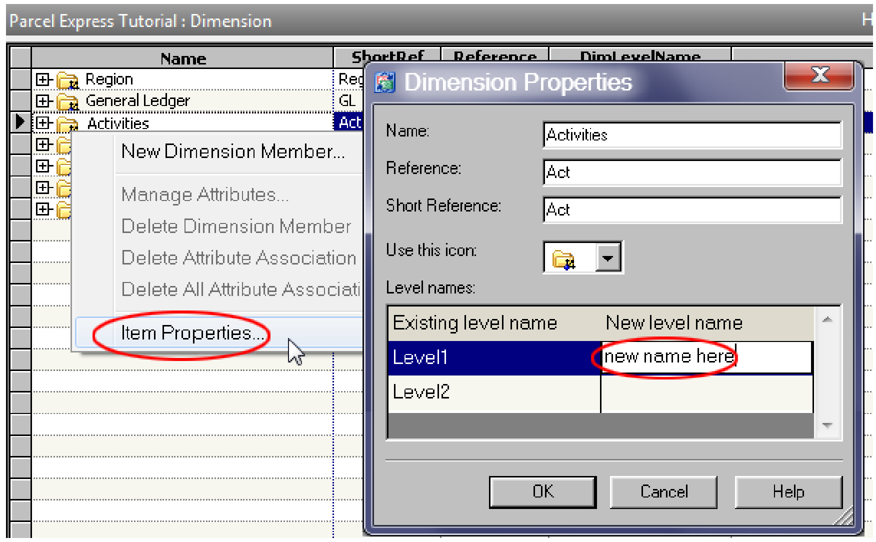
Each of these characters will be replaced with an underscore (_) when a cube is generated.

- Note: Because of the mechanism used by SAS Activity-Based Management to store dimension level names, some user-specified names will cause conflicts with the underlying database (regardless of whether Microsoft SQL Server or Oracle is used). These conflicts will appear as obscure error messages when calculating a model. Dimension level names that will cause conflicts are reserved words in the Microsoft SQL query language. Some of the more common reserved words are: level, group, function, drop, and join. For example, using the name LeVeL will cause errors. To avoid dimension level name conflicts, add a descriptive prefix or suffix.

To change the name of a dimension level:

1. Open a model.
2. Select **Model** ⇨ **Dimensions**.
3. Right-click a dimension and select **Item Properties**.

   The Dimension Properties window opens.
4. Type a new name, and then click **OK**.



Dimension member naming conventions

In addition to the general naming conventions, dimension member names must conform to these rules:

- Dimension member names may contain up to 64 alphanumeric characters. However, dimension member names longer than 50 characters are truncated to 50 characters when a cube is generated for Microsoft Analysis Services. SAS OLAP allows all 64 characters for dimension member names.
- Dimension member names may contain the following characters, even though these characters are not valid in cubes:

  `. [ ]`

Dimension member names must be unique within a parent.

Driver naming conventions

In addition to the general naming conventions, driver names must conform to these rules:

- Driver names must be unique within all drivers.
- Driver names may contain up to 64 alphanumeric characters. However, driver names that are longer than 50 characters are truncated to 50 characters when a cube is generated.
- Driver names may contain these characters, even though these characters are not valid in cubes:

  `. [ ]`

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Entered cost element naming conventions

In addition to the general naming conventions, entered cost element names must conform to these rules:

- Entered cost element names may contain up to 64 alphanumeric characters.
- Entered cost element names must be unique within the same account in the same period/scenario association.
- Entered cost element names may contain these characters, even though these characters are not valid in cubes:

  . [ ]

Each of these characters will be replaced with an underscore (_) when a cube is generated.

External unit naming conventions

In addition to the general naming conventions, external unit names must conform to these rules:

- External unit references must be unique within all external units.
- External unit names may contain up to 64 alphanumeric characters.

Module naming conventions

When renaming modules, the name must conform to these rules::

- Module names may contain up to 64 alphanumeric characters.
- Module names can contain the following characters: alphanumeric, underscores, embedded blanks.

Period naming conventions

In addition to the general naming conventions, period names must conform to these rules:

- Period names must be unique within all periods.
- Period names may contain up to 64 alphanumeric characters. However, period names that are longer than 50 characters are truncated to 50 characters when a cube is generated.
- Period names may contain these characters, even though these characters are not valid in cubes:

  . [ ]

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Period level naming conventions

In addition to the general naming conventions, period level names must conform to these rules:

- Period level names must be unique within all period levels.
- Period level names may contain up to 64 alphanumeric characters. However, period level names that are longer than 50 characters are truncated to 50 characters when a cube is generated.

- Period level names may contain these characters, even though these characters are not valid in cubes:

  ```
  . [ ]
  ```

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Scenario naming conventions

In addition to the general naming conventions, scenario names must conform to these rules:

- Scenario names must be unique within all scenarios.
- Scenario names may contain up to 64 alphanumeric characters. However, scenario names that are longer than 50 characters are truncated to 50 characters when a cube is generated.
- Scenario names may contain these characters, even though these characters are not valid in cubes:

  ```
  . [ ]
  ```

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Scenario level naming conventions

In addition to the general naming conventions, scenario level names must conform to these rules:

- Scenario level names must be unique within all scenario levels.
- Scenario level names may contain up to 64 alphanumeric characters. However, scenario level names that are longer than 50 characters are truncated to 50 characters when a cube is generated.
- Scenario level names may contain these characters, even though these characters are not valid in cubes:

  ```
  [ ]
  ```

Each of these characters will be replaced with an underscore (_) when a cube is generated.

Stage attribute naming conventions

In addition to the general naming conventions and Attribute naming conventions, stage attribute names must conform to these rules:

- Stage names must begin with an alphabetic character (letter).
- Stage names may contain the following characters, but they are not valid in cubes and cause cube generation to fail:

  ```
  . , ; ' ` : ? * & % $ ! - + = ( ) [ ] { } / /
  ```

- You can change stage names as long as they retain their order when sorted. If the sort order changes them, you will need to regenerate the fact tables for all period/ scenarios in the model.

Workspace item naming conventions

In addition to the general naming conventions, workspace item names must conform to these rules:

- Workspace item names cannot contain these characters:

  `/ / or |`

- Workspace item names must be unique within a folder.
- Workspace item names may contain up to 64 alphanumeric characters.

Reference Conventions

Account reference conventions

- Account references must be unique within a module for all period/scenario associations.

For additional considerations concerning stage attributes, see “Stage Attributes” in the *SAS Activity-Based Management User’s Guide*, available from the Help menu.

Attribute reference conventions

- Attribute references must be unique within all attributes.

External unit reference conventions

- External unit references must be unique within all external units.

Dimension reference conventions

- Dimension references must be unique within all dimensions and dimension attributes.

Dimension member reference conventions and dimension attribute reference conventions

- Dimension member references and dimension attribute references must be unique within a dimension.

Period reference conventions

- Period references must be unique within all periods.

Scenario reference conventions

- Scenario references must be unique within all scenarios.

Entered cost element reference conventions

- Entered cost element references must be unique within a module for all period/ scenario associations.

Account table

This table specifies the dimension intersections in the Resource module, the Activity module, and the Cost Object module. The combination of the columns DimRef<n> and DimMemberRef<n> for each valid dimension is the dimension signature. Do not include dimension signatures for the External Units module in this table.

This table is required when creating a new model. It is not required for updating an existing model.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
ModuleType	AlphaNumeric	64	key1, key2	Required	Required	Required
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required
DimSignature	DimensionSignature		key1	Required	Required	Required
DimRef	AlphaNumeric	64		Required	Required	N/A
DimName	AlphaNumeric	64		N/A	Optional	N/A
DimMemberRef	AlphaNumeric	64		Required	Required	N/A
DimMemberName	AlphaNumeric	64		N/A	Optional	N/A
DimLevel	Integer			N/A	Optional	N/A
DimLevelName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key2	Required	Required	Required
DriverName	AlphaNumeric	64		Optional	Required	N/A
Name	AlphaNumeric	64		Optional	Required	N/A
OutputQuantityUE	Float			Optional	Required	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
PeriodicNote	Memo			Optional	Required	N/A
PublishName	AlphaNumeric			Optional	Required	N/A
Revenue	Float			Optional	Required	N/A
SoldQuantity	Float			Optional	Required	N/A
TDQUE	Float			Optional	Required	N/A
UnitOfMeasure	AlphaNumeric	64		Optional	Required	N/A
ValueAttributes	Memo/Float			N/A	Optional	N/A
AllocatedCost	Float			N/A	Optional	N/A
AssignedCost	Float			N/A	Optional	N/A
AssignedIdleCost	Float			N/A	Optional	N/A
AssignedIdleQuantity	Float			N/A	Optional	N/A
AssignedNonReciprocalCost	Float			N/A	Optional	N/A
AssignedReciprocalCost	Float			N/A	Optional	N/A
Cost	Float			N/A	Optional	N/A
DrivableCost	Float			N/A	Optional	N/A
DrivenCost	Float			N/A	Optional	N/A
DrivenQuantity	Float			N/A	Optional	N/A
DriverRate	Float			N/A	Optional	N/A
EnteredCost	Float			N/A	Optional	N/A
IdleCost	Float			N/A	Optional	N/A
IdlePercentage	Float			N/A	Optional	N/A
IdleQuantity	Float			N/A	Optional	N/A
OutputQuantity	Float			N/A	Optional	N/A
Profit	Float			N/A	Optional	N/A
ReceivedAllocatedCost	Float			N/A	Optional	N/A
ReceivedAssignmentCost	Float			N/A	Optional	N/A
ReceivedBOCCost	Float			N/A	Optional	N/A
ReceivedCost	Float			N/A	Optional	N/A
ReceivedDrivenCost	Float			N/A	Optional	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ReceivedNonReciprocalCost	Float			N/A	Optional	N/A
ReceivedReciprocalCost	Float			N/A	Optional	N/A
TDQ	Float			N/A	Optional	N/A
TDQBasic	Float			N/A	Optional	N/A
TDQCalculated	Float			N/A	Optional	N/A
UnassignedCost	Float			N/A	Optional	N/A
UnassignedQuantity	Float			N/A	Optional	N/A
UnitCost	Float			N/A	Optional	N/A
UnitProfit	Float			N/A	Optional	N/A
UnitRevenue	Float			N/A	Optional	N/A
UsedCost	Float			N/A	Optional	N/A
UsedQuantity	Float			N/A	Optional	N/A
IsBehavior	Yes/No			Optional	Required	

Notes:

Field Name	Note
ModuleType	Resource | "Activity" | "CostObject"
DimSignature	There is no column of this name. Dimension signature contains dimension members involved in the account creation. So DimRef(n) and DimmemberRef(n) are mainly required for dimension signature. DimName, DimMemberName, Dimlevel, DimLevelname are optional
Reference	Account must exist to use Reference as the key, otherwise there is no way to hang a dimension reference to a dimension signature. Value must be unique within module. Default value will be generated if blank or not supplied.
Name	Default generated if blank or not supplied
ValueAttributes	There no column of this name. If user want to see the associated attributes (text or numeric) with the account then user can defines these attributes. These attributes will come as column in the staging table.
IsBehavior	Used to Link to Profitability Management

Assignment table

This table specifies assignments, including assignments from the External Units module. The accounts in an assignment can be specified with the columns SourceReference and DestinationReference or with the columns SourceDimRef<n> and SourceDimMemberRef<n>, and DestinationDimRef<n> and DestinationDimMemberRef<n>.

This table is required when creating a new model. It is not required for updating an existing model.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required
SourceModuleType	AlphaNumeric	64	key1, key2	Required	Required	Required
SourceDimSignature	DimensionSignature		key1	Required1	Required	Required
SourceReference	AlphaNumeric	64	key2	Required2	Required	Required
SourceCost	Float				Optional	
DestinationModuleType	AlphaNumeric	64	key1, key2	Required	Required	Required
DestinationDimSignature	DimensionSignature		key1	Required1	Required	Required
DestinationReference	AlphaNumeric	64	key2	Required2	Required	Required
DriverName	AlphaNumeric	64		Optional	Optional	N/A
AllocatedCost	Float			Optional	Required	N/A
DriverQuantityFixed	Float			Optional	Required	N/A
DriverQuantityVariable	Float			Optional	Required	N/A
DriverWeightFixed	Float (>=1)			Optional	Required	N/A
DriverWeightVariable	Float (>=1)			Optional	Required	N/A
Cost	Float			N/A	Optional	N/A
DriverQuantityBasic	Float			N/A	Optional	N/A
DriverQuantityCalculated	Float			N/A	Optional	N/A
IdleDriverQuantity	Float			N/A	Optional	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
IdleDriverQuantityUE	Float (>=1)			Optional	Required	N/A
IdleCost	Float			N/A	Optional	N/A

Notes:

Field Name	Note
SourceModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject"
SourceDimSignature	"Source" is the prefix for all the DimSignature fields. Source account will be created if the AutoCreateAccount="1" attribute is included in the Config XML.
DestinationModuleType	"Resource" | "Activity" | "CostObject"
DestinationDimSignature	"Destination" is the prefix for all the DimSignature fields. Destination account will be created if the AutoCreateAccount=1 attribute is included in the Config XML.
DriverName	Source account's driver will be changed to this driver if it doesn't match already. Ignored if null or empty string. If AutoCreateAccount="1" attribute is included in the Config XML then this will be created as a Basic driver.
DriverWeightFixed	>=0 Ignored unless source account uses Weighted Driver. (Value stored in FixedWeight field)
DriverWeightVariable	same as DriverWeightFixed

AssignmentNonUnique table

This table specifies the driver quantities on destination accounts for all drivers where the driver quantity type is nonunique. The driver quantity on a destination account can be specified with the column DestinationReference or with the columns DimRef<n> and DimMemberRef<n>.

This table is not required when importing data because the unique driver quantities can be specified in the Assignment table. If this table is imported, all columns must exist, but all columns do not need to contain data.

This table is not required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required
DestinationModuleType	AlphaNumeric	64	key1, key2	Required	Required	Required

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
DestinationDimSignature	DimensionSignature		key1	Required1	Required	Required
DestinationReference	AlphaNumeric	64	key2	Required2	Required	Required
DriverQuantityFixed	Float			Optional	Optional	N/A
DriverQuantityVariable	Float			Optional	Optional	N/A
DriverWeightFixed	Float			Optional	Optional	N/A
DriverWeightVariable	Float			Optional	Optional	N/A
SourceDimSignature	Signature		key1	Optional	Optional	Required
DriverName	AlphaNumeric	64	key1, key2	Required	Required	N/A

Notes:

Field Name	Note
DestinationModuleType	"Resource" | "Activity" | "CostObject"
DriverWeightFixed	>=0 Ignored unless source account uses Weighted Driver. (Value stored in FixedWeight field)
DriverWeightVariable	same as DriverWeightFixed

CurrencyRate table

This table specifies currency exchange rates for each period.

If multiple currencies are not required in the model, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key	Required	Required	Required
Scenario	AlphaNumeric	64	key	Required	Required	Required
CurrencyFrom	AlphaNumeric	64		Required	Required	Required
CurrencyTo	AlphaNumeric	64		Required	Required	Required
Rate	Float			Optional	Required	N/A

Notes:

Field Name	Note
CurrencyFrom	Currency code defined in CurrencyDefinition
CurrencyTo	Currency code defined in CurrencyDefinition
Rate	CurrencyTo = CurrencyFrom * Rate

Dimension table

This table specifies the dimensions in the model. This table must include the dimensions that are required for building the structure of the Resource module, the Activity module, the Cost Object module, and the External Units module. This table includes the dimension attributes if dimension attributes are used in the model. Do not include numeric attributes, text attributes, and Boolean attributes.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64			Optional	
Reference	AlphaNumeric	64	key	Required	Required	
Name	AlphaNumeric	64		Required	Required	
ShortReference	AlphaNumeric	18		Optional	Required	

DimMemberDimAttrAssociation

This table stores attributes that are attached to dimension members.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
Period	AlphaNumeric	64	key	Required	Required	
Scenario	AlphaNumeric	64	key	Required	Required	
DimMemberRef	AlphaNumeric	64	key	Required	Required	
DimRef	AlphaNumeric	64	key	Required	Required	

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
AttributeDimRef	AlphaNumeric	64	key	Required	Required	
AttributeDimMemberRef	AlphaNumeric	64	key	Required	Required	

DimMemberValueAttrAssociation

This table stores the value of attributes that are attached to dimension members.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
Period	AlphaNumeric	64	key	Required	Required	
Scenario	AlphaNumeric	64	key	Required	Required	
DimMemberRef	AlphaNumeric	64	key	Required	Required	
DimRef	AlphaNumeric	64	key	Required	Required	
AttributeReference	AlphaNumeric	64	key	Required	Required	
NumericValue	Float			Optional	Optional	
StringValue	Memo			Optional	Optional	

DimensionAttributeAssociation table

This table specifies the accounts that are associated with dimension attributes. The association can be specified with the column ItemReference or with the columns DimRef<n> and DimMemberRef<n>.

This table is not required when importing data. If this table is imported, all columns must exist, but all columns do not need to contain data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ItemModuleType	AlphaNumeric		key1, key2	Required	Required	Required
ItemDimSignature	DimensionSignature		key1	Required1	Required	Required
ItemReference	AlphaNumeric		key2	Required2	Required	N/A
AttributeDimRef	AlphaNumeric		key1, key2	Required	Required	Required
AttributeDimName	AlphaNumeric			N/A	Optional	N/A
AttributeDimMemberRef	AlphaNumeric		key1, key2	Required	Required	Required
AttributeDimMemberName	AlphaNumeric			N/A	Optional	N/A
IsSystem	Number			Optional	Optional	N/A

Notes:

Field Name	Note
ItemModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject"
AttributeDimRef	Non-structural dimensions

DimensionLevel table

This table specifies the name of each level in a dimension hierarchy.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
DimRef	AlphaNumeric	64	key	Required	Required	Required
LevelNo	Integer (>=1)		key	Required	Required	Required
Name	AlphaNumeric	64		Required	Required	N/A

DimensionMember table

This table specifies the hierarchy for each dimension.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
DimRef	AlphaNumeric	64	key	Required	Required	Required
DimName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key	Required	Required	Required
Name	AlphaNumeric	64		Required	Required	N/A
ParentReference	AlphaNumeric	64		Required	Required	N/A
DimLevel	Integer			Optional	Optional	N/A
DimLevelName	AlphaNumeric	64		Optional	Optional	N/A
DisplayOrder	Float			Optional	Optional	N/A

Notes:

Field Name	Note
ParentReference	Must be an existing DimensionMember Refererence in the model or in this table. Root element can be identified as one of: empty string, Null, "All".
DimLevel	Hierarchy level override. Defaults to ParentLevel+1. Ignored if null or <=0.
DimLevelName	Hierarchy level override. Ignored if null. This takes precedant over DimLevel if this is non-null and DimLevel is also included.
DisplayOrder	Determines the order in which dimension members are displayed in the user interface. It is suggested that you start with 10 and increment by 10 (e.g., 10, 20, 30, ...) so that you can interpose later rows (e.g., 10, 20, 30, 15, 40). The numbers used need only be unique within a dimension.

DimensionOrder table

This table specifies the order of the dimensions in the Resource module, the Activity module, the Cost Object module, and the External Units module. The records in this table must be sorted according to module type and sequence number.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
ModuleType	AlphaNumeric	64	key	Required	Required	N/A
SequenceNumber	(Integer>=1)		key	Required	Required	N/A
DimRef	AlphaNumeric	64		Required	Required	N/A
DimName	AlphaNumeric	64		N/A	Optional	N/A

Notes:

Field Name	Note
ModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject" | "Profitability"
SequenceNumber	Order of dimension in module structure

Driver table

This table specifies the drivers.

This table is required when creating a new model. It is not required for updating an existing model. When importing, if the account does not exist, the account is added.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Name	AlphaNumeric	64	key	Required	Required	N/A
DriverType	AlphaNumeric	64		Required	Required	N/A
UniqueDriverQuantities	Yes/No			Optional	Optional	N/A
Formula	Memo	2048		Optional	Optional	N/A
UseFixedQuantities	Yes/No			Optional	Optional	N/A
UseVariableQuantities	Yes/No			Optional	Optional	N/A
UseWeightedQuantities	Yes/No			Optional	Optional	N/A
SequenceNumber	Integer			Optional	Optional	N/A
FixedDriverQuantityOverride	AlphaNumeric	64		Optional	Optional	N/A
VariableDriverQuantityOverride	AlphaNumeric	64		Optional	Optional	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
IdleFlowMethod	AlphaNumeric	64		Optional	Optional	N/A
UserEnteredCostAllocation	Yes/No			Optional	Required	N/A
RuleFormula	AlphaNumeric	2048		Optional	Optional	N/A
UseRuleFormula	Yes/No			Optional	Optional	N/A

Notes:

Field Name	Note
DriverType	"Percentage" | "Evenly Assigned" | "Basic" | "Weighted" | "Bill of Cost" | "Sales Volume" | "Calculated" | "Center". Default="Basic".
UniqueDriverQuantities	Default=Yes (for DriverType<>"Center")
Formula	Default=""
UseFixedQuantities	Default based on DriverType
UseVariableQuantities	Default based on DriverType
UseWeightedQuantities	Default based on DriverType
SequenceNumber	Default=1
FixedDriverQuantityOverride	Property API name
VariableDriverQuantityOverride	Property API name
IdleFlowMethod	"DontAssign" | "EvenlyAssign" | "UseDriverQuantities" | "UserEntered" | "UserProportion", Default="DontAssign"
UserEnteredCostAllocation	Default=No

EnteredCostElement table

This table specifies the entered cost elements and the account that is associated with each entered cost element. The account that is associated with an entered cost element can be specified with the column AccountReference or with the columns DimRef<n> and DimMemberRef<n>.

This table is required when creating a new model. It is not required for updating an existing model.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key	Required	Required	Required
Scenario	AlphaNumeric	64	key	Required	Required	Required
ModuleType	AlphaNumeric	64	key	Required	Required	Required
AccountReference	AlphaNumeric	64		Required1	Optional	Required
AccountDimSignature	DimensionSignature			Required2	Required	Required
Reference	AlphaNumeric	64	key	Required	Required	Required
Name	AlphaNumeric	64		Required	Required	N/A
EnteredCost	Float			Optional	Optional	N/A

Notes:

Field Name	Note
ModuleType	"Resource" | "Activity" | "CostObject"
AccountDimSignature	There is no column of this name. Dimension signature contains dimension members involved in the account creation. So DimRef(n) and DimmemberRef(n) are mainly required for dimension signature. DimName, DimMemberName, Dimlevel, DimLevelname are optional
EnteredCost	Default=0.0

ExternalUnit table

This table specifies dimension intersections for the External Units module. The combination of the columns DimRef<n> and DimMemberRef<n> for each valid dimension is the dimension signature.

If the model does not use external units, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
DimSignature	DimensionSignature		key1	Required	Required	Required
DimRef	AlphaNumeric	64		N/A	Required	N/A
DimName	AlphaNumeric	64		N/A	Optional	N/A
DimMemberRef	AlphaNumeric	64		N/A	Required	N/A
DimMemberName	AlphaNumeric	64		N/A	Optional	N/A
DimLevel	Integer			N/A	Optional	N/A
DimLevelName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key2	Required	Required	Required
DriverName	AlphaNumeric	64		Optional	Required	N/A
Name	AlphaNumeric	64		Optional	Required	N/A
PeriodicNote	Memo			Optional	Required	N/A
PublishName	AlphaNumeric	64		Optional	Required	N/A
UnitCostEntered	Float			Optional	Required	N/A
UnitOfMeasure	AlphaNumeric			Optional	Required	N/A
AllocatedCost	Float			N/A	Optional	N/A
AssignedCost	Float			N/A	Optional	N/A
Cost	Float			N/A	Optional	N/A
DrivableCost	Float			N/A	Optional	N/A
DrivenCost	Float			N/A	Optional	N/A
DrivenQuantity	Float			N/A	Optional	N/A
DriverRate	Float			N/A	Optional	N/A
OutputQuantity	Float			N/A	Optional	N/A
TDQ	Float			N/A	Optional	N/A
TDQBasic	Float			N/A	Optional	N/A
TDQCalculated	Float			N/A	Optional	N/A
UsedCost	Float			N/A	Optional	N/A
UsedQuantity	Float			N/A	Optional	N/A
ValueAttributes	Memo/Float			N/A	Optional	N/A

Notes:

Field Name	Note
DimSignature	There is no column of this name. Dimension signature contains dimension members involved in the account creation. So DimRef and DimmemberRef are mainly required for dimension signature. DimName, DimMemberName, Dimlevel, DimLevelname are optional. Only a single Dim/DimMember pair can be specified.
Reference	ExternalUnit must exist to use Reference as the key, otherwise there is no way to hang a dimension reference to a dimension signature. Value must be unique across all External Units. Default value will be generated if blank or not supplied.
DriverName	Only BOC drivers and Calculated drivers are allowed
Name	Default generated if blank or not supplied
UnitCostEntered	Default=0.0
ValueAttributes	There no column of this name. If user want to see the associated attributes (text or numeric) with the account then user can defines these attributes. These attributes will come as column in the staging table.

Model table

Overview

This table specifies the model name, the description, and the default values for the model. SAS Activity-Based Management uses this table for export only.

This table is not required for importing data because you must specify the model name when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
Name	AlphaNumeric	64		N/A	Optional	N/A
Description	AlphaNumeric	64		Optional	Optional	N/A
DefaultOutputQuantity	Float			Optional	Optional	N/A
BaseCurrency	AlphaNumeric	64		Optional	Optional	N/A
AdvancedOptions	Memo			Optional	Optional	N/A
Reference	AlphaNumeric	8		Optional	Optional	N/A
ColumnLayout	AlphaNumeric	256		Optional	Optional	N/A

Notes:

Field Name	Note
Name	Model Name and the full path name in the workspace
DefaultOutputQuantity	=0 or =1
BaseCurrency	Currency code defined in CurrencyDefinition. Can only be defined when creating a new model.
AdvancedOptions	Information about default drivers for modules, numeric attributes and stages include in the model. It is stored in XML format

AdvancedOptions Field

Overview

When you import data into a model, you can write XML that specifies the advanced options of the model's properties. You specify this XML in the AdvancedOptions field of the Model table.

Sample XML for the Advanced Options Field

The following example illustrates the use of the XML AdvancedOptions field:

```
<AdvancedOptions>
    <Modules>
        <Module Type="ExternalUnit" DefaultDriver="Bill of Cost" />
        <Module Type="Resource" DefaultDriver="Evenly Assigned" />
        <Module Type="Activity" DefaultDriver="Evenly Assigned" />
        <Module Type="CostObject" DefaultDriver="Evenly Assigned" />
    </Modules>
    <Cube Type="Common">
        <NumericAttributes>
            <NumericAttribute Reference="FI" />
        </NumericAttributes>
    </Cube>
    <Cube Type="DetailedContribution">
        <StageDefinition Type="Attribute" IncludeExternalUnitAsStage="true">
            <Stage Reference="S1R" CostFlow="In" />
            <Stage Reference="S2PA" CostFlow="In" />
            <Stage Reference="S3I" CostFlow="In" />
            <Stage Reference="S4S" CostFlow="In" />
            <Stage Reference="S5D" CostFlow="In" />
            <Stage Reference="S6F" CostFlow="In" />
            <Stage Reference="S7f" CostFlow="In" />
        </StageDefinition>
    </Cube>
</AdvancedOptions>
```

The following is an example of the StageDefinition element with a Type of Module:

```
<StageDefinition Type="Module">
    <Stage Reference="ExternalUnit" CostFlow="Out" />
    <Stage Reference="Resource" CostFlow="Out" />
    <Stage Reference="Activity" CostFlow="Out" />
```

```
    <Stage Reference="CostObject" CostFlow="In" />
</StageDefinition>
```

XML AdvancedOptions structure

The following structure shows the elements used in the XML AdvancedOptions field and their relationships to each other. All of the elements are required.

```
<AdvancedOptions>
    <Modules>
        <Module attributes/>
    </Modules>
    <Cube Type="Common">
        <NumericAttributes>
            <NumericAttribute attributes/>
        </NumericAttributes>
    </Cube>
    <Cube Type="DetailedContribution">
        <StageDefinition attributes>
            <Stage attributes/>
        </StageDefinition>
    </Cube>
</AdvancedOptions>
```

AdvancedOptions element (required)

AdvancedOptions is the root element. This element has no attributes.

Modules element (required)

This element contains elements that specify information for each module in a model. This element has no attributes.

Module element (required)

This element contains attributes that specify the type of module and its default driver.

Attribute	Required	Values	Description
Type	Required (one Type attribute for each module)	"Resource" "Activity" "CostObject" "ExternalUnit"	Modules in a model
DefaultDriver	Required	String	Name of the default driver

Cube Type=Common element (required)

This element contains an attribute that specifies information that is common to all cubes.

NumericAttributes element (required)

This element contains elements that specify the numeric attributes that are included in cubes. This element has no attributes.

NumericAttribute element (required)

This element contains attributes that specify information about the numeric attributes that are included in cubes.

Attribute	Required	Values	Description
Reference	Required	String	Reference of the numeric attribute

Cube Type=DetailedContribution element (required)

This element contains an attribute that specifies information about the Multi-stage Contributions cube.

StageDefinition element (required)

This element contains attributes that specify how stages are defined.

Attribute	Required	Values	Description
Type	Required	"Module" "Attribute"	Method used to define stages
IncludeExternalUnitAsStage	Required when Type is "Attribute"	"true" "false"	Indicates whether external units are included in the stages

Stage element (required)

This element contains attributes that specify how stages are defined.

Attribute	Required	Values	Description
Reference	Required	String	Reference of each stage When the StageDefinition Type is "Module": "Resource" "Activity" "CostObject" "ExternalUnit" When the StageDefinition Type is "Attribute": Reference of the attribute
CostFlow	Required	"In" "Out"	Type of cost to use for a stage

Related Topics:

- Example XML AdvancedOptions field

MultiStageContribution table

This table specifies how cost flows through each predefined stage and what accounts in those predefined stages participate in the flow.

SAS Activity-Based Management uses this table for export only.

This table is not required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1	N/A	Optional	N/A
Scenario	AlphaNumeric	64	key1	N/A	Optional	N/A
Cost	Float			N/A	Optional	N/A
Revenue	Float			N/A	Optional	N/A
OutputQuantity	Float			N/A	Optional	N/A
SoldQuantity	Float			N/A	Optional	N/A
TDQUE	Float			N/A	Optional	N/A
OutputQuantityUE	Float			N/A	Optional	N/A
StageModuleTypeReference1	AlphaNumeric	64	key2	N/A	Optional	N/A
StageDimSignature1	Signature		key1	N/A	Optional	N/A
StageDimSignature(n)	Signature		key1	N/A	Optional	N/A
StageAccountReference1	AlphaNumeric	64	key2	N/A	Optional	N/A
StageAccountReference(n)	AlphaNumeric	64	key2	N/A	Optional	N/A
Value Attributtes	Float/Memo			N/A	Optional	N/A

Notes:

Field Name	Note
Value Attributtes	There no column of this name. If user want to see the associated attributes (text or numeric) with the account then user can defines these attributes. These attributes will come as column in the staging table.

PerformanceMeasure table

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	
Period	AlphaNumeric	64	key	Required	Required	Required
Scenario	AlphaNumeric	64	key	Required	Required	Required
ItemDimSignature	Signature		key1	Required1	Required	Required
ItemModuleType	AlphaNumeric	64		Required	Required	
ItemReference	AlphaNumeric	64	key2	Required2	Required	Required
MeasureType	AlphaNumeric	64	key	Required	Required	Required
MeasureReference	AlphaNumeric	64	key	Required	Required	

Period table

This table specifies the period names, the descriptions, and the start dates and the end dates.

If the required periods already exist on the SAS Activity-Based Management server, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key	Required	Required	Required
ParentReference	AlphaNumeric	64		Required	Required	N/A
Name	AlphaNumeric	64		Required	Required	N/A
StartDate	Date			Required	Required	N/A
EndDate	Date			Required	Required	N/A
Description	Memo			Optional	Optional	N/A

Notes:

Field Name	Note
ParentReference	Root element can be identified as one of: empty string, Null, "All"
StartDate	May need to change if parent is in different date range
EndDate	May need to change if parent is in different date range

PeriodLevel table

This table specifies the level names for period hierarchies.

If the required period levels already exist on the SAS Activity-Based Management server, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
LevelNo	Integer (>=1)		key	Required	Required	Required
Name	AlphaNumeric	64		Required	Required	N/A

Notes:

ResourceContribution table

This table specifies the cost contribution relationship and the percentage from each starting account (typically a resource account) to each ending account (typically a cost object account).

SAS Activity-Based Management uses this table for export only.

This table is not required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1	N/A	Optional	N/A
Scenario	AlphaNumeric	64	key1	N/A	Optional	N/A
SourceModuleType	AlphaNumeric	64	key1, key2	N/A	Optional	N/A
SourceDimSignature	DimensionSignature		key1	N/A	Optional	N/A
SourceReference	AlphaNumeric	64	key2	N/A	Optional	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
DestinationModuleType	AlphaNumeric	64	key1, key2	N/A	Optional	N/A
DestinationDimSignature	ExportDimensionSignature		key1	N/A	Optional	N/A
DestinationReference	AlphaNumeric	64	key2	N/A	Optional	N/A
DestinationCost	Float			N/A	Optional	N/A
DestinationOutputQty	Float			N/A	Optional	N/A
DestinationSoldQty	Float			N/A	Optional	N/A
ContribCost	Float			N/A	Optional	N/A
ContribPcnt	Float			N/A	Optional	N/A

Notes:

Field Name	Note
SourceModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject"
SourceDimSignature	"Source" is the prefix for all the DimSignature fields.
DestinationModuleType	"Resource" | "Activity" | "CostObject"
DestinationDimSignature	"Destination" is the prefix for all the DimSignature fields.

RollupAccount table

This table specifies the rollup accounts that are at a higher level than the leaf accounts. The rollup accounts' numeric properties are the rollup of the corresponding properties of the descendant leaf accounts, based on the dimensional hierarchy.

SAS Activity-Based Management uses this table for export only.

This table is not required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
ModuleType	AlphaNumeric	64	key	N/A	Optional	N/A
Period	AlphaNumeric	64	key	N/A	Optional	N/A
Scenario	AlphaNumeric	64	key	N/A	Optional	N/A
DimSignature	DimensionSignature		key	N/A	Optional	N/A

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
DimRef	AlphaNumeric	64		N/A	Optional	N/A
DimName	AlphaNumeric	64		N/A	Optional	N/A
DimMemberRef	AlphaNumeric	64		N/A	Optional	N/A
DimMemberName	AlphaNumeric	64		N/A	Optional	N/A
DimLevel	Integer			N/A	Optional	N/A
DimLevelName	AlphaNumeric	64		N/A	Optional	N/A
AllocatedCost	Float			N/A	Optional	N/A
AssignedCost	Float			N/A	Optional	N/A
AssignedNonReciprocalCost	Float			N/A	Optional	N/A
AssignedReciprocalCost	Float			N/A	Optional	N/A
Cost	Float			N/A	Optional	N/A
DrivenCost	Float			N/A	Optional	N/A
EnteredCost	Float			N/A	Optional	N/A
IdleCost	Float			N/A	Optional	N/A
IdlePercentage	Float			N/A	Optional	N/A
Profit	Float			N/A	Optional	N/A
ReceivedAllocatedCost	Float			N/A	Optional	N/A
ReceivedAssignmentCost	Float			N/A	Optional	N/A
ReceivedBOCCost	Float			N/A	Optional	N/A
ReceivedCost	Float			N/A	Optional	N/A
ReceivedDrivenCost	Float			N/A	Optional	N/A
ReceivedNonReciprocalCost	Float			N/A	Optional	N/A
ReceivedReciprocalCost	Float			N/A	Optional	N/A
Revenue	Float			N/A	Optional	N/A
UnassignedCost	Float			N/A	Optional	N/A

Notes:

Field Name	Note
ModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject"

Field Name	Note
DimSignature	There is no column of this name. Dimension signature contains dimension members involved in the account creation. So DimRef and DimmemberRef are mainly required for dimension signature. DimName, DimMemberName, Dimlevel, DimLevelname are optional. Only a single Dim/DimMember pair can be specified.

Scenario table

This table specifies the scenario names and the descriptions.

If the required scenarios already exist on the SAS Activity-Based Management server, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key	Required	Required	Required
ParentReference	AlphaNumeric	64		Required	Required	N/A
Name	AlphaNumeric	64		Required	Required	N/A
Description	Memo			Optional	Optional	N/A

Notes:

Field Name	Note
ParentReference	Root element can be identified as one of: empty string, Null, "All"

ScenarioLevel table

This table specifies the level names for scenario hierarchies.

If the required scenario levels already exist on the SAS Activity-Based Management server, then this table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
LevelNo	Integer (>=1)		key	Required	Required	Required

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
Name	AlphaNumeric	64		Required	Required	N/A

ValueAttribute table

This table specifies the attribute hierarchy for numeric attributes , text attributes , and Boolean attributes.

This table is not required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Reference	AlphaNumeric	64	key	Required	Required	Required
Name	AlphaNumeric	64		Required	Required	N/A
ParentReference	AlphaNumeric	64		Required	Required	N/A
Type	AlphaNumeric	64		Required	Required	N/A
UnitOfMeasure	AlphaNumeric	64		Optional	Optional	N/A

Notes:

Field Name	Note
Type	"Tagged" | "Numeric" | "Text" | "Center"
UnitOfMeasure	Only applicable for "Numeric"

ValueAttributeAssociation table

This table specifies the accounts that are associated with numeric attributes, text attributes, and Boolean attributes. The association can be specified with the column ItemReference or with the columns DimRef<n> and DimMemberRef<n>.

This table is not required when importing data. If this table is imported, all columns must exist, but all columns do not need to contain data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key1, key2	Required	Required	Required
Scenario	AlphaNumeric	64	key1, key2	Required	Required	Required
ItemModuleType	AlphaNumeric	64	key1, key2	Required	Required	Required
ItemDimSignature	DimensionSignature		key1	Required1	Required	Required
ItemReference	AlphaNumeric	64	key2	Required2	Optional	Required
AttributeReference	AlphaNumeric	64	key1, key2	Required	Required	Required
NumericValue	Float			Optional	Optional	N/A
StringValue	Memo			Optional	Optional	N/A
IsSystem	Number			Optional	Optional	N/A

Notes:

Field Name	Note
ItemModuleType	"ExternalUnit" | "Resource" | "Activity" | "CostObject"

ValueAttributePeriodicDef table

This table specifies the default value and formula for the numeric attributes.

This table is required when importing data.

This table is required for archive.

Field Name	Type of Data	Length	Key	Import/Add	Archive	Delete
ModelName	AlphaNumeric	64		N/A	Optional	N/A
Period	AlphaNumeric	64	key	Required	Required	N/A
Scenario	AlphaNumeric	64	key	Required	Required	N/A
Reference	AlphaNumeric	64	key	Required	Required	N/A
DefaultValue	AlphaNumeric	1024		Optional	Optional	N/A
Formula	AlphaNumeric	1024		Optional	Optional	N/A

Notes:

Field Name	Note
Formula	Default=""

Part 5

Using the API

Chapter 17
Programming the API

Overview

The SAS Actvity-Based Management Web Services Integration API enables other applications or 3rd party systems to seamlessly integrate SAS Activity-Based Management into their operational systems. Users or consultants can write code or script (sample programs in several languages are provided) to automate their business reporting processes which may include periodically obtaining data from various financial data sources, cleansing and importing data in to the SAS Activity-Based Management system, computing the activity costs, building OLAP cubes for contribution or profitability analysis or exporting computed data to generate custom reports.

SAS ETL and Analytical tools combined with the SAS Activity-Based Management Integration API provide numerous options for users to integrate, automate and report on data that resides on most any system.

SAS Activity-Based Management Web Services Integration API on client is surfaced through a Microsoft COM/Automation interface. This API component on the client uses SOAP Protocol to communicate with the .NET webservices on server and hides all the complexities of sending operation requests and receiving result or logs at the completion of execution. Users who can run their integration components or code on Microsoft Windows Operating systems can effectively use this API.

Note: The API must be invoked from a machine on which the SAS Activity-Based Management client is installed.

Prerequisites for the API

- SAS Activity-Based Management client and all the dependent components must be installed prior to using client integration API.
- SAS Activity-Based Management client should be in working condition (does not have to be currently running) to use this API functionality.

Languages supported

You can invoke the API using the following languages:

- C#
- Java
- Visual C++

See the sample programs at

```
<C:\Program Files\SAS\>SASActivityBasedManagementClient\7.2\Samples\ServicesAPI
```

You can write a program that uses certain functions (such as importing data and exporting data) within SAS Activity-Based Management. You can use any COM-compliant language (such as C#, Visual Basic, or Visual C++) to access these functions, which are included in the SAS Activity-Based Management Web Services Integration API.

The SAS Activity-Based Management Web Services Integration API supports a COM dual interface: vtable (for early binding) and IDispatch (for late binding).

SAS Activity-Based Management must be installed on the computer on which you run your program.

Note: You can perform this task without first opening a model.

1. Review the information about importing model data or exporting model data.
2. If you are importing from a database:
   a. Create a database or a database view that matches the data schema and then load the data to be imported.
   b. Verify that the database user has the correct access.
3. If you are exporting to a database:
   a. Create an empty database that matches the data schema.
   b. Verify that the database user has the correct access.
4. Verify that the SAS Activity-Based Management client application is installed on the computer on which your program will run.
5. Verify that the XML import configuration, the XML export configuration, or the XML calculate configuration matches the appropriate data schema.
6. If you are using XML files, ensure that your program has access to the data:
   a. For the XML import configuration, provide the user ID and password for the DataSource attribute of either the StagingArea element or the StagingTable element.
   b. For the XML export configuration, provide the user ID and password for the Connection attribute of the StagingArea element.
7. Run your program.

   About the SAS Activity-Based Management Web Services Integration API (link???)
8. Verify the results.

   To receive a report of events that occur during the operation, your program can call the GetOperationProgress function.

Progress reports

The functions that you can perform using the SAS Activity-Based Management Web Services Integration API run on the SAS Activity-Based Management server and might require a considerable amount of time to complete. These functions regularly report progress to SAS Activity-Based Management. You can retrieve this progress report in the form of an XML string by invoking the GetOperationProgress function. Your program can read this string into the XML parser of your choice. Alternatively, your program can display the progress report information.

Here is the structure of the XML string:

```
<SasServicesStatus>
    <Operation Type="OPTYPE string" Status="OPSTATUS string" />
    <OperationMessages>
        <Message Text="Message text" />
        <Message Text="Message text" />
    </OperationMessages>
</SasServicesStatus>
```

The following table shows the possible values of the OPTYPE string:

This API function…	…has an OPTYPE string value of…
Calculate	Calculate
CopyModelData	Copy Model Data
ExportModelData	Export Model Data
ExportReportData	Export Report Data
ImportModelData	Import Model Data
ValidateModel	Verify Model

The OPSTATUS string can have any of the following values:

- Succeeded
- Failed
- Not Complete

Messages typically contain progress information, such as Exporting accounts.

Log files

The SAS Activity-Based Management Web Services Integration API creates two log files that contain information about the functions that were performed: sasservices.log and sasoperations.log. These log files are stored in the folder that contains your program's executable.

Note: These files can become large because new content is appended to the files. (The contents of these files are never overwritten.) Therefore, you should archive or delete these files periodically.

Writing a program in Java

To use the SAS Activity-Based Management Web Services Integration API from Java, a Java Virtual Machine (JVM) must be installed on the computer on which the Java class is being executed. Also, a bridge is necessary for Java and COM to exchange data. SAS recommends Jawin.

Here is an example of the Java source code that you might use to access the ExportModelData function. Notice that Jawin converts an unsuccessful COM call to a Java exception.

```
try
{
    Ole32.CoInitialize();    // Initialize COM
    Application api = new _Application("new:SasServices.API");
    api.Connect( “MyUserName”, “MyPassword” );
    String xmlConfig;
    String operationKey;
// ... Build the configuration xml
    api.ExportModelData( xmlConfig, operationKey );
// ... At some later time
    String xmlExportReport;
    api.GetOperationProgress( operationKey, xmlExportReport);
    api.Disconnect();
    Ole32.CoUninitialize(); // Shutdown COM
}

catch(COMException e)
{
    System.out.println("Got a COM Error: " + e);
}

catch(Exception e)
{
    e.printStackTrace();   // Handle an exception.
}
```

Example programs

Example programs that are written in various programming languages are installed with SAS Activity-Based Management. The default folder for these example programs is:

```
C:\Program Files\SAS\SASActivityBasedManagementClient\7.2\Samples\ServicesAPI
```

Chapter 18
API Functions

Calculate

This function calculates a model in the SAS Activity-Based Management database.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to calculate a model (Example)
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

CancelOperation

This function cancels the import, export, or calculation operation in progress.

Type	Argument	Data Type	Description
IN	OperationKey	BSTR	GUID string that uniquely identifies the operation

CopyModelData

This function copies model data from one period/scenario association to another.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to copy model data
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

Connect

This function initializes a session with the SAS Activity-Based Management server.

Type	Argument	Data Type	Description
IN	ServerName	BSTR	SAS Activity-Based Management server name
IN	UserName	BSTR	SAS Activity-Based Management user with sufficient permissions to perform the operations in the program. A domain name may be required, such as when connecting to a server.

Type	Argument	Data Type	Description
IN	Password	BSTR	Plain text password for the SAS Activity-Based Management user

Disconnect

This function ends the session with the SAS Activity-Based Management server. This function has no arguments.

ExportCubeConfigurations

This function exports cube configurations to an xml file.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to export cube configurations
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

ExportModelData

This function exports model structures and data to a Microsoft SQL Server, Oracle, or Microsoft Access database.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to export(Example)
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

ExportReportData

This function exports report data to a data source.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to export report data
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

GenerateCube

This function generates a cube and/or a fact table.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to generate a cube (Example)
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

GetOperationLog

This function retrieves the operation log as XML.

Type	Argument	Data Type	Description
IN	OperationKey	BSTR	GUID string that uniquely identifies the operation
OUT	Log	BSTR	The log in XML format

GetOperationProgress

This function retrieves progress messages regularly reported by SAS Activity-Based Management operations in the form of an XML string progress report.

Type	Argument	Data Type	Description
IN	OperationKey	BSTR	GUID string that uniquely identifies this operation
OUT	xmlReport	BSTR	The reported progress of the operation

ImportModelData

This function imports model structures and data into a SAS Activity-Based Management model from a Microsoft SQL Server, Oracle, or Microsoft Access database, or from an XML file that was previously exported from SAS Activity-Based Management.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to import (Example)
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

ImportCubeConfigurations

This function imports cube configurations from a previously exported xml file.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to import cube configurations
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

OnOperationComplete

This function retrieves the status code at the end of an operation.

Type	Argument	Data Type	Description
IN	OperationKey	BSTR	GUID string that uniquely identifies the operation
OUT	Status	Integer	The status code for the operation. The status code may be one of the following: 0=OPSTATUS_ FAILED 1=OPSTATUS_ SUCCEEDED 2=OPSTATUS_ UNDETERMINED

PublishPeriodsScenarios

This function publishes periods and their associated scenarios.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to publish periods and scenarios
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

PublishSPMMetrics

This function logs in to the SAS Strategy Management server (middle tier). This is the same information that is specified in the model properties.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to publish SAS Strategy Management metrics
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

ValidateModel

This function validates a model.

Type	Argument	Data Type	Description
IN	XML	BSTR	XML to validate model
OUT	OperationKey	BSTR	GUID string that uniquely identifies the operation

Chapter 19

XML Passed to the API

Overview

When you use certain functions in the SAS SAS Activity-Based Management Web Services Integration API, you must provide an XML configuration string. This string may be included in your program, contained in a file, or entered through your program's user interface.

Each XML configuration has a specific purpose and a unique structure. Example XML configuration strings are available in the folder in which SAS Activity-Based Management is installed, which is typically:

C:\Program Files\SAS\Activity-Based Management Solution\Client\Samples\Services API\Configurations

Note: Some of the attributes in an XML configuration contain IDs to other data that resides on a specific SAS Activity-Based Management server, such as periods and scenarios. Therefore, do not use an XML configuration that was created for another server. If you use an XML configuration created for one server, these IDs might not correspond to any data on another server. Or, in the worst case, the IDs might correspond to the wrong data and the model might become corrupted.

XML to Calculate a Model

Sample XML

```
<OROSCOMMAND Version="2.0">
   <MODELCONTEXT Model="peAPI">
      <PeriodScenario PeriodId="9" ScenarioId="1"/>
   </MODELCONTEXT>
   <COMMANDPARAMS MessageLimit="50">
      <CostFlow SeqDriverLimit="-1" Force="1" DisableDriverRules="0"/>
   </COMMANDPARAMS>
</OROSCOMMAND>
```

```
<OROSCOMMAND Version="2.0">
   <MODELCONTEXT Model="peAPI">
      <PeriodScenario PeriodRef="2008 Q1" ScenarioRef="Actual" />
   </MODELCONTEXT>
   <COMMANDPARAMS MessageLimit="50">
      <CostFlow SeqDriverLimit="-1" Force="1" DisableDriverRules="0"/>
   </COMMANDPARAMS>
</OROSCOMMAND>
```

Overview

When you use the SAS | ABM Web Services Integration API to calculate a model, you must write XML that specifies what you want to calculate and how you want to calculate it.

Example XML to Calculate a Model

Elements and attributes are case sensitive. String comparisons are not case sensitive.

XML Syntax

The following structure shows the elements used in the XML and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<OROSCOMMAND Version="2.0">
    <MODELCONTEXT attributes>
        <PeriodScenario attributes/>
    </MODELCONTEXT>
    <COMMANDPARAMS attributes>
        <CostFlow attributes/>
    </COMMANDPARAMS>
</OROSCOMMAND>
```

OROSCOMMAND Element (Required)

OROSCOMMAND is the root element. This element has one version attribute. The OROSCOMMAND element must contain one MODELCONTEXT element and one COMMANDPARAMS element.

Attribute	Required	Values	Description
Version	Optional	"2.0"	If you omit the version number, then the command reverts to version 1 syntax which is supported but deprecated. Version 1 syntax is significantly different.

MODELCONTEXT Element (Required)

This element specifies the model to calculate. A MODELCONTEXT element can contain multiple PeriodScenario elements.

Attribute	Required	Values	Description
Model	Required	String	Full Workspace path of the model, excluding the Models Workspace folder

PeriodScenario Element (Required)

Each PeriodScenario element specifies a period/scenario association to calculate. You can have multiple PeriodScenario elements.

Attribute	Required	Values	Description
PeriodId | PeriodRef	Required	Numeric if ID, String if Reference	Period
ScenarioId | ScenarioRef	Required	Numeric if ID, String if Reference	Scenario

COMMANDPARAMS Element (Required)

This element configures the calculate operation. A COMMANDPARAMS element can have one CostFlow element.

Attribute	Required	Values	Description
MessageLimit	Optional	String that represents an integer between and including 0 and 2,147,483,648	Sets the maximum number of messages to include for each error/warning message type. Defaults to 50. If value is less than zero, all messages are included in the output. A message type is equivalent to a specific type of error or warning. The error/warning may apply to more than one item, so a message is normally generated for each item.

CostFlow Element (Required)

This element specifies information about the calculation.

Attribute	Required	Values	Description
Diagnostic	Optional	"yes", "no"	Indicates whether detailed information about the calculation is returned. This information includes data such as the module that is being calculated and the amount of time that is required to calculate.
Force	Optional	"1" or "0"	"1" means to perform calculations even if a system flag says that calculations are up to date. "0" means to perform calculations only if a system flag says that calculations are out of date. Omitting the Force attribute is the same as Force="0".
DisableDriverRules	Optional	"1", "0"	"1" means to ignore rule-based drivers. Do not generate assignments for any rule-based drivers. Specifying "1" for this option causes assignment generation not to take place, which means that assignments created by a previous calculation are retained. "0" is the default. It means that rule-based drivers are enabled and generate assignments.
SeqDriverLimit	Optional	"-1" "1" - "99"	Sequence number at which to stop the calculation. "-1" indicates that driver sequencing is not used

XML to Copy a Model

Sample XML

The following example copies model data from one period and scenario (IDs 1 and 3) to another period and scenario (IDs 6 and 2):

```
<CopyModelData>
    <ModelContext ModelId="1001" SrcPeriod="1" DestPeriod="6"
                              SrcScenario="3" DestScenario="2" />
</CopyModelData>
```

Overview

When you use the SAS ABM Web Services Integration API to calculate a model, you must write an XML CopyModelData configuration that specifies what you want to calculate (and how you want to calculate it).

String comparisons are not case sensitive.

XML copy model data configuration structure

The following structure shows the elements used in the XML CopyModelData configuration and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<CopyModelData>
   <ModelContext attribute />
</CopyModelData>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.N

CopyModelData element (required)

CopyModelData is the root element. This element has no attributes. It must contain one ModelContext element.

ModelContext element (required)

This element specifies the model and the model's period/scenario association to copy.

Attribute	Required	Values	Description
ModelId	Required	String	ID of the model
SrcPeriod	Required	String	ID of the period to copy from
SrcScenario	Required	String	ID of the scenario to copy from

Attribute	Required	Values	Description
DestPeriod	Required	String	ID of the period to copy to
DestScenario	Required	String	ID of the scenario to copy to

Related Topics:

- Example XML Copy Model Data configuration

XML to Export Cube Configurations

Sample XML

```
<CubeConfigurationsExport Filename="c:\temp\peCubeConfigurationsExport.xml" DoFolders="True">
    <CubeConfiguration Name="Fred\PE_MSC63_Modules" />
    <CubeConfiguration Name="Fred\PE_MSC63_Stages" />
    <CubeConfiguration Name="Fred\PE_MSC64_Modules" />
    <CubeConfiguration Name="Fred\PE_MSC64_Stages" />
    <CubeConfiguration Name="Fred\PE_RC" />
    <CubeConfiguration Name="Fred\PE_SS" />
</CubeConfigurationsExport>
```

Overview

When you use the SAS Activity-Based Management Web Services Integration API to export cube configurations, you must write XML that specifies the cube configurations to be exported.

Elements and attributes are case sensitive. String comparisons are not case sensitive.

XML syntax

The following structure shows the elements used in the XML and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<CubeConfigurationsExport attributes >
    <CubeConfiguration attribute />
</CubeConfigurationsExport>
```

CubeConfigurationsExport element (required)

CubeConfigurationsExport is the root element. This element specifies the name of the file to be created during the export. The CubeConfigurationsExport element can contain multiple CubeConfiguration elements.

Attribute	Required	Values	Description
Filename	Required	String	Specifies the complete path, including the file name, of the file to be exported to. If the file already exists, the file is replaced.
DoFolders	Optional	"True", "False"	"True" stores folder information in the exported file. When you reimport the file the folders are recreated. "False" causes no folder information to be stored in the exported file. On reimporting, all cube configurations are stored in the topmost Cube Configurations folder of the Workspace Manager. You should use this option only if no two cube configurations have the same name.

CubeConfiguration element (required)

Each CubeConfiguration element specifies the name and path of a cube configuration to be exported.

Attribute	Required	Values	Description
Name	Required	String	Specifies the name and complete path (folders) of the cube configuration to be exported.

XML to Export a Model

Sample XML

The following example exports model data from a model named Inflight to a database:

```
<OEExport Model="Inflight" Version="7.2" Type="Staging" MessageLimit="50" Filename="">
    <PeriodScenario Period="2000" Scenario="Actual" />
    <StagingArea
        JdbcDriverClass=""
        RepositoryName=""
        RepositoryId=""
        LibraryName=""
        LibraryID=""
        LibraryReference=""
        MetaWorkspaceServer=""
        MetaWorkspaceServerId=""
        WorkspaceServerName=""
        Port=""
        Engine=""
        DBType="1"
        DriverType="0"
        HostName="D7920"
        PortNumber="1433"
        ServiceName="TestDBForStagingTables"
        UserName="DBUser"
        Password="xxxx"
        AdvanceOptions="Security=SSPI;Persist Security Info=False;">
        <StagingTable Name="Period" TableName="Period">
            <Column Name="ModelId" ColumnName="ModelId" />
            <Column Name="PeriodId" ColumnName="PeriodId" />
            <Column Name="Name" ColumnName="Name" />
            <Column Name="Reference" ColumnName="Reference" />
            <Column Name="ParentId" ColumnName="ParentId" />
            <Column Name="ParentReference" ColumnName="ParentReference" />
            <Column Name="StartDate" ColumnName="StartDate" />
            <Column Name="EndDate" ColumnName="EndDate" />
            <Column Name="Description" ColumnName="Description" />
        </StagingTable>
        <StagingTable Name="Scenario" TableName="Scenario">
            <Column Name="ModelId" ColumnName="ModelId" />
            <Column Name="ScenarioId" ColumnName="ScenarioId" />
            <Column Name="Name" ColumnName="Name" />
            <Column Name="Reference" ColumnName="Reference" />
            <Column Name="ParentId" ColumnName="ParentId" />
            <Column Name="ParentReference" ColumnName="ParentReference" />
            <Column Name="Description" ColumnName="Description" />
        </StagingTable>
    </StagingArea>
</OEExport>
```

The following example exports model data from a model named "Inflight" to XML instead (by replacing the previous first three lines):

```
<OEExport
    Model="Inflight"
    Version="6.2"
    Type="XML"
    MessageLimit="50"
    Filename="c:\temp\inflight.xml">
    <PeriodScenario Period="2000" Scenario="Actual" />
    <StagingArea DataSource="">
```

Overview

When you use the SAS Activity-Based Management Web Services Integration API to export information from SAS Activity-Based Management, you must write an XML that identifies the model to be exported and the destination database or XML file. The XML can select all periods and scenarios to export or a subset of them. It can specify a whole model or specific tables (as defined in the data schema) to export.

Example XML to Export

String comparisons are not case sensitive.

XML Syntax

The following structure shows the elements used in the export XML and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<OEExport attributes />
<PeriodScenario attributes />
<StagingArea attributes>
    <StagingTable attributes >
        <Column attributes />
        <ColumnFilter>
            <![CDATA[ expression ]]>
        </ColumnFilter>
        <DimensionColumn attributes />
        <DimensionFilter>
            <Dimension attribute />
            <DimensionMember attributes />
            </Dimension>
        </DimensionFilter>
        <NestedObject attributes>
            <Column attributes />
            <DimensionColumn attributes />
        </NestedObject/>
    </StagingTable>
</StagingArea>
</OEExport>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.

OEExport Element (Required)

OEExport is the root element. This element specifies general export information.

Attribute	Required	Values	Description
Model	Required	String	Full Workspace path to the SAS Activity-Based Management model, excluding the Workspace Name folder and Models folder; the separator character between folders is the backslash (\); the path is not case sensitive
Version	Required	String	SAS Activity-Based Management version number
Type	Required	"Staging", "XML"	Format for the exported file; "Staging" is for a database, "XML" is for XML
MessageLimit	Optional	String	Maximum number of similar error messages to write to the log file; when this limit is reached, no other similar error messages are written; default is "50"
Filename	Required when Type is "XML"	String	Absolute path and file name for the exported XML file

PeriodScenario Element (Required)

This element specifies a period/scenario association. The OEExport element can have multiple PeriodScenario elements.

Attribute	Required	Values	Description
Period	Required	String	Reference for the period
Scenario	Required	String	Reference for the scenario

StagingArea Element (Required)

This element specifies the beginning of the export information.

Attribute	Required	Values	Description
JdbcDriverClass	Optional	String	You must specify this attribute if you are using a JDBC driver and the database type is Other. Otherwise you can omit this attribute.
RepositoryName	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
RepositoryId	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
LibraryName	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
LibraryID	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.

Attribute	Required	Values	Description
LibraryReference	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
MetaWorkspaceServer	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
MetaWorkspaceServerId	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
WorkspaceServerName	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
Port	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.

Attribute	Required	Values	Description
Engine	Optional	String	You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
DBType	Required	String	Specify one of the following numbers from 0 to 6: 0 SAS 1 SQLServer 2 Oracle 3 MySQL 4 Microsoft Access 5 Microsoft Excel 6 Other
DriverType	Required	String	Specify either 0 or 1: 0 JDBC 1 SAS/ACCESS
HostName	Optional	String	For the following databases, specify the information indicated: SQLServer Machine name MySQL Machine name Oracle Machine name Microsoft Access Physical file path Microsoft Excel Physical file path Otherwise you can omit this attribute.

Attribute	Required	Values	Description
PortNumber	Optional	String	You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.
ServiceName	Optional	String	For the following databases, specify the information indicated: SQLServer Database name MySQL Database name Oracle Oracle service name where the database instance is running. Otherwise you can omit this attribute.
UserName	Optional	String	You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.
Password	Optional	String	You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute. *Note:* The password, if specified, is encrypted.

Attribute	Required	Values	Description
AdvanceOptions	Optional	String	Specify optional parameters for the following databases: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.

StagingTable Element (Required)

This element identifies the table and the columns of a data schema to export. The StagingArea element can have none or more StagingTable elements.

Attribute	Required	Values	Description
%> Name	Required	String	Name of the table
TableName	Optional	String	Name of the table in the exported file

Column Element (Required)

This element corresponds to a column in a table of the data schema.

Attribute	Required	Values	Description
Name	Required	String	Name of the column
ColumnName	Optional	String	Name of the column in the exported file

ColumnFilter Element (Optional)

This element identifies a filter that is based on the values of properties to filter the export data.

![CDATA [expression]] Element (Required)

This element specifies an expression that is used for the filter.

Attribute	Required	Values	Description
Expression	Required	String	Expression

DimensionColumn Element (Optional)

This element corresponds to a column that contains dimension information in a table of the data schema.

Attribute	Required	Values	Description
Name	Required	String	Name of the column that contains dimension information
ColumnName	Optional	String	Name of the column in the exported file
DimensionReference	Required for dimension information	"Resource" "Activity" "Cost Object" "External Units"	Reference of the dimension

DimensionFilter Element (Optional)

This element identifies a filter that is based on a dimension to filter the export data.

Dimension Element (Required)

This element specifies a dimension that is used for the filter.

Attribute	Required	Values	Description
Reference	Required	String	Reference of the dimension

DimensionMember Element (Optional)

This element specifies the dimension member (or members) that is used for the filter.

Attribute	Required	Values	Description
Reference	Required	String	Reference of the dimension member
IncludeChildren	Required	"yes", "no"	Indicates whether the children of the dimension member are included

NestedObject Element (Optional)

This element specifies an item that is related to another item.

Attribute	Required	Values	Description
Relation	Required	"Attribute" "Destination" "Item" "Source"	Method by which the item is related to the other item

XML to Export a Report

Sample XML

The following example exports report data for a report named Destination Furthest from the model with ID 1001.

```
<Object Custom="1" version="7.2" Name="Destination Furthest">
    <Periods Names="2000" Ids="1" />
    <Scenarios Names="Actual" Ids="1" />
    <Models Modules="ALL">
        <Model Id="1001" />
    </Models>
    <Dimensions IdStr="" IdStrWiz="" DimNames="" />
    <Connection
    JdbcDriverClass=""
    RepositoryName=""
    RepositoryId=""
    LibraryName=""
    LibraryID=""
    LibraryReference=""
    MetaWorkspaceServer=""
    MetaWorkspaceServerId=""
    WorkspaceServerName=""
    Port=""
    Engine=""
    DBType="1"
    DriverType="0"
    HostName="D7920"
    PortNumber="1433"
    ServiceName="TestDBForStagingTables"
    UserName="DBUser"
    Password="xxxx"
    AdvanceOptions="Security=SSPI;Persist Security Info=False;" />
</Object>
```

Overview

When you use the SAS | ABM Web Services Integration API to export a report, you must write an XML export report data configuration that identifies the report to be exported.

String comparisons are not case sensitive.

XML export report data configuration structure

The following structure shows the elements used in the XML export report data configuration and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<Object attributes>
    <Periods attributes/>
    <Scenarios attributes/>
    <Models attribute/>
       <Model attribute/>
    </Models>
    <Dimensions attributes/>
    <Options attribute/>
    <Connection attributes/>
</Object>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.

Object element (required)

Object is the root element.

Attribute	Required	Values	Description
Custom	Required	"1"	Must always be "1"
Version	Required	"2"	Must always be "2"
Name	Required	String	Name of the report

Periods element (required)

This element specifies the periods.

Attribute	Required	Values	Description
Names	Required	String	Names of one or more periods; separate each name with a comma

Attribute	Required	Values	Description
Ids	Required	String	IDs of one or more periods; separate each ID with a comma

Scenarios element (required)

This element specifies the scenarios.

Attribute	Required	Values	Description
Names	Required	String	Names of one or more scenarios; separate each name with a comma
Ids	Required	String	IDs of one or more scenarios; separate each ID with a comma

Models element (required)

This element specifies the modules for all selected models.

Attribute	Required	Values	Description
Modules	Required	"Resource" "Activity" "Cost Object" "ALL"	Names of one or more modules; separate each name with a comma

Model element (required)

This element specifies the model. You can have multiple Model elements.

Attribute	Required	Values	Description
Id	Required	String	ID of a model

Dimensions element (required for certain reports)

This element specifies the dimensions to include. This element is required for the Dimensional View report and the Profit Cliff report; it is optional for all other reports.

Attribute	Required	Values	Description
IdStr* **	Required/ Optional	String***	IDs of one or more dimensions or dimension members; separate each ID with a comma The format is: "<ID>|24" (for example, "2004|24" or "2003|24,2004|24")
IdStrWiz	Optional	String	Used by the Report Data wizard only
DimNames**	Required/ Optional	String***	Names of one or more dimensions; separate each name by a comma

* Use only when exporting from one model.

** If both IdStr and DimNames are specified, DimNames is used.

*** Specify either IdStr and PLAttributeDim or DimNames and PLAttributeDimName. See also note 2.

Connection element (required)

This element specifies the information to communicate with the export database.

Attribute	Required	Values	Description
ConnectionStr	Required	String	Connection string
DestTableName	Required	String	Name of the export database table

XML to Generate a Cube

Overview

When you use the SAS | ABM Web Services Integration API to generate a cube using a cube configuration, you must write XML that specifies what cube configuration you want to use.

Example XML to Generate a Cube

Elements and attributes are case sensitive. String comparisons are not case sensitive.

Sample XML

```
<OROSCOMMAND Version="2.0">
    <MODELCONTEXT ModelId="1242">
        <PeriodScenario PeriodId="13" ScenarioId="1"/>
        <PeriodScenario PeriodId="14" ScenarioId="1"/>
    </MODELCONTEXT>
    <COMMANDPARAMS MessageLimit="50" CubeAction="Generate">
        <CubeConfig Id="837"/>
    </COMMANDPARAMS>
</OROSCOMMAND>
```

XML Syntax

The following structure shows the elements used in the XML to generate a cube and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

<OROSCOMMAND Version="2.0">

<MODELCONTEXT attributes>

< PeriodScenario attributes/>

</MODELCONTEXT>

<COMMANDPARAMSattributes>

<CubeConfig attributes/>

</COMMANDPARAMS>

</OROSCOMMAND>

OROSCOMMAND Element (Required)

OROSCOMMAND is the root element. This element has one version attribute. The OROSCOMMAND element must contain one MODELCONTEXT element and one COMMANDPARAMS element.

Attribute	Required	Values	Description
Version	Optional	"2.0"	If you omit the version number, then the command reverts to version 1 syntax which is supported but deprecated. Version 1 syntax is significantly different.

MODELCONTEXT Element (Required)

This element specifies the model for which to generate a cube. An MODELCONTEXT element can have multiple PeriodScenario elements.

Attribute	Required	Values	Description
Model | ModelID	Required	String if Model name; Numeric if ModelID	Full Workspace path of the model, excluding the Models Workspace folder if you use the Model Name. Specifying the path is not necessary if you use the ModelID.

PeriodScenario Element (Required)

Each PeriodScenario element specifies a period/scenario association to generate. You can have multiple PeriodScenario elements.

Attribute	Required	Values	Description
PeriodId | PeriodRef	Required	Numeric if ID, String if Reference	Period
ScenarioId | ScenarioRef	Required	Numeric if ID, String if Reference	Scenario

COMMANDPARAMS Element (Required)

This element configures the calculate operation. An COMMANDPARAMS element can have one CubeConfig element.

Attribute	Required	Values	Description
CubeAction	Optional	"Generate", "CountRows"	"Generate" generates a cube and/or fact table. "CountRows" counts the rows in the cube to be generated.
MessageLimit	Optional	String that represents an integer between and including 0 and 2,147,483,648	Sets the maximum number of messages to include for each error/warning message type. Defaults to 50. If value is less than zero, all messages are included in the output. A message type is equivalent to a specific type of error or warning. The error/warning may apply to more than one item, so a message is normally generated for each item.

CubeConfig Element (Required)

This element specifies the cube configuration to be used in the generation.

Attribute	Required	Values	Description
ID	Required	Numeric	ID of the cube configuration to be used for generating the cube and/or fact table.

Configuring How Cubes Are Generated

When SAS Activity-Based Management generates cubes, it includes all dimensions to provide the most flexible data analysis. The cubes include redundant information so that the information is available from many data analysis perspectives. However, if your model contains a large amount of information, the process of generating the cubes and viewing them interactively on the OLAP page can take a long time.

To improve performance, a Modeler can create a configuration file that controls which dimensions are included in the cubes. This configuration file must be named CubeConfig.xml and it must be located in the Enterprise Server folder in the product installation folder (typically: **`C:/Program Files/SAS/Activity-Based Management Solution/Enterprise Server`**). If this configuration file is not in this location, SAS Activity-Based Management generates the cubes by using the default information.

The XML in the configuration file must conform to the XML cube configuration schema. (See: Example of the configuration file)

Errors in the Cube Configuration File

When cubes are generated and the configuration file is processed, the following errors in the configuration file are reported:

- Failed to parse the configuration file

  Although the configuration file exists in the appropriate location, the XML code within the configuration file contains one or more errors.

- Invalid cube name

  The name that was specified for a cube is not valid. See: About the cube names.

The following errors in the configuration file are not reported:

- Invalid stage names

  The name that was specified for a stage is not valid. See: About the stage names.

- Invalid dimension names

  The name that was specified for a dimension is not valid. See: About dimension names.

- A cube configuration that does not specify any dimensions

The configuration file does not contain the required dimension element.

XML to Import Cube Configurations

Sample XML

```
<CubeConfigurations UploadFile="c:\temp\peCubeConfigurationsExport.xml" Action="rename"/>
```

Overview

When you use the SAS | ABM Web Services Integration API to import cube configurations, you must write XML to specify the file to be imported.

Elements and attributes are case sensitive. String comparisons are not case sensitive.

XML syntax

The XML to import cube configurations consists of a single element.

```
< CubeConfigurations attributes/>
```

CubeConfigurations element (required)

CubeConfigurations is the root and only element. It specifies the name of the file to be imported.

Attribute	Required	Values	Description
UploadFile	Required	String	Specifies the complete path, including the file name of the file to be imported.

Attribute	Required	Values	Description
Action	Optional "rename" is the default	"rename" "replace" "donotimport"	"rename" If a cube configuration exists with the same name as a cube configuration being imported, the one that you are importing is renamed. This applies to every cube configuration being imported if multiple cube configurations are imported. "replace" If a cube configuration exists with the same name as a cube configuration being imported, then the existing cube configuration is replaced with the imported one. "donotimport" If a cube configuration exists with the same name as a cube configuration that you are importing, then the duplicate cube configuration is not imported. Leave the existing one in place.

XML to Import a Model

Sample XML

The following example imports a model named TestModel into SAS Activity-Based Management:

```
<OEImport
  Model="TestModel"
  Action="Create"
  Version="7.2"
  Type="Staging"
  Description="TestModel import">
    <StagingArea
       JdbcDriverClass=""
           RepositoryName=""
```

```
    RepositoryId=""
    LibraryName=""
    LibraryID=""
    LibraryReference=""
    MetaWorkspaceServer=""
    MetaWorkspaceServerId=""
    WorkspaceServerName=""
    Port=""
    Engine=""
    DBType="1"
    DriverType="0"
    HostName="D920"
    PortNumber="1433"
    ServiceName="TestDBForStagingTables"
    UserName="DBUser "
    Password="xxxx"
    AdvanceOptions="Security=SSPI;Persist Security Info=False;">
<StagingTable Name="Period" TableName="tbl_Period">
    <Column Name="Reference" />
    <Column Name="Name" />
    <Column Name="ParentReference" />
    <Column Name="StartDate" />
    <Column Name="EndDate" />
</StagingTable>
<StagingTable Name="PeriodLevel" TableName="tbl_PeriodLevel">
    <Column Name="LevelNo" />
    <Column Name="Name" />
</StagingTable>
<StagingTable Name="Scenario" TableName="tbl_Scenario">
    <Column Name="Reference" />
    <Column Name="Name" />
    <Column Name="ParentReference" ColumnName="ParentRef" />
</StagingTable>
<StagingTable Name="ScenarioLevel" TableName="tbl_ScenarioLevel">
    <Column Name="LevelNo" />
    <Column Name="Name" />
</StagingTable>
<StagingTable Name="CurrencyRate" TableName="tbl_CurrencyRate">
    <Column Name="Period" />
    <Column Name="Scenario" />
    <Column Name="CurrencyFrom" />
    <Column Name="CurrencyTo" />
    <Column Name="Rate" />
</StagingTable>
<StagingTable Name="Driver" TableName="tbl_Driver">
    <Column Name="Name" />
    <Column Name="DriverType" />
    <Column Name="UniqueDriverQuantities" ColumnName="UniqueQuantity" />
</StagingTable>
<StagingTable Name="Dimension" TableName="tbl_Dimension">
    <Column Name="Reference" ColumnName="DimRefnum" />
    <Column Name="Name" ColumnName="DimName" />
</StagingTable>
<StagingTable Name="DimensionOrder" TableName="tbl_DimViewOrder">
    <Column Name="ModuleType" />
    <Column Name="DimRef" ColumnName="DimRefnum" />
```

```
        <Column Name="SequenceNumber" ColumnName="SequenceNo" />
    </StagingTable>
    <StagingTable Name="DimensionLevel" TableName="tbl_DimLevel">
        <Column Name="LevelNo"/>
        <Column Name="DimRef" />
        <Column Name="Name" />
    </StagingTable>
    <StagingTable Name="DimensionMember" TableName="tbl_DimMember">
        <Column Name="DimRef" ColumnName="DimRefnum" />
        <Column Name="Name" />
        <Column Name="Reference" />
        <Column Name="ParentReference" />
        <Column Name="DimLevelName" />
        <Column Name="DimLevel" />
    </StagingTable>
    <StagingTable Name="Account" TableName="tbl_Account">
        <Column Name="ModuleType" ColumnName="Module" />
        <Column Name="Period" />
        <Column Name="Scenario" />
        <Column Name="DriverName" ColumnName="Driver" />
        <Column Name="Reference" />
        <Column Name="TDQUE" />
    </StagingTable>
    <StagingTable Name="Account" TableName="tbl_AccountUpdate">
        <Column Name="ModuleType" ColumnName="Module" />
        <Column Name="Period" />
        <Column Name="Scenario" />
        <Column Name="DriverName" ColumnName="Driver" />
        <Column Name="Reference" />
        <Column Name="TDQUE" />
    </StagingTable>
    <StagingTable Name="EnteredCostElement" TableName="tbl_EnteredCE">
        <Column Name="ModuleType" ColumnName="Module" />
        <Column Name="Period" />
        <Column Name="Scenario" />
        <Column Name="Reference" />
        <Column Name="EnteredCost" ColumnName="Cost" />
        <Column Name="Name" />
    </StagingTable>
    <StagingTable Name="Assignment" TableName="tbl_IBOCSourceDestinationByDimension">
        <Column Name="Period" />
        <Column Name="Scenario" />
        <Column Name="SourceModuleType" ColumnName="SrcModule" />
        <Column Name="DestinationModuleType" ColumnName="DestModule" />
        <Column Name="DriverQuantityFixed" ColumnName="FixedQuantity" />
        <Column Name="DriverQuantityVariable" ColumnName="VariableQuantity" />
        <Column Name="DriverName" ColumnName="Driver" />
    </StagingTable>
    <StagingTable Name="Assignment" TableName="tbl_AssignmentsSourceDestinationByDimension">
        <Column Name="Period" />
        <Column Name="Scenario" />
        <Column Name="SourceModuleType" ColumnName="SrcModule" />
        <Column Name="DestinationModuleType" ColumnName="DestModule" />
        <Column Name="DriverQuantityFixed" ColumnName="FixedQuantity" />
        <Column Name="DriverQuantityVariable" ColumnName="VariableQuantity" />
    </StagingTable>
```

```
        <StagingTable Name="ExternalUnit" TableName="tbl_ExternalUnit">
            <Column Name="Period" />
            <Column Name="Scenario" />
            <Column Name="Reference" />
            <Column Name="UnitCostEntered" />
        </StagingTable>
        <StagingTable Name="AssignmentNonUnique" TableName="tbl_AssignmentNonUnique">
            <Column Name="Period" />
            <Column Name="Scenario" />
            <Column Name="DriverName" />
            <Column Name="DriverQuantityFixed" ColumnName="DriverQuantityFixed" />
            <Column Name="DriverQuantityVariable" ColumnName="DriverQuantityVariable" />
            <Column Name="DestinationModuleType" />
        </StagingTable>
        <StagingTable Name="ValueAttribute" TableName="tbl_Attribute">
            <Column Name="Type" />
            <Column Name="Reference" />
            <Column Name="Name" />
            <Column Name="ParentReference" ColumnName="Parent" />
        </StagingTable>
        <StagingTable Name="ValueAttributeAssociation" TableName="tbl_ValueAttributeAssociation">
            <Column Name="Period" />
            <Column Name="Scenario" />
            <Column Name="ItemModuleType" />
            <Column Name="AttributeReference" />
            <Column Name="Value" />
        </StagingTable>
        <StagingTable Name="DimensionAttributeAssociation" TableName="tbl_DimensionAttributeAssociation">
            <Column Name="Period" />
            <Column Name="Scenario" />
            <Column Name="ItemModuleType" />
            <Column Name="AttributeDimRef" />
            <Column Name="AttributeDimMemberRef" />
        </StagingTable>
    </StagingArea>
</OEImport>
```

Overview

When you use the SAS Activity–Based Management Web Services Integration API to import data, you must write XML that maps the structures and periodic data in a database or XML file to the model database.

Create one XML import configuration for new structures. Create a different XML import configuration to delete or update existing structures. Do not add, delete, and update structures in the same XML import configuration.

For importing from an XML file, the configuration does not have to contain detailed table and column information. A sample XML import configuration can be as follows:

```
<OEImport Version="1.1" Type="XML" Action="Create" Model="Test" Filename="C:\temp\TestModel.xml"></OEImport>
```

The following elements and attributes specify the structures and periodic data to import into a model. Elements and attributes are case sensitive. String comparisons are not case sensitive.

Each XML can import into one model.

XML Syntax

The following structure shows the elements used in the XML and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<OEImport attributes >
    <StagingArea attribute >
        <StagingTable attributes >
            <Column attributes >
        </StagingTable>
    </StagingArea>
 </OEImport>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.

OEImport Element (Required)

OEImport is the root element. This element specifies general import information. You can have one OEImport element in an XML import configuration.

If you include diagnostic information, you will see LoadCache elements within ImportTable elements.

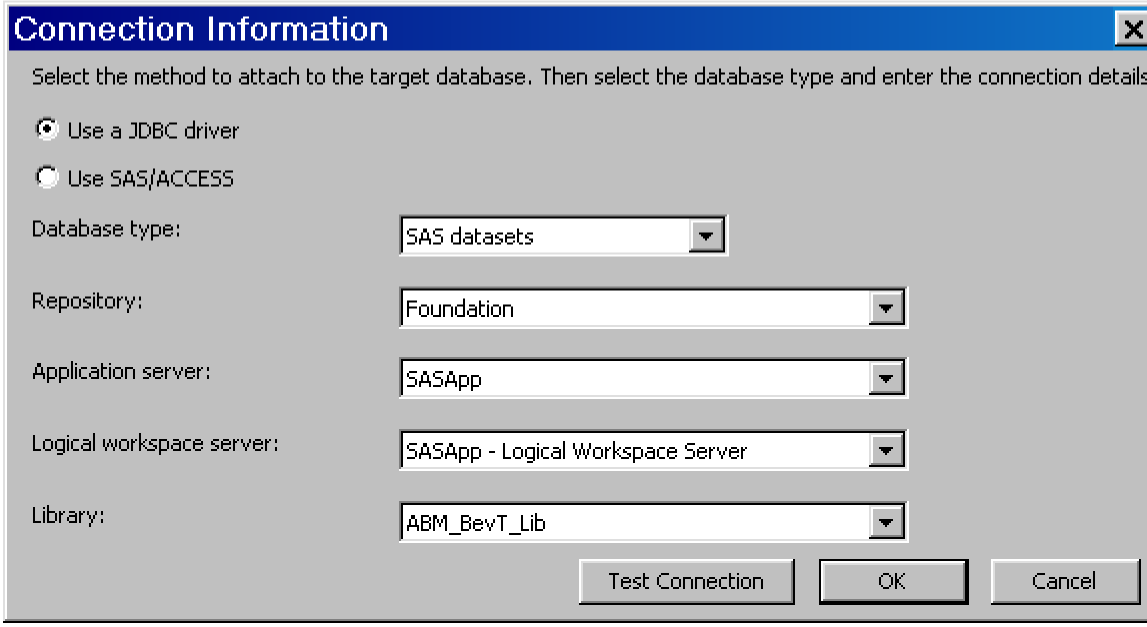
Attribute	Required	Values	Description
Model	Required	String	Full Workspace path, excluding the Workspace Name folder and Models folder; the separator character between folders is the backslash (\); if the model does not exist, a new one is created; if the model exists, the import process adds and deletes structures and periodic data in the existing model; new periods and scenarios can be created

Attribute	Required	Values	Description
Action	Required	<blank> Create ReplaceAll	Inserts and merges data into a model Create creates a new model ReplaceAll removes all model-specific data from the existing model before processing information; periods and scenarios, which are not model-specific, are not removed
Version	Required	String	SAS Activity-Based Management version number
CustomSchema		1	Must always be 1
Type	Required	XML, Staging	Type of import
Filename	Required when Type is XML	String	Absolute path and file name for the exported XML file
Description	Optional	String	Description of the import

StagingArea Element (Required)

This element defines the connection information. An OEImport element can have one or more StagingArea elements.

Note: The system generates a connection string when you use the Connection Information dialog to connect to a database.



Attribute	Required	Values	Description
JdbcDriverClass	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using a JDBC driver and the database type is Other. Otherwise you can omit this attribute.
RepositoryName	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.

Attribute	Required	Values	Description
RepositoryId	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
LibraryName	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
LibraryID	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
LibraryReference	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.

Attribute	Required	Values	Description
MetaWorkspaceServer	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
MetaWorkspaceServerId	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
WorkspaceServerName	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
Port	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.

Attribute	Required	Values	Description
Engine	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute if you are using either of the following: • SAS datasets • SAS/Access Otherwise you can omit this attribute.
DBType	Required	String	% " class=TableCell style="width: 40%;"> Specify one of the following numbers from 0 to 6: 0 SAS 1 SQLServer 2 Oracle 3 MySQL 4 Microsoft Access 5 Microsoft Excel 6 Other
DriverType	Required	String	% " class=TableCell style="width: 40%;"> Specify either 0 or 1: 0 JDBC 1 SAS/ACCESS

Attribute	Required	Values	Description
HostName	Optional	String	% " class=TableCell style="width: 40%;"> For the following databases, specify the information indicated: SQLServer Machine name MySQL Machine name Oracle Machine name Microsoft Access Physical file path Microsoft Excel Physical file path Otherwise you can omit this attribute.
PortNumber	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.
ServiceName	Optional	String	% " class=TableCell style="width: 40%;"> For the following databases, specify the information indicated: SQLServer Database name MySQL Database name Oracle Oracle service name where the database instance is running. Otherwise you can omit this attribute.

Attribute	Required	Values	Description
UserName	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.
Password	Optional	String	% " class=TableCell style="width: 40%;"> You must specify this attribute for any of the following: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute. Note: The password, if specified, is encrypted.
AdvanceOptions	%>Optional	String	% " class=TableCell style="width: 40%;"> Specify optional parameters for the following databases: • SQLServer • Oracle • MySQL Otherwise you can omit this attribute.

StagingTable Element (Required)

This element maps a SAS Activity-Based Management model table to one of the tables in the source database or XML file.

When you import information from a database or XML file, all of the tables in the database must exist, but the following tables must contain information (other tables can be empty):

- Period
- Scenario

- Dimension
- DimensionOrder

You must define at least one dimension for each of these tables or the import will fail or an empty table will be created.

When you use the SAS | ABM Web Services Integration API to import information into a model, the XML import configuration can provide any StagingTable element as long as model items exist before they are referenced. For example, before importing an account, the period, scenario, driver, and dimension members that the account references must exist.

The DimensionOrder table is required when you create a model. When a model is created, it uses the information in the DimensionOrder table to determine which dimensions define the hierarchies in each module. If you are not creating a model, this table is ignored.

A StagingArea element can have one or more StagingTable elements. If there is more than one StagingTable element, each one must have a different value for the TableName attribute.

You can import Column elements and DimensionSignature elements from different source database tables; for example, you can import some accounts from one table and other accounts from a different table.

If a model table requires either a reference or dimension signature, and if both are supplied, then the dimension signature is used to identify an account.

Attribute	Required	Values	Description
Name	Required	String	Name of the model table to import into
TableName	Optional if it is the same as Name	String	Name of the table in the source database or XML file that has the information to import into a model

Column Element (Optional)

This element maps a field in a SAS Activity-Based Management model table to one of the fields in the source database or XML file. A StagingTable element can have one or more Column elements.

Attribute	Required	Values	Description
Name	Required	String	Name of the model field to import into
ColumnName	Optional if it is the same as Name	String	Name of the field in the source database or XML file that has the information to import into a model
DefaultVal	Optional	String	Default value

Related Topics

Building a model by importing

Example XML to Import a Model

XML to Publish Period/Scenario Associations

Sample XML

```
<PublishPeriodScenarios >
   <ModelContext  ModelFullName="peAPI" >
      <PeriodScenario PeriodId="9" ScenarioId="1" Publish="true" />
   </ModelContext>
</PublishPeriodScenarios>

<PublishPeriodScenarios >
   <ModelContext ModelFullName="peAPI">
      <PeriodScenario PeriodRef="2008 Q1" ScenarioRef="Actual" Publish="false" />
    </ModelContext>
</PublishPeriodScenarios>
```

Overview

When you use the SAS Activity-Based Management Web Services Integration API to publish period/scenario associations you must write XML to specify the associations to be published.

Example XML to Publish Period/Scenario Associations

XML syntax

The following structure shows the elements used in the XML to publish period/scenario associations and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

Elements and attributes are case sensitive. String comparisons are not case sensitive.

```
<PublishPeriodScenarios >
    <ModelContext attributes>
        <PeriodScenario attributes>
    </ModelContext>
            </ModelContext>
</PublishPeriodScenarios>
```

PublishPeriodScenarios element (required)

PublishPeriodScenarios is the root element. It has no attributes. A PublishPeriodScenarios element can have one ModelContext element.

ModelContext element (required)

This element specifies the model whose period/scenario associations are to be published. A ModelContext element can have multipel PeriodScenario elements.

Attribute	Required	Values	Description
ModeFullName	Required	String	Full Workspace path of the model, excluding the Models Workspace folder

PeriodScenario element (required)

Each PeriodScenario element specifies a period/scenario association one period and one scenario. You can have multiple PeriodScenario elements.

Attribute	Required	Values	Description
PeriodId | PeriodRef	Required	Numeric if ID, String if Reference	Period
ScenarioId | ScenarioRef	Required	Numeric if ID, String if Reference	Scenario

XML to Publish SPM Metrics

Overview

When you use the SAS Activity-Based Management Web Services Integration API to publish a model to SAS Strategy Management, you must write an XML publish SAS Strategy Management metrics configuration that specifies login information for the SAS Strategy Management server.

String comparisons are not case sensitive.

XML publish SAS Strategy Management Metrics configuration structure

The following structure shows the elements used in the XML publish SAS Strategy Management metrics configuration and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<OROSCOMMAND>
    <MODELCONTEXT attributes />
```

```
        <COMMANDPARAMS attributes />
</OROSCOMMAND>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.

OROSCOMMAND element (required)

OROSCOMMAND is the root element. This element has no attributes. It must contain one MODELCONTEXT element.

MODELCONTEXT element (required)

This element specifies the model and the model's period/scenario associations to validate. An OROSCOMMAND element can have one MODELCONTEXT element.

Attribute	Required	Values	Description
ModelId	Required	String	ID of the model
PeriodId	Required	String	ID of the period
ScenarioId	Required	String	ID of the scenario

COMMANDPARAMS element (required)

This element configures the publish operation. An OROSCOMMAND element can have one COMMANDPARAMS element.

Attribute	Required	Values	Description
Username	Required	String	SAS Strategy Management user with sufficient privileges to perform the operations in the program
Password	Required	String	Plain text password for the SAS Strategy Management user

XML to Validate a Model

Overview

When you use the SAS Activity-Based Management Web Services Integration API to validate a model, you must write an XML validate model configuration that specifies what you want to validate.

String comparisons are not case sensitive.

XML validate model configuration structure

The following structure shows the elements used in the XML validate model configuration and their relationships to each other. Position the mouse over a linked element to see if it is required or optional. Click a link for a complete description of the element.

```
<OROSCOMMAND>
    <MODELCONTEXT attributes />
        <COMMANDPARAMS attributes />
</OROSCOMMAND>
```

Note: Although an element might be optional, if you include the element, there might be required elements within it.

OROSCOMMAND element (required)

OROSCOMMAND is the root element. This element has no attributes. It must contain one MODELCONTEXT element.

MODELCONTEXT element (required)

This element specifies the model and the model's period/scenario associations to validate. An OROSCOMMAND element can have one MODELCONTEXT element.

Attribute	Required	Values	Description
ModelId	Required	String	ID of the model
PeriodId	Required	String	ID of the period
ScenarioId	Required	String	ID of the scenario

COMMANDPARAMS element (required)

This element configures the validate operation. An OROSCOMMAND element can have one COMMANDPARAMS element.

Attribute	Required	Values	Description
OverdrivenSourceAccount	Optional	"1","0"	Indicates whether the validate operation includes overdriven source accounts
UnassignedAccount	Optional	"1","0"	Indicates whether the validate operation includes unassigned accounts
ZeroCostAccount	Optional	"1","0"	Indicates whether the validate operation includes accounts with zero costs
EmptyAttribute	Optional	"1","0"	Indicates whether the validate operation includes attributes without values

Attribute	Required	Values	Description
NegativeDriverQuantityAssignment	Optional	"1","0"	Indicates whether the validate operation includes accounts with negative driver quantities
MessageLimit	Optional	String	Maximum number of similar error messages to write to the log file; when this limit is reached, no other similar error messages are written; default is 50

Chapter 20
Easy API

Using Easy API

Overview

Using Easy API, you can do in batch many of the same operations that you can do inside SAS Activity-Based Management. With Easy API, you can

- Import and export model data
- Calculate a model
- Generate a cube
- Export report data
- Copy model data from one period/scenario to another
- Import and export cube configurations

In addition, you can use Easy API to run SAS stored processes, an external SAS Enterprise Guide project, or any other executable that you want to invoke. So, for example, you can use Easy API to export model data, invoke a SAS stored process to update the exported data, and finally import the updated data back into your model.

Operations are run in the order specified in your EasyAPI.txt file. Whatever operations you perform, Easy API synchronizes them so that the next operation to run does not begin until the previous one has finished. For example, a SAS program to update exported tables does not run until the tables have been exported.

Invoking Easy API to perform a SAS Activity-Based Management operation involves three steps:

1. “Create an XML File” on page 310

   The XML file describes the operation to be performed.

2. “Save Easy API Commands in a Text File” on page 311

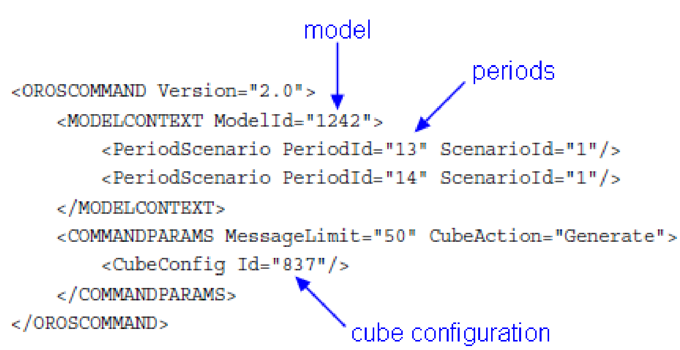
   Easy API commands invoke SAS Activity-Based Management and pass an XML file to tell it what to do.

3. “Invoke Easy API” on page 312

   Easy API uses the text file to run its commands.

Create an XML File

SAS Activity-Based Management uses XML internally to encode the information that it needs for performing operations. Easy API uses the same XML to invoke SAS Activity-Based Management in batch to perform those operations. Following is sample XML to generate a cube. Notice that the XML specifies the model, periods, and cube configuration to be used in generating the cube.



Because Easy API uses exactly the same XML to invoke SAS Activity-Based Management that SAS Activity-Based Management itself uses internally, the easiest way for you to create the XML that you need to run Easy API is to ask SAS Activity-Based Management to create it.

To ask SAS Activity-Based Management to create XML:

1. Inside SAS Activity-Based Management, select **Tools** ⇨ **User Options**.
2. Click the **Easy API Configuration** tab.
3. Select **Save operation xml in directory path**.
4. Specify the directory path where the XML will be saved.

   Now, when you perform an operation inside SAS Activity-Based Management, the XML for that operation is saved in a file in the directory that you specified.

You can modify the XML file to suit your purposes. For example, you might modify the XML file shown here to generate different periods for the same model or to generate the same periods for a different model.

For information on the XML files, see "Using the API" in the *SAS Activity-Based Management Data Administration Guid*e available from the Help menu or at **`http://support.sas.com/documentation/onlinedoc/abm/`**.

Save Easy API Commands in a Text File

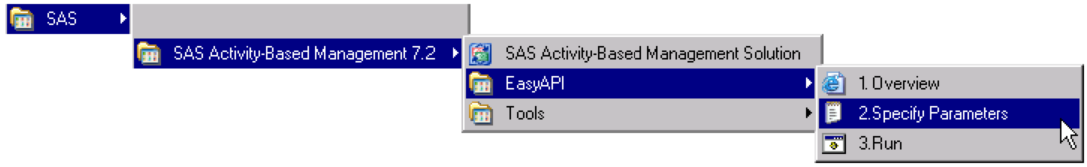
The following table lists the Easy API commands and tells what each command does. Notice that each command takes one parameter which is either the path and name of an XML file, or the path and name of an external program. The parameters shown are only examples given that your path is probably different.

Command and sample argument	What it does
Export "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Export model data
Run "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.sas"	Execute external programs including, but not limited to, SAS stored processes. For example, you can also run SAS Enterprise Guide vbscripts using this Run command.
Import "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Import model data
Calculate "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Calculate and/or generate a cube *Note:* The XML file that you use determines whether this command does a calculation or generates a cube.
Export Report "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Export a report
Copy Period "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Copy model data from one period/scenario to another
Export Cube "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Export cube configurations
Import Cube "C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\your.xml"	Import cube configurations
// Comment	You can put ' or // in front of the command line to comment out (skip) a particular Easy API command.

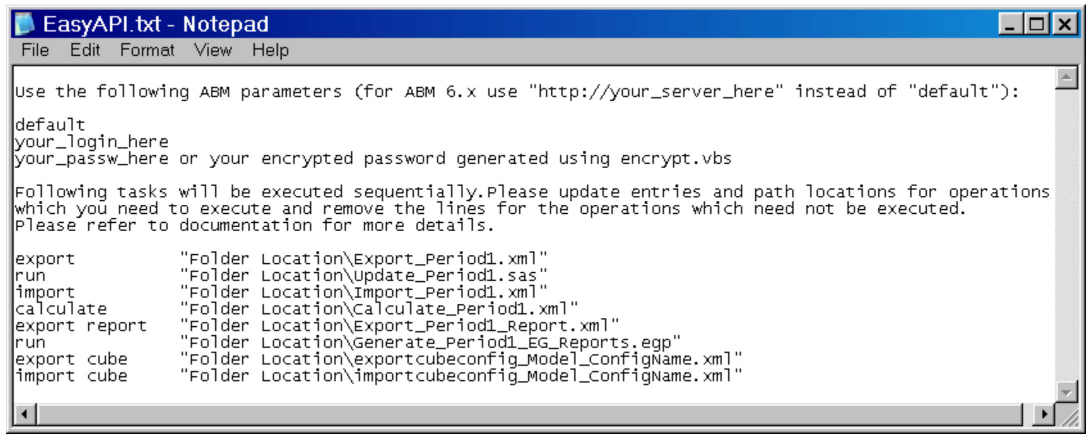
To issue an Easy API command, put it in a text file named EasyAPI.txt residing in the following directory:

```
<installation directory>SASActivityBasedManagementClient\7.2\EasyAPI\
For example: C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\
```

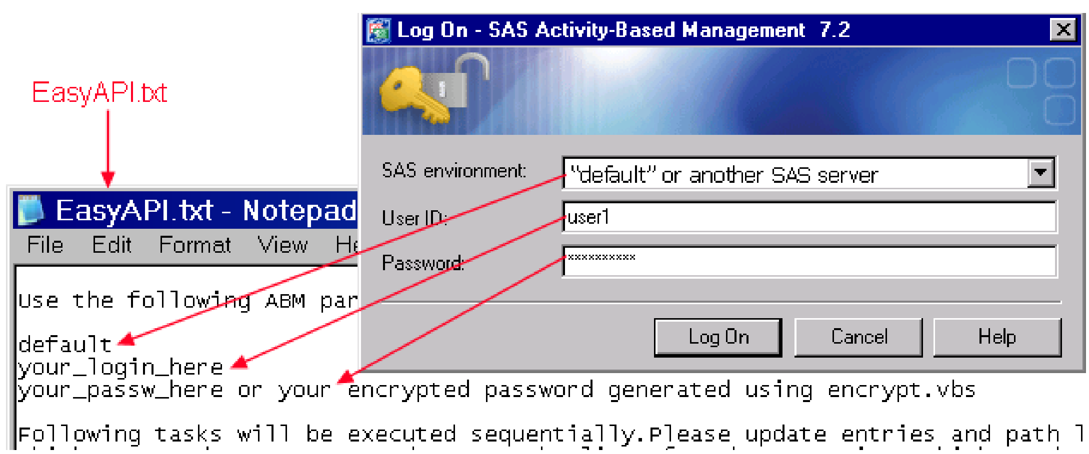
An easy way to open EasyAPI.txt is by selecting **Start** ⇨ **Programs** ⇨ **SAS** ⇨ **SAS Activity-Based Management 7.2** ⇨ **EasyAPI** ⇨ **2. Specify parameters** from the **Start** menu.



The following picture shows EasyAPI.txt as it is appears on installation of SAS Activity-Based Management.

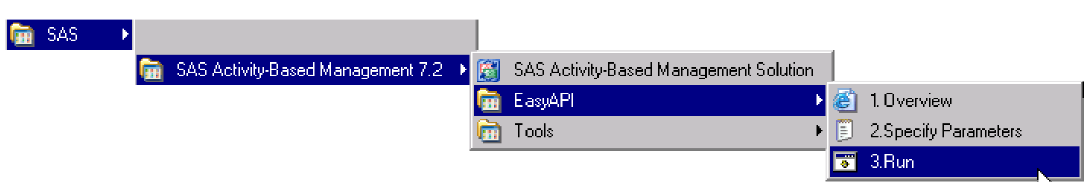


Don't forget to include your login credentials in the txt file.



Invoke Easy API

To invoke Easy API, select **Start** ⇨ **Programs** ⇨ **SAS** ⇨ **SAS Activity-Based Management 7.2** ⇨ **EasyAPI** ⇨ **3. Run** from the **Start** menu.

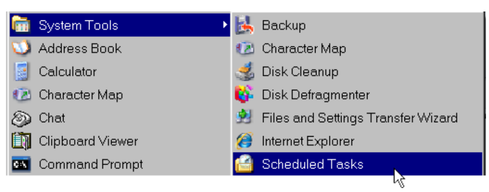


Easy API can e-mail you the results of its operation. To receive an e-mail with operation results:

1. Inside SAS Activity-Based Management, select **Tools** ⇨ **User Options**.
2. Click the **Easy API Configuration** tab.
3. Specify an **SMTP server** for sending the mail.
4. Specify an **Email Id for successful operation**.
5. Specify an **Email Id for failure of operation**.

Notes:

- Log files named EasyAPI.log or CutomEasyAPI.log are created in the Easy API Installation folder. You can also access the Easy API operations log in the Windows Event Viewer.
- You can also invoke Easy API by running EasyAPI.exe, which is installed in **<*installation directory*>SASActivityBasedManagementClient \7.2\EasyAPI**.
- By supplying a path argument to EasyAPI.exe, you can tell it to use a different txt file for Easy API commands, for example, **EasyAPI.exe "c:\MyPath \EasyAPI2.txt"**. If you don't supply a path argument, then Easy API uses EasyAPI.txt in its installation directory.
- You can use the Microsoft Windows Scheduled Tasks Wizard to schedule EasyAPI.exe to run automatically at selected intervals.



- In the EasyAPI.txt file, you can either store your password in clear text or you can encrypt it using EncryptPassword.exe located at **<*installation directory*> \SASActivityBasedManagementClient\7.2\EasyAPI**. EncryptPassword.exe produces an encrypted string that you can paste into EasyAPI.txt. Easy API then decrypts the password before performing Easy API operations.

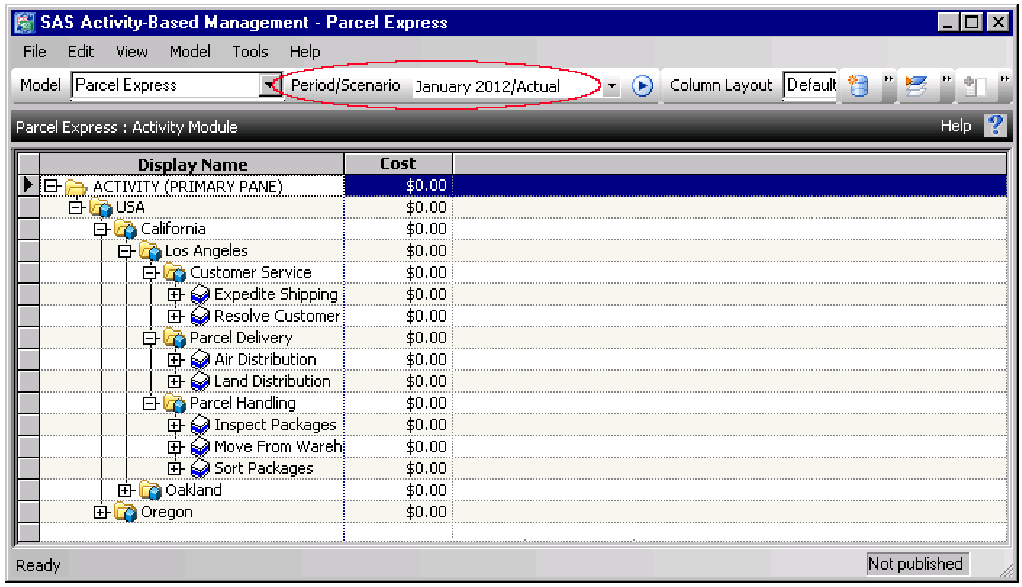
Example of Using Easy API

Overview

This section presents an example of using Easy API to automate performing the following tasks:

1. Export the tables for a particular period
2. Run a SAS job to update the tables
3. Re-import the updated tables back into the model
4. Calculate the model
5. Export report data
6. Generate a SAS Enterprise Guide report

Notice in the following picture of a sample model that the cost is $0 for January activities because the model hasn't been updated with data for the month. So, in this example, we export January tables, update them, and re-import them back into the model for calculation. The example shows how to automate all of these tasks using Easy API.



Export January Tables

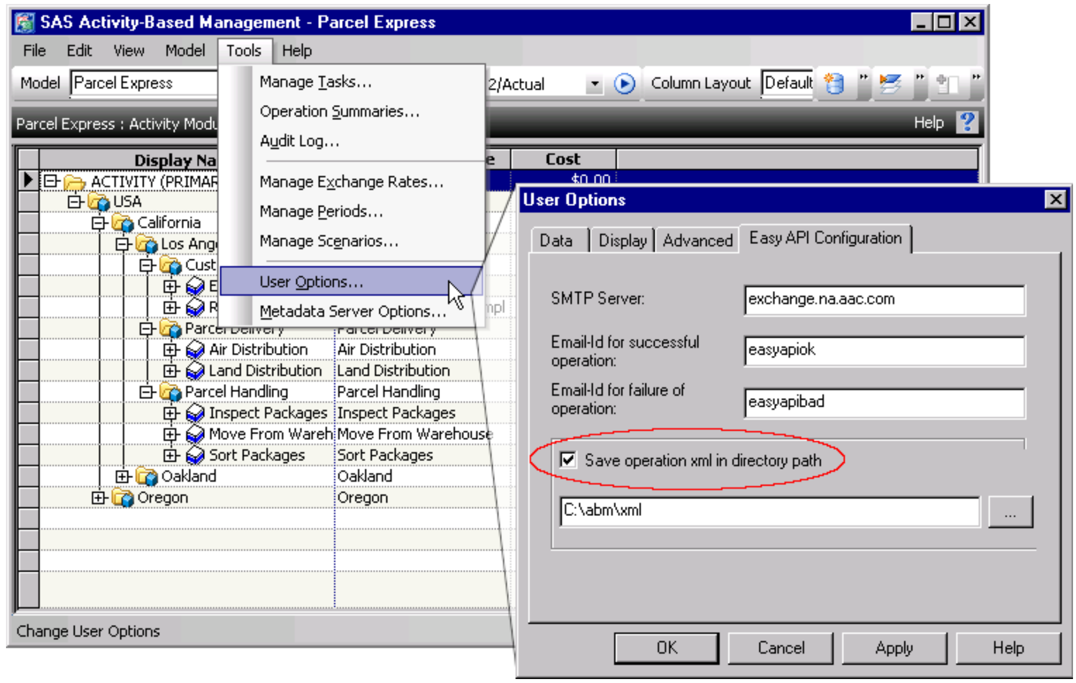
Turn on Writing of XML

Before exporting the January tables, we must do one thing—ask SAS Activity-Based Management to write to a file the XML that it uses to perform operations. Because Easy API uses exactly the same XML to invoke SAS Activity-Based Management that SAS Activity-Based Management itself uses internally, the easiest way to create the XML that you need to run Easy API is to ask SAS Activity-Based Management to create it.

To ask SAS Activity-Based Management to create XML:

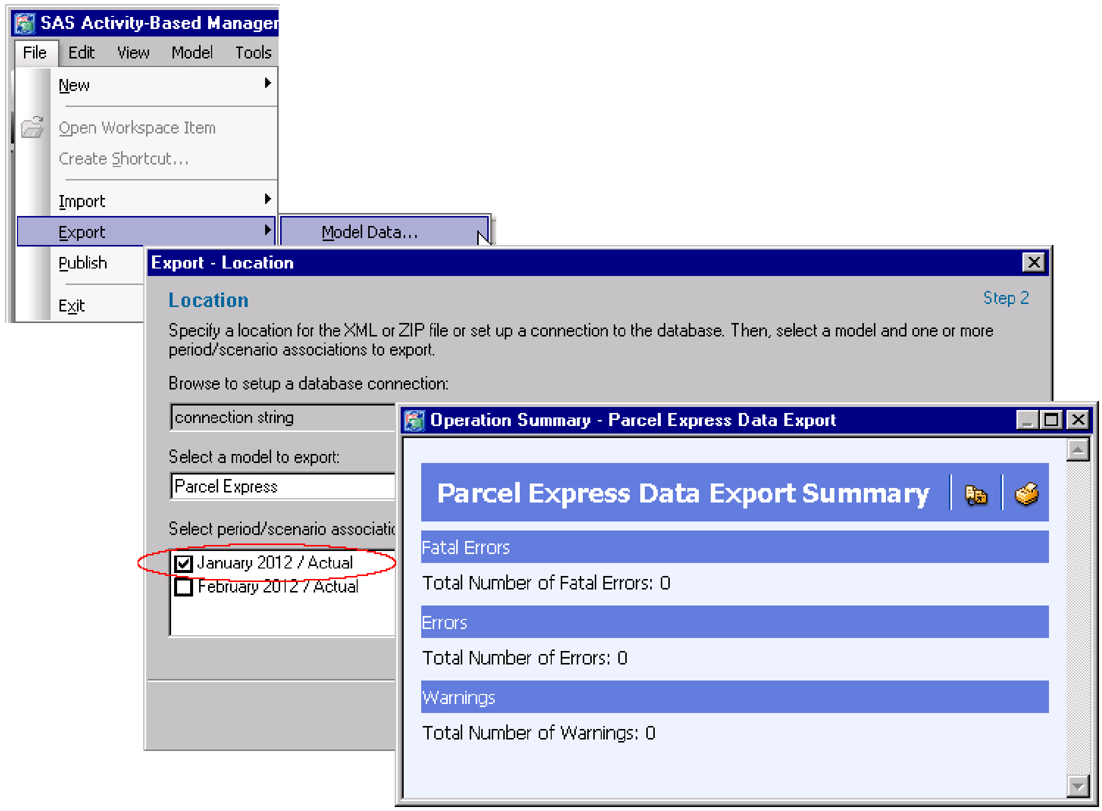
1. Inside SAS Activity-Based Management, select **Tools** ⇨ **User Options**.
2. Click the **Easy API Configuration** tab.
3. Select **Save operation xml in directory path**.
4. Specify the directory path where the XML will be saved.

   Now, when you perform an operation inside SAS Activity-Based Management, the XML for that operation is saved in a file in the directory that you specify.

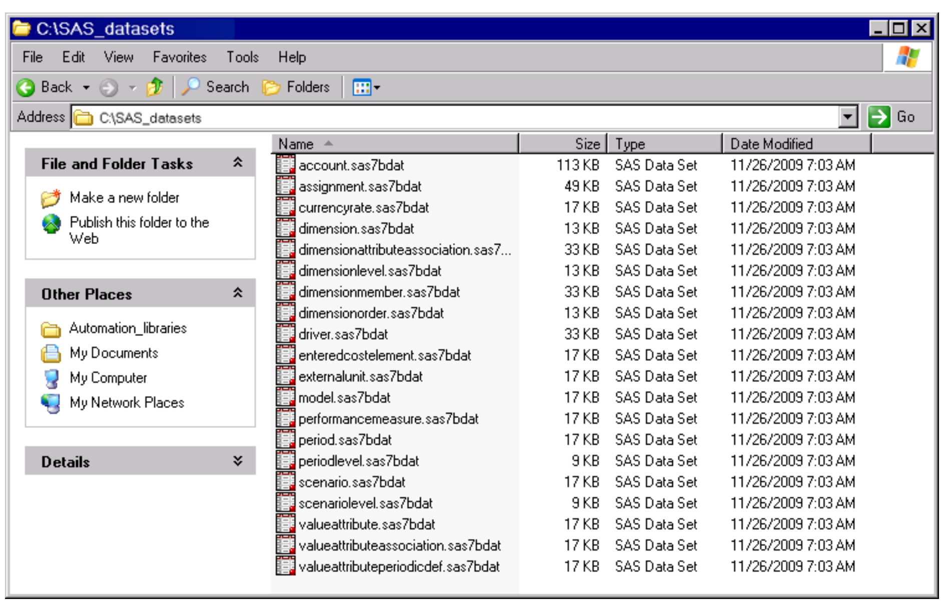


Export Tables

Now we are ready to export the January tables. We don't show the export in any detail. Note, however, that the export has completed with no errors. It is important that you can successfully do inside SAS Activity-Based Management whatever task it is that you want to automate in batch with Easy API. Because Easy API calls SAS Activity-Based Management to perform tasks, if you can't perform the task inside SAS Activity-Based Management, then you won't be able to perform the same task outside SAS Activity-Based Management using Easy API.



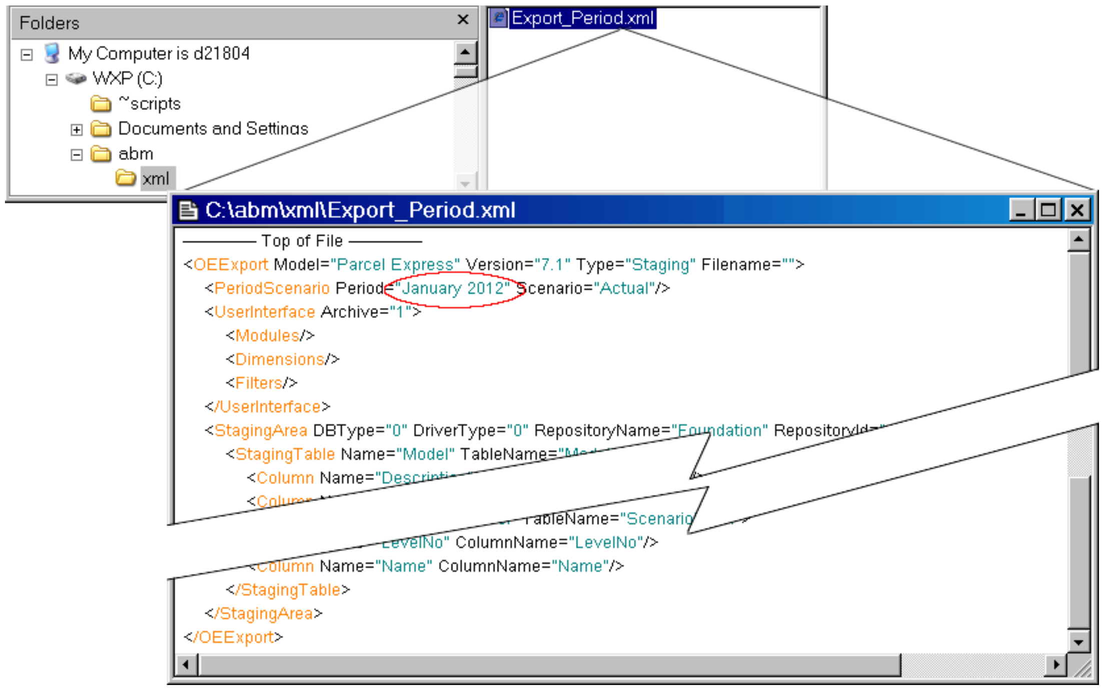
Because the export was successful, it produced the SAS tables shown in the following picture. Although this example shows SAS tables, you can use Easy API to automate exporting any type of database tables that you want.



Modify the Export XML

Because we asked SAS Activity-Based Management to write to a file the XML that it uses to perform operations, you can see in the following picture the XML file that it produced during the export. The filename, however, has been changed for the sake of this example.

Notice the parameter that calls for exporting the January table. If you want to export the February table next when the export task is performed in batch using Easy API, then you can modify this XML to specify February instead.



Automate the Export for Next Time

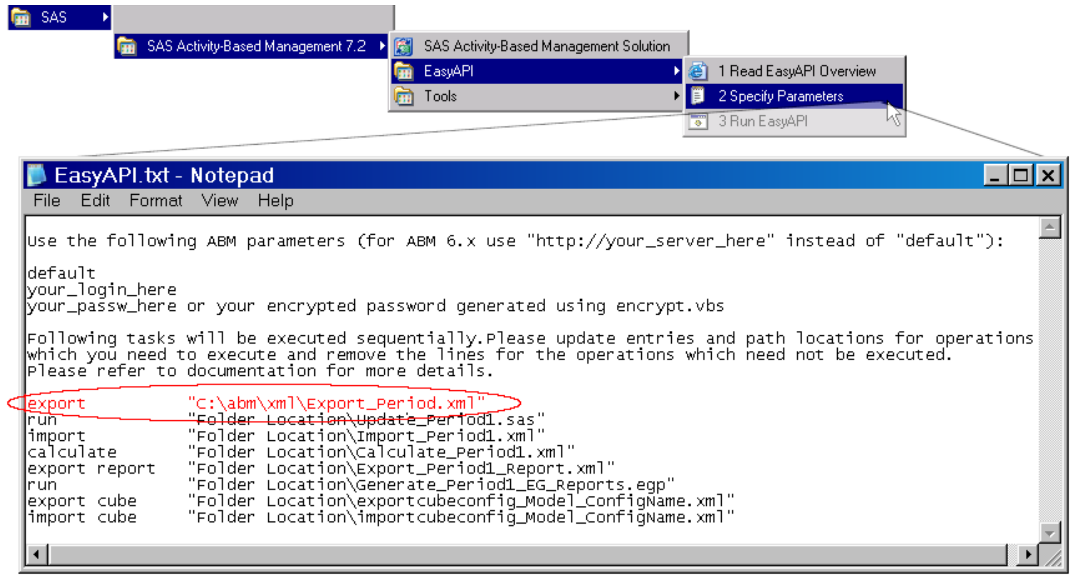
Having modified the XML file to export the period that you want, you next need to point Easy API to the modified file. Easy API uses a text file, EasyAPI.txt, for its commands. Each command takes one argument, which is usually the name and path of the XML file to be passed to SAS Activity-Based Management for performing an operation. For a list of Easy API commands, see "Save Easy API Commands in a Text File" on page 311.

EasyAPI.txt is installed in the following directory:

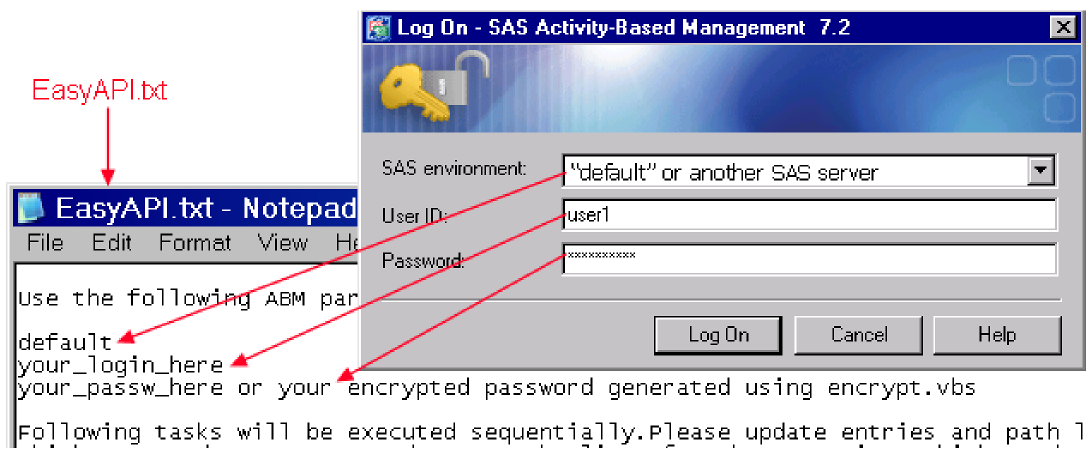
```
<installation directory>SASActivityBasedManagementClient\7.2\EasyAPI\
For example: C:\Program Files\SASHome\SASActivityBasedManagementClient\7.2\EasyAPI\
```

To open EasyAPI.txt, select **Start** ⇨ **Programs** ⇨ **SAS** ⇨ **SAS Activity-Based Management 7.2** ⇨ **Easy API** ⇨ **2. Specify parameters** from the **Start** menu.

The following picture shows EasyAPI.txt with its first command line (export) modified to pass the XML file "c:\abm\xml\Export_Period.xml".



Don't forget to include your login credentials in the EasyAPI.txt file.

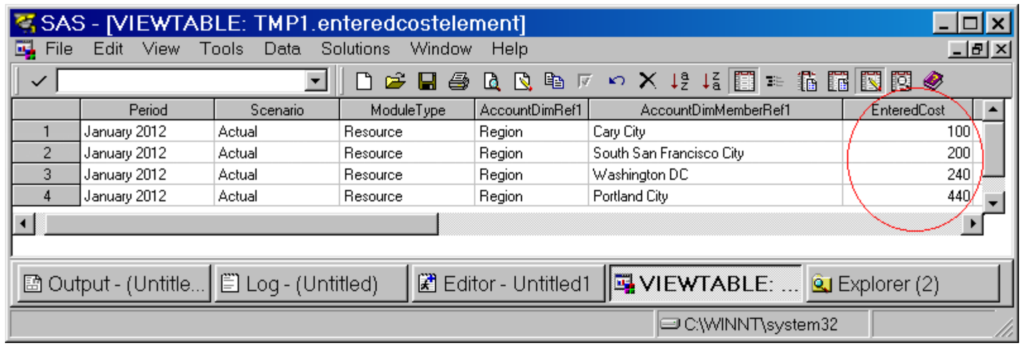


Run a SAS Job to Update the Tables

Run the Job

Having exported the January tables, your next task is to run a SAS job to update the tables. We don't show any details of the SAS job, and you can run any program that you want to update the tables. Regardless of what program you use, you can automate its running with Easy API.

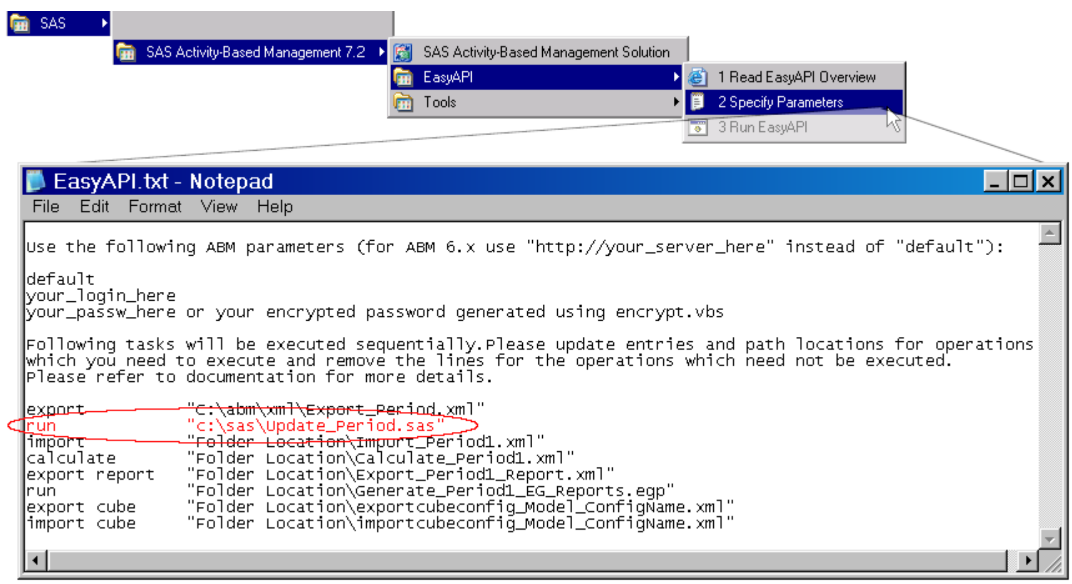
The following picture shows one of the tables after it was updated with entered costs for January.



Automate the SAS Job for Next Time

To automate running the job, all you have to do is modify EasyAPI.txt again to invoke the desired program. The following picture shows the second command line changed to the following:

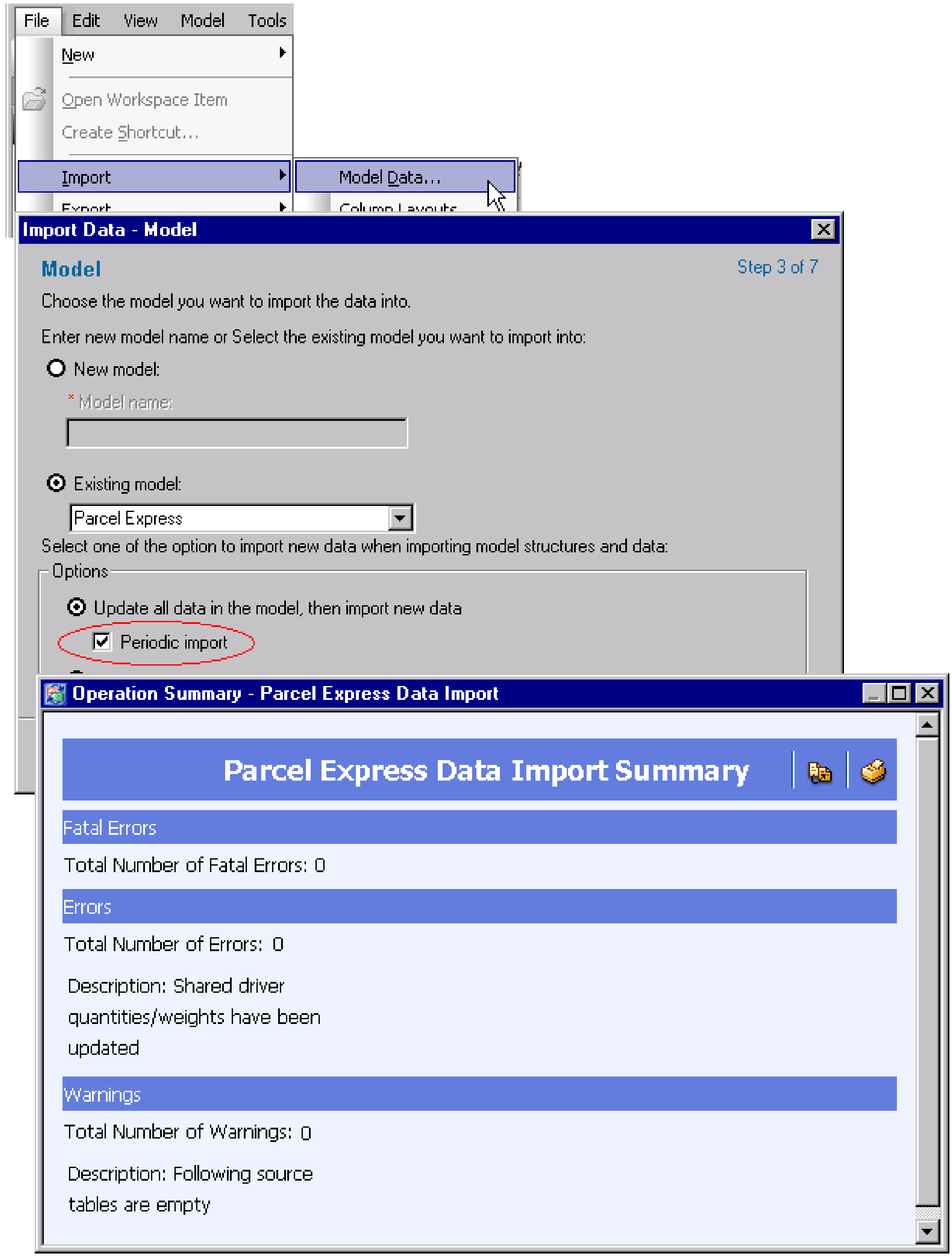
```
run "c:\sas\Update_Period.sas"
```



Re-import the Tables

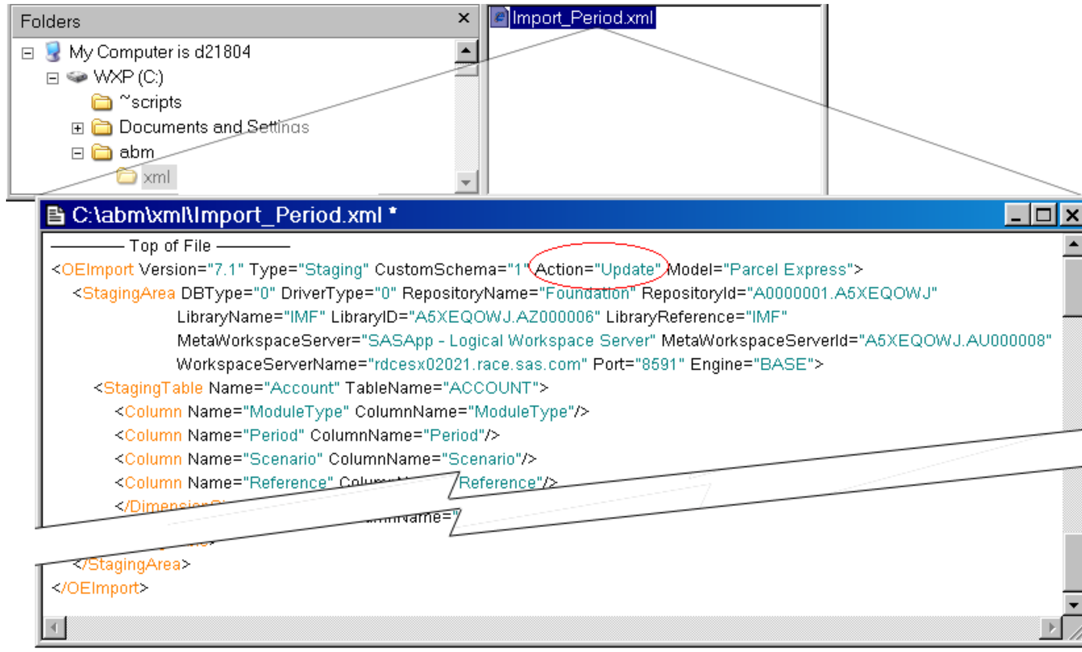
Re-import the Tables

Now that the tables have been updated, you can re-import them back into the model. In the following picture, note that we have selected **Periodic import**. This option causes only those staging tables that contain periodic data to be displayed in the Import Wizard for you to select for importing. (Non-periodic data does not change from one period to the next and, so, does not have to be updated and re-imported).



Modify the Import XML

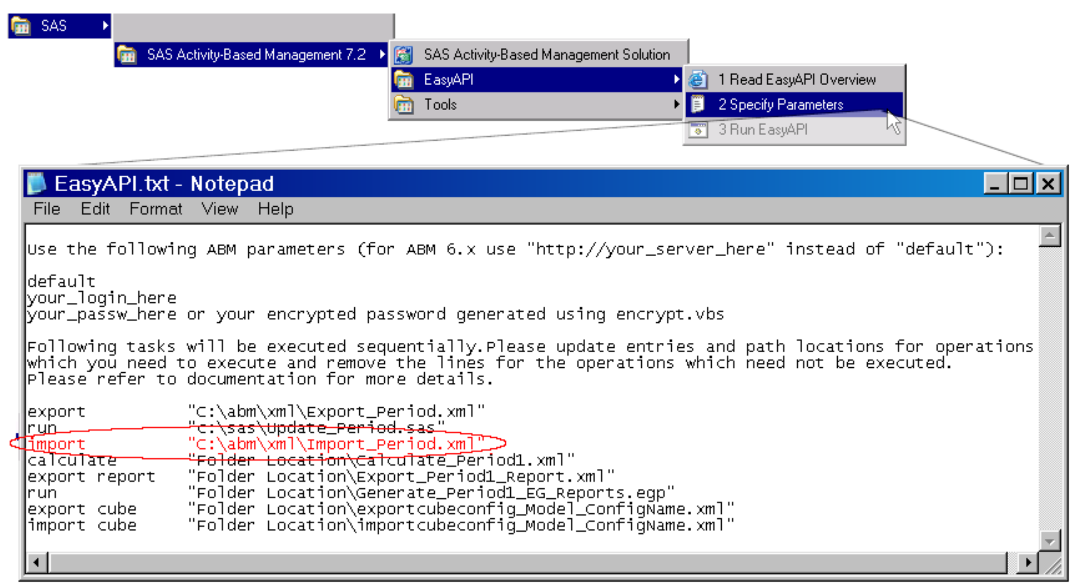
Once again, SAS Activity-Based Management has writen to a file the XML that it used for performing the import. Notice that the Activity= parameter specifies Update. That means that the model is updated rather than a new model being created. Also notice that the file does not have a Period= parameter. The period information (Janauary, in this case) is contained inside the tables being imported rather than inside the XML file. That means that you do not have to modify this XML file at all to automate the import with Easy API. You do, however, have to let Easy API know where the file is located, which we do next.



Automate the Import for Next Time

To make Easy API perform the import, once again you must modify EasyAPI.txt. Notice in the following picture that the third command line has been changed to specify

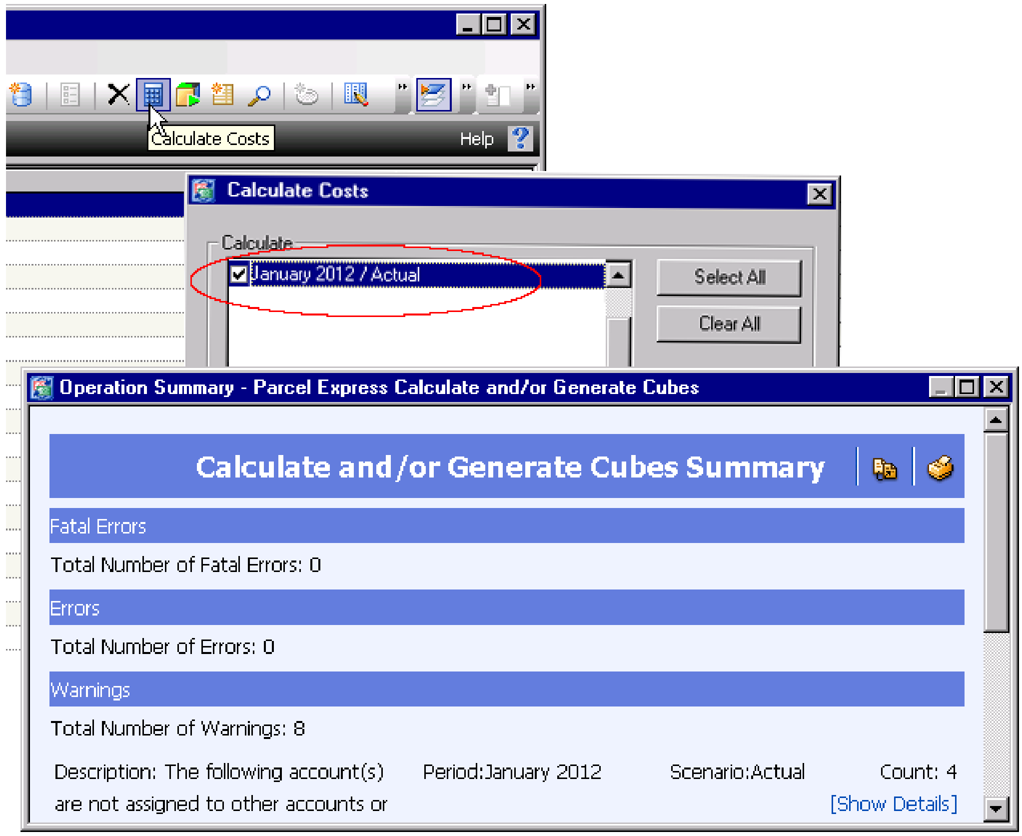
```
Import "c:\abm\xml\Import_Period.xml"
```



Calculate the Model

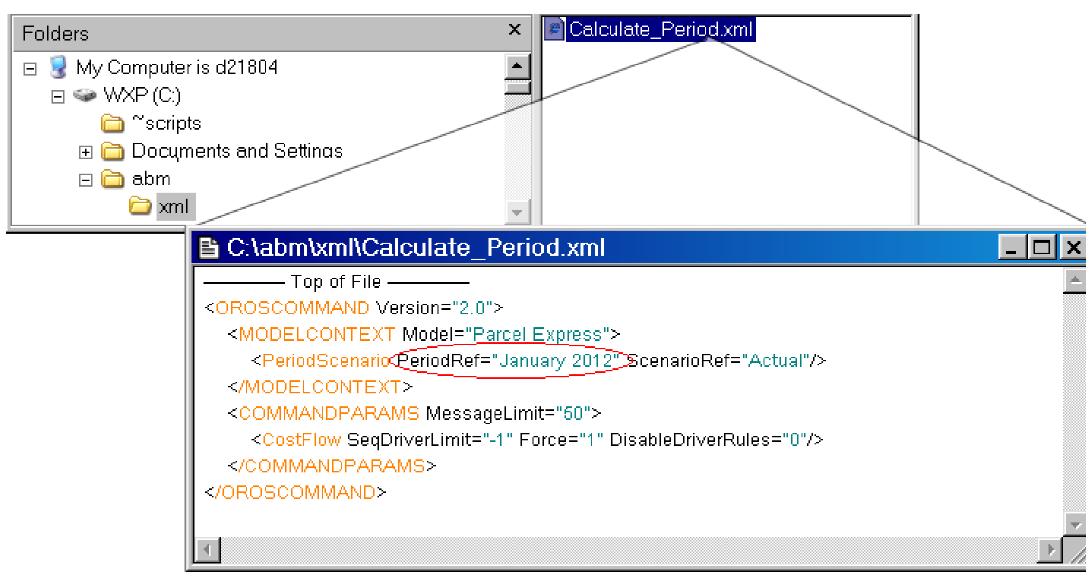
Calculate the Model

Once the updated tables have been re-imported back into the model, the model must be calculated. Assuming that the other periods have already been calculated, it is necessary to calculate only January.



Modify the Calculate XML

Looking at the XML that SAS Activity-Based Management produced for the calculation, you can see that to use it with Easy API, you only need to modify the XML to specify the period to be calculated.



Note: Suppose the calculation produces XML like the following which uses IDs.

```
<OROSCOMMAND Version="2.0">
    <MODELCONTEXT ModelId="1242">
        <PeriodScenario PeriodId="103" ScenarioId="1"/>
```

```
        </MODELCONTEXT>
        <COMMANDPARAMS MessageLimit="50">
            <CostFlow SeqDriverLimit="-1" Force="1" DisableDriverRules="0"/>
        </COMMANDPARAMS>
    </OROSCOMMAND>
```

Then you can change the XML to use references instead:

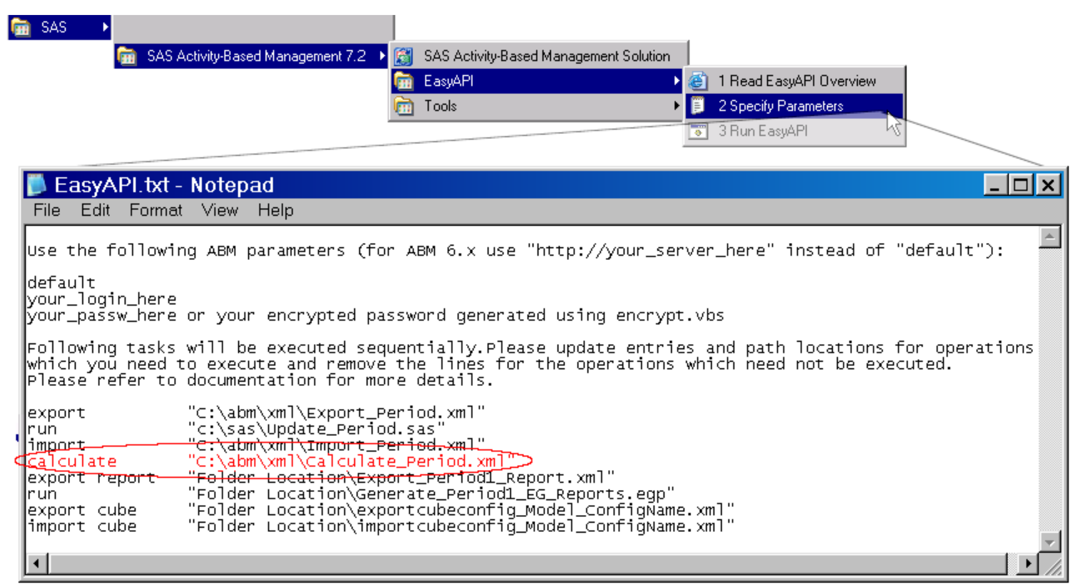
```
<OROSCOMMAND Version="2.0">
    <MODELCONTEXT Model="Parcel Express">
        <PeriodScenario PeriodRef="January 2012" ScenarioRef="Actual"/>
    </MODELCONTEXT>
    <COMMANDPARAMS MessageLimit="50">
        <CostFlow SeqDriverLimit="-1" Force="1" DisableDriverRules="0"/>
    </COMMANDPARAMS>
</OROSCOMMAND>
```

Note: Instead of specifying a period to calculate, you can select all periods by specifying **`PeriodID="0"`**.

Automate the Calculate for Next Time

Once again, after modifying the XML file to calculate the period you want, you need to tell Easy API where the file is located. In the following picture, you can see that the calculate command has been changed to

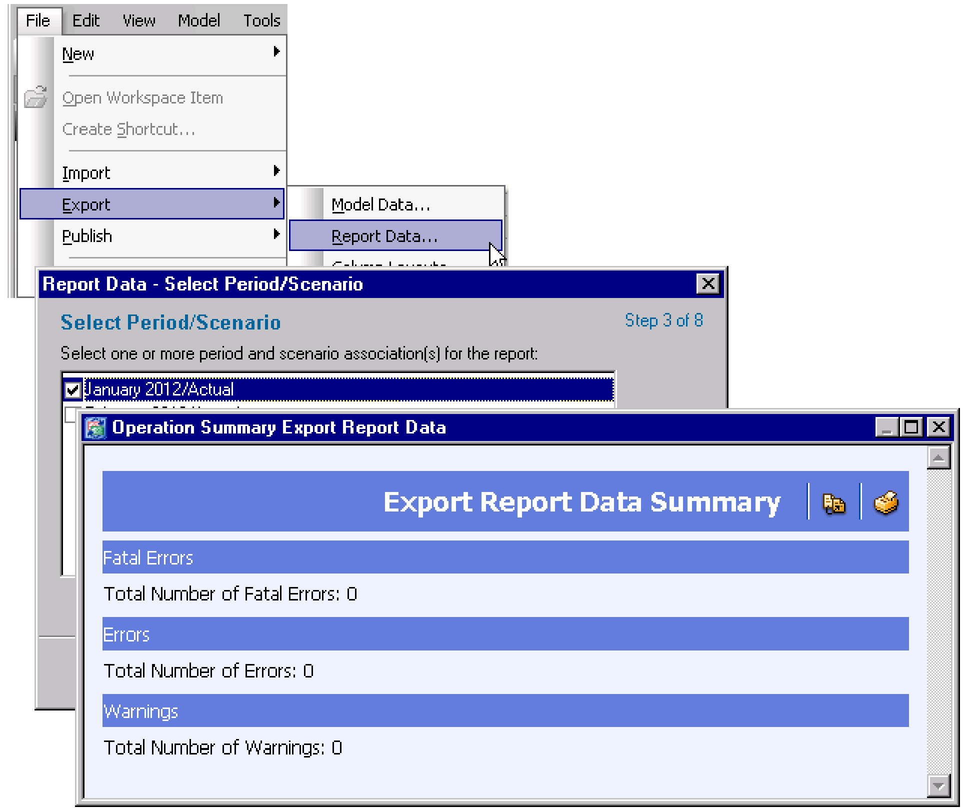
```
Calculate "c:\abm\xml\Calculate_Period.xml"
```



Export Report Data

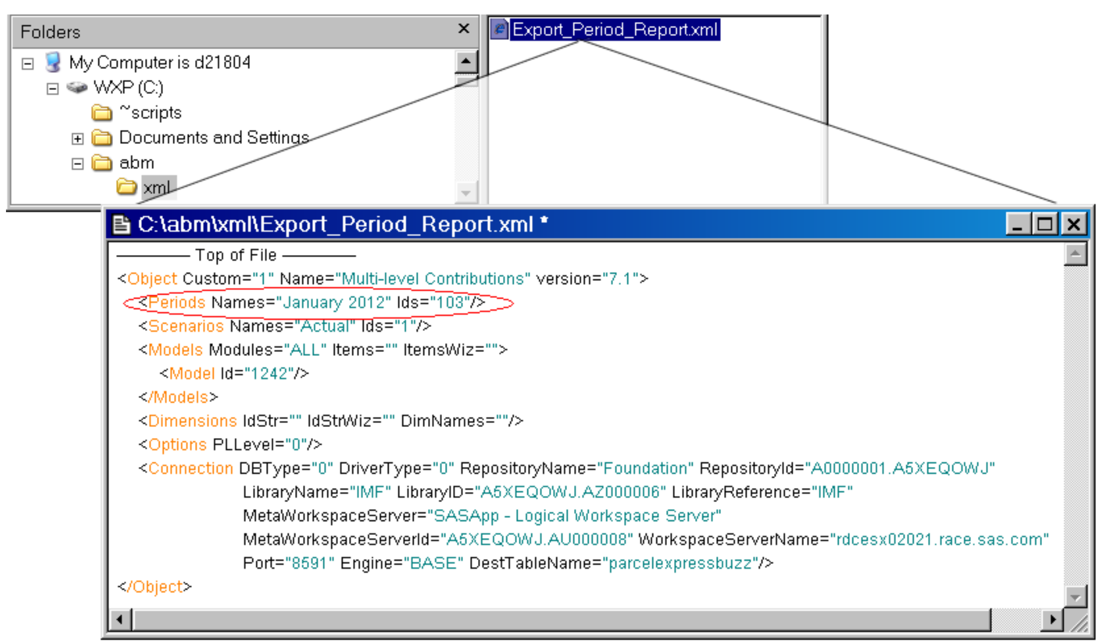
Export Report Data

After the model is calculated, you might want to export report data and, for example, produce a report using SAS Enterprise Guide.



Modify the Export Report XML

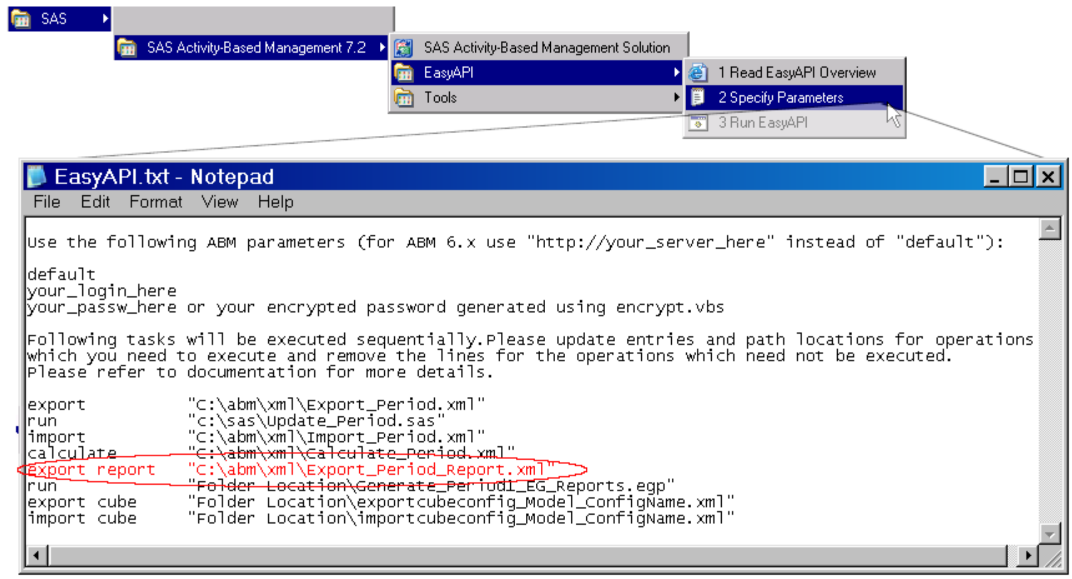
Looking at the XML that is produced by the export, you can see that you only need to modify the XML to specify the period to be exported.



Automate the Export Report for Next Time

And, again, after you have modified the XML file, you need to tell Easy API where the file is located. In the following picture, you can see that the export report command has been changed to

```
Export report "c:\abm\xml\Export_Period.xml"
```

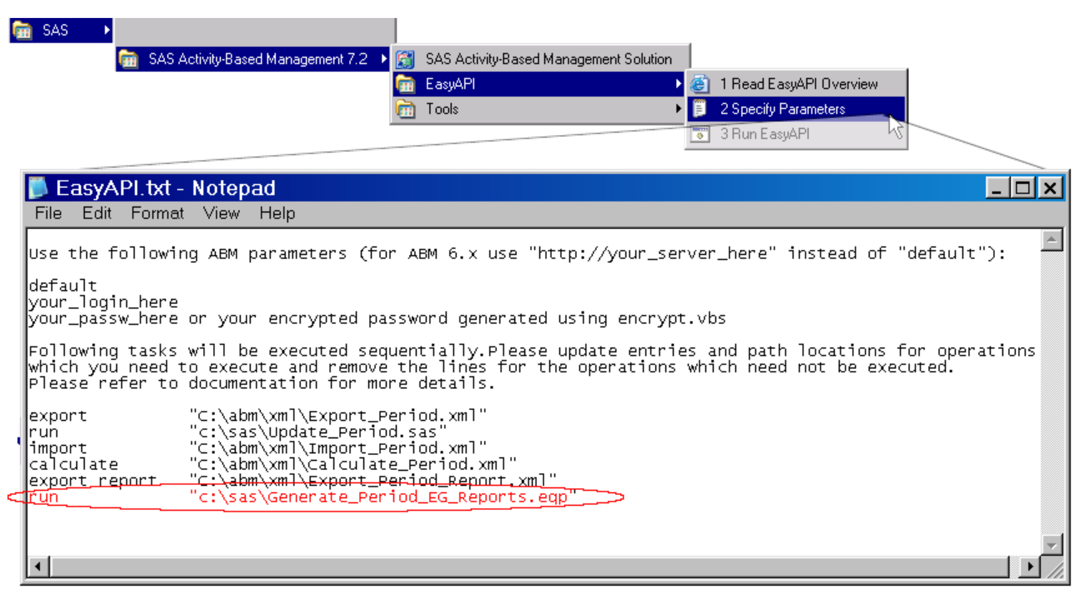


Generate a Report Using SAS Enterprise Guide

We don't show how to code a SAS Enterprise Guide project to produce a report. To automate the job using Easy API, you only need to modify EasyAPI.txt to run it. In the following picture, the run command has been changed to

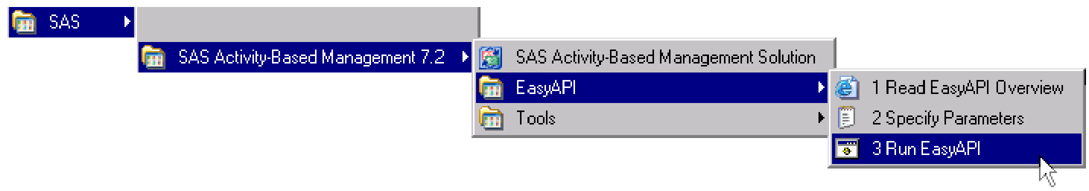
```
run "c:\sas\Generate_Period_EG_Reports.egp"
```

You can also see that the two lines following this command have been deleted because they are not needed.



Run Easy API

Now that EasyAPI.txt has been modified and all the XML files are in place, we are ready to run a batch job. To run Easy API, select **Start** ⇨ **Programs** ⇨ **SAS** ⇨ **SAS Activity-Based Management 7.2** ⇨ **Easy API** ⇨ **3. Run** from the **Start** menu.



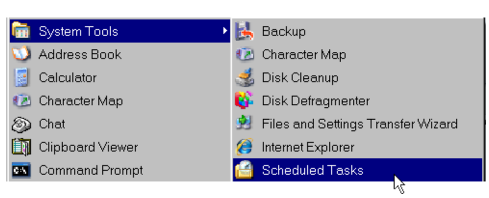
Easy API synchronizes all the operations so that one operation does not begin until the operation before it has completed. So, for example, the program to update exported tables is not launched until the export has completed. And the operation to re-import the updated tables does not begin until the update operation has completed.

Easy API can e-mail you the results of its operations. To receive an e-mail with operation results:

1. Inside SAS Activity-Based Management, select **Tools** ⇨ **User Options**.
2. Click the **Easy API Configuration** tab.
3. Specify an **SMTP server** for sending the mail.
4. Specify an **Email Id for successful operation**.
5. Specify an **Email Id for failure of operation**.

Notes:

- You can also invoke Easy API by running EasyAPI.exe, which is installed in <*installation directory*>**`SASActivityBasedManagementClient \7.2\EasyAPI\`**
- By supplying a path argument to EasyAPI.exe, you can tell it to use a different txt file for Easy API commands, for example, **`EasyAPI.exe "c:\MyPath \EasyAPI2.txt"`**. If you don't supply a path argument, then Easy API uses EasyAPI.txt in its installation directory.
- You can use the Microsoft Windows Scheduled Tasks Wizard to schedule EasyAPI.exe to run automatically at selected intervals.



Part 6

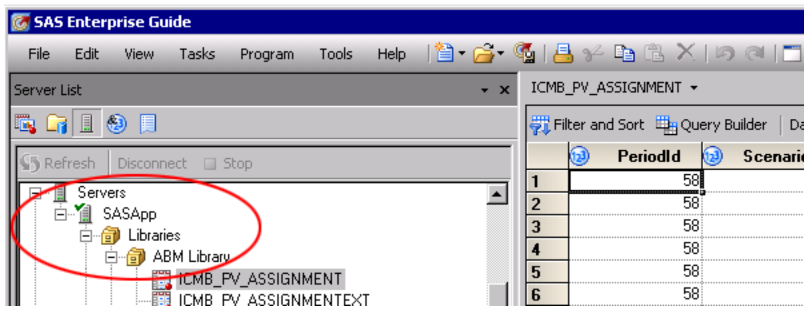
Public Views

Chapter 21

Public Views

Overview

SAS Activity-Based Management creates public views for each model in the database. As an example, using SAS Enterprise Guide, expand **Servers** ⇨ **SASApp** ⇨ **Libraries** ⇨ **ABM Library**. You can see a list of the public views. The following picture shows opening the Assignment view for the ICMB model. The table name is ICMB_PV_ASSIGNMENT because ICMB is the model reference.



To open the public views using Microsoft SQL Server Enterprise Manager, open the **Databases** folder and expand the ABM Models database. Under **ABM Models**, double-click the **Views** icon to display the database views for the various models in the right pane. Right-click the **PublicModel** view and select **Open View** ⇨ **Return all rows** to display the table view that maps a model ID to a model name.

Listed below are the four non-model-specific views followed by the model-specific views. Notice that some of these views are automatically created when the model is created, while others are created after you calculate the model.

Non-model-specific view tables:

- "PublicModel " on page 332
- "PublicModelStatus " on page 333
- "PublicPeriod " on page 334
- "PublicScenario " on page 334

Model-specific view tables created when the model is created:

- "<modelRef>_PV_Assignment " on page 338
- "<modelRef>_PV_AssignmentExt " on page 340
- "<modelRef>_PV_Attribute" on page 341
- "<modelRef>_PV_AttributeValue " on page 342
- "<modelRef>_PV_Dimension " on page 343
- "<modelRef>_PV_DimMember " on page 343
- "<modelRef>_PV_Driver " on page 344
- "<modelRef>_PV_EnteredCE " on page 345

Model-specific view tables created by calculating the model and selecting **Calculate Costs**:

- "<modelRef>_PD_<DimRef> " on page 346

- “<modelRef>_PD_CostElement ” on page 346
- “<modelRef>_PD_Driver ” on page 347
- “<modelRef>_PD_Module ” on page 348
- “<modelRef>_PD_Period ” on page 348
- “<modelRef>_PD_Scenario ” on page 349
- “<modelRef>_PD_YesNo ” on page 350
- “<modelRef>_PV_Account ” on page 335

Model Specific view tables created by selecting the option **Fact table only** when generating a model:

- “<modelRef>_PF_MSC ” on page 350
- “<modelRef>_PF_RC ” on page 351 .
- “<modelRef>_PF_SSC ” on page 352
- “<modelRef>_PF_<cubeConfigRef> ” on page 353

These public view names can be found in the SAS Activity-Based Management database and are automatically provided for every model in SAS Activity-Based Management. The tables in this section (arranged by public view name) describe the schema for each of the public views. For individual properties, see "Properties List Alphabetically" in the *SAS Activity Based Management: User's Guide*

The following table shows the name of legacy public views and the name of the corresponding public view that uses short references rather than IDs. The legacy public views are supported for SAS Activity-Based Management 7.2, but they will not be supported in subsequent releases.

Legacy Public View	New Public View
M<*modelID*>_PublicAccount	<*modelRef*>_PV_Account
M<*modelID*>_PublicAssignment	<*modelRef*>_PV_Assignment
M<*modelID*>_PublicAssignmentExt	<*modelRef*>_PV_AssignmentExt
did not exist before	<*modelRef*>_PV_Attributes
M<*modelID*>_PublicAttributeValue	<*modelRef*>_PV_AttributeValue
M<*modelID*>_PublicDimension	<*modelRef*>_PV_Dimension
M<*modelID*>_PublicDimMember	<*modelRef*>_PV_DimMember
M<*modelID*>_PublicDriver	<*modelRef*>_PV_Driver
M<*modelID*>_PublicEnteredCE	<*modelRef*>_PV_EnteredCE
M<*modelID*>_PublicDim<*dimID*>	<*modelRef*>_PD_<*dimRef*>
M<*modelID*>_PublicDimCostElement	<*modelRef*>_PD_CostElement
M<*modelID*>_PublicDimDriver	<*modelRef*>_PD_Driver
M<*modelID*>_PublicDimModule	<*modelRef*>_PD_Module

Legacy Public View	New Public View
M<*modelID*>_PublicDimPeriod	<*modelRef*>_PD_Period
M<*modelID*>_PublicDimScenario	<*modelRef*>_PD_Scenario
M<*modelID*>_PublicDimYesNo	<*modelRef*>_PD_YesNo
M<*modelID*>_PublicFactMSC	<*modelRef*>_PF_MSC
M<*modelID*>_PublicFactRC	<*modelRef*>_PF_RC
M<*modelID*>_PublicFactSSC	<*modelRef*>_PF_SSC
did not exist before	<*modelRef*>_PF_<*cubeConfigRef*>

Table Column Descriptions

Field Name
The field name that is provided in the public view.

Field Type
The definition of each field type (numeric or text field).

Join Field
The specific ID fields to join between tables. These fields define keys that can be used to link multiple tables for reporting. This column identifies the corresponding join field.

Properties Reference
The specific corresponding property. See "Properties List Alphabetically" in the *SAS Activity Based Management: User's Guide*.

Model View Columns
The specific property that is displayed in the model column view.

PublicModel

Table 21.1 *Model View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
Name	Text			Model

The following is a sample PublicModel:

PUBLICMODEL

	ID	NAME
1	1002	AM2
2	1009	Test_AM2
3	1011	FirstE2EScenario
4	1012	FourthSceETOE
5	1000	OLAPView_TC01
6	1001	CreateNewM
7	1018	FifthE2EScenario
8	1019	SecondE2EScen...
9	1020	yubope
10	1021	pecopy
11	1022	Parcel Express b...
12	1023	pe_murali

PublicModelStatus

Table 21.2 *ModelStatus View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
CalculationStatus	Text			Calculation status not viewable in the user interface
CubeStatus	Text			Cube status not viewable in the user interface
ModelID	Number			ID values not viewable in the user interface
Period	Text		Name	Name
Scenario	Text		Name	Name

The following is a sample PublicModeStatusl:

PUBLICMODELSTATUS

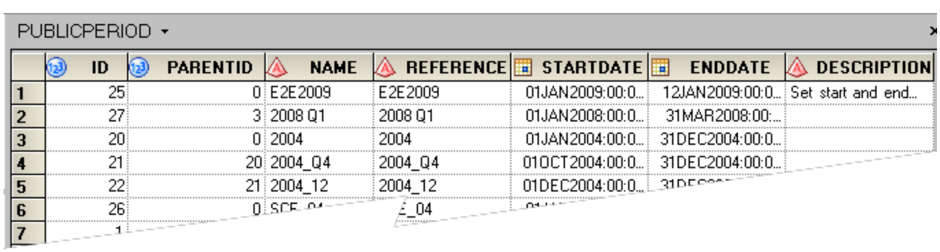
	MODELID	PERIOD	SCENARIO	CALCULATIONSTATUS
1	1012	E2E2009	ETE2_Actual	CALCULATED
2	1019	E2E2009	ETE2_Actual	NOT CALCULATED
3	1020	2008 Q1	Actual	CALCULATED
4	1021	2008 Q1	Actual	CALCULATED
5	1022	2008 Q1	Actual	[illegible]
6	1023	2008 Q1	Actual	
7	1020	[illegible]	[illegible]	
8				

PublicPeriod

Table 21.3 *Period View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Description	Text			Description
EndDate	Number			EndDate
ID	Number	PeriodID		ID values not viewable in the user interface
Name	Text		Name	Name
ParentID	Number	PeriodID		ID values not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			StartDate

The following is a sample PublicPeriod:


PUBLICPERIOD

	ID	PARENTID	NAME	REFERENCE	STARTDATE	ENDDATE	DESCRIPTION
1	25	0	E2E2009	E2E2009	01JAN2009:00:0...	12JAN2009:00:0...	Set start and end...
2	27	3	2008 Q1	2008 Q1	01JAN2008:00:0...	31MAR2008:00:...	
3	20	0	2004	2004	01JAN2004:00:0...	31DEC2004:00:0...	
4	21	20	2004_Q4	2004_Q4	01OCT2004:00:0...	31DEC2004:00:0...	
5	22	21	2004_12	2004_12	01DEC2004:00:0...	[illegible]	
6	26	0	[illegible]	[illegible]_04	[illegible]		
7	1						

PublicScenario

Table 21.4 *Scenario View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Description	Text			Description
ID	Number	ScenarioID		ID values not viewable in the user interface
Name	Text		Name	Name

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ParentID	Number	ScenarioID		ID values not viewable in the user interface
Reference	Text		Reference	Reference

The following is a sample PublicScenario:

PUBLICSCENARIO

	ID	PARENTID	NAME	REFERENCE	DESCRIPTION
1	23	0	ETE2_Actual	ETE2_Actual	Add ETE2_Actual...
2	25	0	Plan	Plan	
3	20	0	Planning	Planning	Tested by Song fr...
4	21	20	What_if1	What_if1	
5	22	20	What_If2	What_If2	
6	24	0	SCE_04	SCE_04	
7	1	0	Actual	Actual	Actual
8	2	0	Budget	Budget	

<*modelRef*>_PV_Account

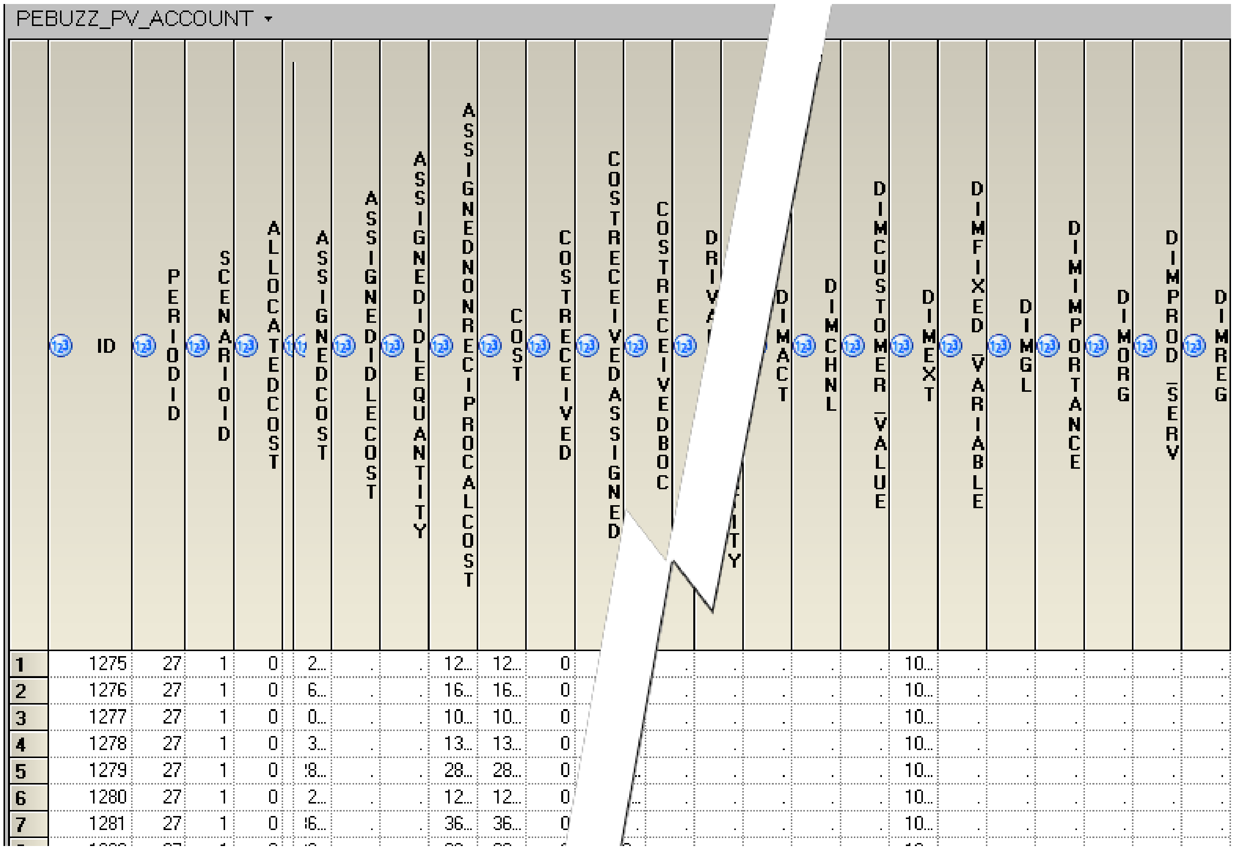
Table 21.5 *PublicAccount View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedCost	Number		Assigned Cost	Assigned Cost
AssignedIdleCost	Number		Assigned Idle Cost	Assigned Idle Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
AssignedNonReciprocalCost	Number		Assigned Non-Reciprocal Cost	Assigned Non-Reciprocal Cost
Cost	Number		Cost	Cost
CostReceived	Number		Received Cost	Received Cost
CostReceivedAssigned	Number		Received Assignment Cost	Received Assignment Cost
CostReceivedBOC	Number		Received BOC Cost	Received BOC Cost
Dim<*dimension_short_ref*>	Number			Specific dimensions in the model - one row per dimension in the model
DrivableCost	Number		Drivable Cost	Drivable Cost

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverID	Number	Driver.ID		ID values not viewable in the user interface
DriverRate	Number		Driver Rate	Driver Rate
HasAssignments	Number		Has Assignments	Has Assignments
HasBOC	Number		Has BOC	Has BOC
HasEnteredCost	Number		Has Entered Cost	Has Entered Cost
HasIdleCost	Number		Has Idle Cost	Has Idle Cost
HasNotes	Number		Has Notes	Has Notes
HasUsedCost	Number		Has Used Costs	Has Used Costs
IdleCost	Number		Idle Cost	Idle Cost
IdlePercentage	Number		Idle Percentage	Idle Percentage
IdleQuantity	Number		Idle Quantity	Idle Quantity
Measure	Text		Unit of Measure	Unit of Measure
ModuleID	Number			ID values not viewable in the user interface
ModuleType	Text		Module Type	Module Type
Name	Text		Name	Name
Note	Text		Periodic Note	Periodic Note
OutputQuantity	Number		Output Quantity	Output Quantity
PeriodID	Number	Period.ID		ID values not viewable in the user interface
Profit	Number		Profit	Profit
PublishName	Text			ID values not viewable in the user interface
Reference	Text		Reference	Reference
Revenue	Number		Revenue	Revenue
ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
SoldQuantity	Number		Sold Quantity	Sold Quantity

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
TotalDriverQuantity	Number		Total Driver Quantity (TDQ)	Total Driver Quantity (TDQ)
TotalDriverQuantityBasic	Number		Total Driver Quantity Basic (TDQBasic)	Total Driver Quantity Basic (TDQBasic)
TotalDriverQuantityCalculated	Number		Total Driver Quantity Calculated (TDQCalc)	Total Driver Quantity Calculated (TDQCalc)
Type	Text		Type	Type
UnassignedCost	Number		Unassigned Cost	Unassigned Cost
UnassignedQuantity	Number		Unassigned Quantity	Unassigned Quantity
UnitCost	Number		Unit Cost	Unit Cost
UnitCostEntered	Number		Unit Cost	Unit Cost - external units only
UnitProfit	Number		Unit Profit	Unit Profit
UnitRevenue	Number		Unit Revenue	Unit Revenue
UsedCost	Number		Used Cost	Used Cost
UsedQuantity	Number		Used Quantity	Used Quantity
UserOutputQuantity	Number		Output Quantity UE	Output Quantity UE
UserTotalDriverQuantity	Number		Total Driver Quantity UE (TDQUE)	Total Driver Quantity UE (TDQUE)

The following is a sample PV_Account:



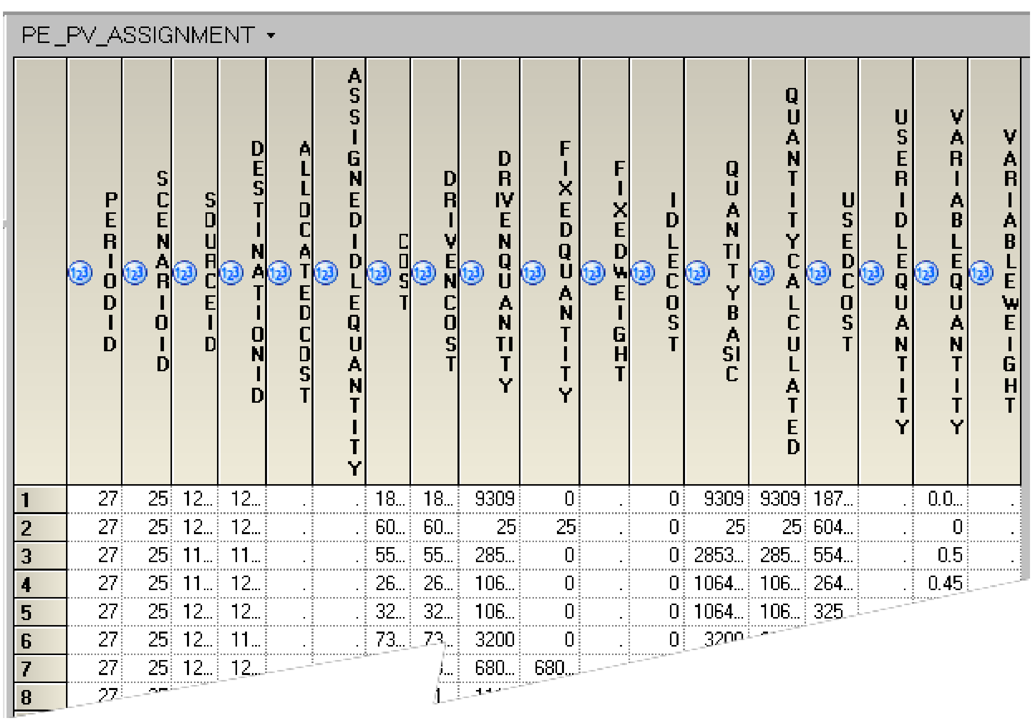
*<modelRef>*_PV_Assignment

Table 21.6 *PublicAssignment View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationID	Number	Account.ID		ID values not viewable in the user interface
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
PeriodID	Number	Period.ID		ID values not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic
QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
SourceID	Number	Account.ID		ID values not viewable in the user interface
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable

The following is a sample PV_Assignment:


PE_PV_ASSIGNMENT

	PERIODID	SCENARIOID	SOURCEID	DESTINATIONID	ALLOCATEDCOST	ASSIGNEDIDLEQUANTITY	COST	DRIVENCOST	DRIVENQUANTITY	FIXEDQUANTITY	FIXEDWEIGHT	IDLECOST	QUANTITYBASIC	QUANTITYCALCULATED	USEDCOST	USERIDLEQUANTITY	VARIABLEQUANTITY	VARIABLEWEIGHT
1	27	25	12...	12...	.	.	18...	18...	9309	0	.	0	9309	9309	187...	.	0.0...	.
2	27	25	12...	12...	.	.	60...	60...	25	25	.	0	25	25	604...	.	0	.
3	27	25	11...	11...	.	.	55...	55...	285...	0	.	0	2853...	285...	554...	.	0.5	.
4	27	25	11...	12...	.	.	26...	26...	106...	0	.	0	1064...	106...	264...	.	0.45	
5	27	25	12...	12...	.	.	32...	32...	106...	0	.	0	1064...	106...	325			
6	27	25	12...	11...	.	.	73...	73...	3200	0	.	0	3200					
7	27	25	12...	12...					680...	680...								
8	27																	

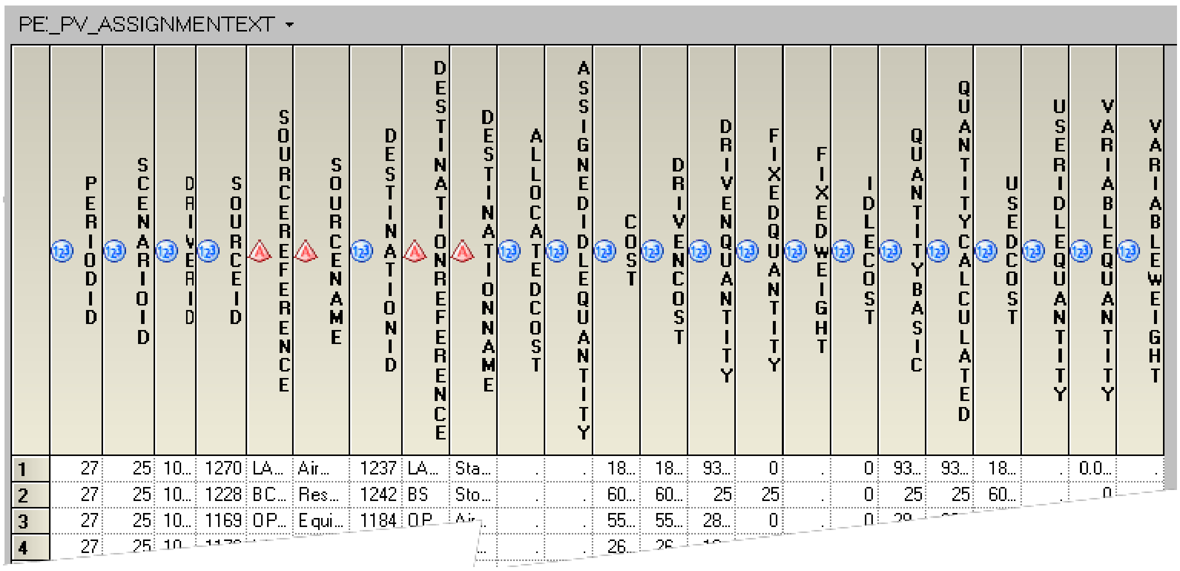
<*modelRef*>_PV_AssignmentExt

Table 21.7 *PublicAssignmentExt View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationID	Number	Account.ID		ID values not viewable in the user interface
DestinationName	Text		Name	Name
DestinationReference	Text		Reference	Reference
DrivenCost	Number		Driven Cost	Driven Cost
DriverID	Number			ID values not viewable in the user interface
DriverQuantity	Number			Driver Quantity
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost
PeriodID	Number	Period.ID		ID values not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic
QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
SourceID	Number	Account.ID		ID values not viewable in the user interface
SourceName	Text		Name	Name
SourceReference	Text		Reference	Reference
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable

The following is a sample PV_AssignmentExt:



PE_PV_ASSIGNMENTEXT

	PERIODID	SCENARIOID	DRIVERID	SOURCEID	SOURCEREFERENCE	SOURCENAME	DESTINATIONID	DESTINATIONREFERENCE	DESTINATIONNAME	ALLOCATEDCOST	ASSIGNEDIDLEQUANTITY	COST	DRIVENCOST	DRIVENQUANTITY	FIXEDQUANTITY	FIXEDWEIGHT	IDLECOST	QUANTITYBASIC	QUANTITYCALCULATED	USEDCOST	USERIDLEQUANTITY	VARIABLEQUANTITY	VARIABLEWEIGHT
1	27	25	10...	1270	LA...	Air...	1237	LA...	Sta...	.	.	18...	18...	93...	0	.	0	93...	93...	18...	.	0.0...	.
2	27	25	10...	1228	BC...	Res...	1242	BS	Sto...	.	.	60...	60...	25	25	.	0	25	25	60...	.	0	
3	27	25	10...	1169	OP...	Equi...	1184	OP...		.	.	55...	55...	28...	0	.	0						
4	27									.	.	26...											

<*modelRef*>_PV_Attribute

This view is new for attributes on a dimension member.

Table 21.8 *AttributeValue View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
Name	Text		Name	Name
Reference	Text		Reference	Reference
ParentID	Number			ID values not viewable in the user interface
Type	Text			Type
DimensionID	Number			ID values not viewable in the user interface

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ReferenceID	Number			ID values not viewable in the user interface
Measure	Text		Unit of Measure	UoM

The following is a sample PV_Attribute:

PE_PV_ATTRIBUTE

	ID	NAME	REFERENCE	PARENTID	TYPE	DIMENSIONID	REFERENCEID	MEASURE
1	1069	Compled Expedit...	Compled Expedit...	0	32	.	.	Completed Requ...
2	1070	Average Time To...	Average Time To...	0	32	.	.	Hours
3	1071	Number of Inspe...	Number of Inspe...	0	32	.	.	Inspections
4	1072	Inspections Pass...		0	32			
5								

<*modelRef*>_PV_AttributeValue

Table 21.9 *AttributeValue View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Attribute	Text		Name	Name
ItemID	Number	Account.ID		ID values not viewable in the user interface
NumericValue	Number			Specific Attribute Value
PeriodID	Number	Period.ID		ID values not viewable in the user interface
ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
StringValue	Text			Specific Attribute Value

The following is a sample PV_AttributeValue:

PE_PV_ATTRIBUTEVALUE

	PERIODID	SCENARIOID	ITEMID	ATTIBUTE	NUMERICVALUE	STRINGVALUE
1	27	1	1091	Compled Expedit...	2000	
2	27	1	1091	Average Time To...	0.5	
3	27	25	1165	Capacity Varianc...	-0.1568	
4	27	25	1165	Units Available	2	
5	27	25	1167	Capacity Available	3500	
6	27	25	1167	Capacity Provided		
7	27	25	1167			
8	27					
9						

<*modelRef*>_PV_Dimension

Table 21.10 *Dimension View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
Name	Text		Dimension Name	Dimension Name

The following is a sample PV_Dimension:

PE_PV_DIMENSION

	ID	NAME
1	1001	Activities
2	1002	Channel
3	1003	General Ledger
4	1004	External Units
5	1005	Organization
6	1006	Products and Se...
7	1007	Region
8	1008	Stages
9	1009	Fixed_Variable
10	1010	Customer Value
11	1011	Importance

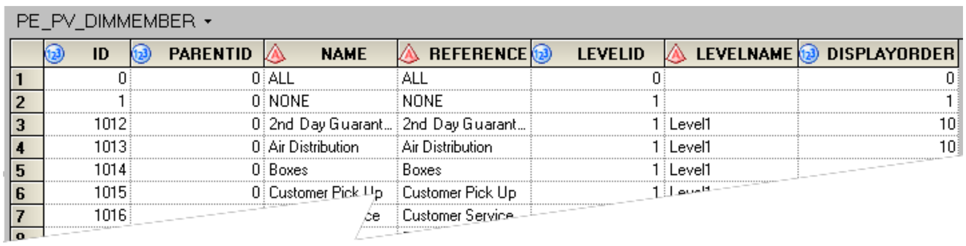
<*modelRef*>_PV_DimMember

Table 21.11 *DimMember View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
LevelID	Number		Dimension Level Number	Dimension Level Number
LevelName	Text		Dimension Level Name	Dimension Level Name
Name	Text		Dimension Member Name	Dimension Member Name
ParentID	Number			ID values not viewable in the user interface
Reference	Text		Dimension Member Reference	Dimension Member Reference

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
DisplayOrder	float			

The following is a sample PV_DimMember:


PE_PV_DIMMEMBER

	ID	PARENTID	NAME	REFERENCE	LEVELID	LEVELNAME	DISPLAYORDER
1	0	0	ALL	ALL	0		0
2	1	0	NONE	NONE	1		1
3	1012	0	2nd Day Guarant...	2nd Day Guarant...	1	Level1	10
4	1013	0	Air Distribution	Air Distribution	1	Level1	10
5	1014	0	Boxes	Boxes	1	Level1	
6	1015	0	Customer Pick Up	Customer Pick Up	1		
7	1016		ce	Customer Service			

<*modelRef*>_PV_Driver

Table 21.12 *Driver View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Formula	Text		Driver Formula	Formula
ID	Number	Account.DriverID and Assignment.DriverI D		ID values not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver Checkbox: Shared or Unique
RuleFormula	Text		Rule Formula	Rule Formula
UseRuleFormula	Number		Use Rule Formula	Driver Checkbox: Use Rule Formula

The following is a sample PV_Driver:

PE_PV_DRIVER

	ID	NAME	IMPLEMENTATIONTYPE	QUANTITYTYPE	FORMULA	USEFIXEDQUANTITIES	USEVARIABLEQUANTITIES	USEWEIGHTEDQUANTITIES	FIXEDDQOVERRIDE	VARIABLEDQOVERRIDE	IDLEFLOWMETHOD	SEQUENCENO	USERENTEREDCOSTALLOCATION	RULEFORMULA	USERULEFORMULA
1	1	Evenly Assigned	Ev...	UN...		1	0	0	0	0	Do...	1	0		0
2	2	Percentage	Per...	UN...		1	0	0	0	0	Do...	1	0		0
3	1085	# of Complaints	Ba...	UN...		1	0	0	0	0	Do...	1	0		0
4	1086	FTE	Ba...	UN...		1	0	0	0	0	Do...	1	0		0
5	1087	Allocated Cost	Ba...	SH...		0	0	0	0	0	Us...	1	1		0
6	1088	Consumption D...	Ba...	UN...		1	1	0	0	0	Do...	1	0		0
7	1089	# of Packages	Ba...	UN...		1	1	0	0	0	Do...	1	0		0
8	3	Bill of Cost	Bill...	UN...		1	1	0	0	0	Do...	1	0		0
9	1090	# of Expedite R...	Cal...	UN...	("A...	1	0	0	0	0	Do...	1	0		0
10	4	Sales volume	Sal...	SH...		0	1	0	0	0	Do...	1	0		0

<*modelRef*>_PV_EnteredCE

Table 21.13 *EnteredCE View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Cost	Number		Cost	Cost
Name	Text		Name	Name
ParentID	Number	Account.ID		ID values not viewable in the user interface
PeriodID	Number	Period.ID		ID values not viewable in the user interface
Reference	Text		Reference	Reference
ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface

The following is a sample PV_EnteredCE:

PE_PV_ENTEREDCE

	PERIODID	SCENARIOID	PARENTID	NAME	REFERENCE	COST
1	27	1	1151	Equipment Expe...	BPDEE	52300
2	27	1	1156	Equipment Expe...	OPHEE	33900
3	27	1	1160	Equipment Expe...	EPDEE	21500
4	27	1	1163	Equipment Expe...	EPHEE	
5	27	1	1165	Equipment Expe...	ECSEE	
			1167	Equipment E		

<*modelRef*>_PD_<DimRef>

Table 21.14 *<modelRef>_PD_<DimRef> View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
All_*nnnnnn*	Text			Specific model dimension (where *nnnnnn* is the dimension name)
ID	Number			ID values not viewable in the user interface
Level1	Text		Dimension Level Name	Dimension Level Name
DisplayOrder	float			

The following is a sample <*modRef*>_PD_<*dimRef*>:

PE_PD_CHNL

	ID	DISPLAYORDER	ALL_CHANNEL	ID_0	DISPLAYORDER_0	LEVEL1	ID_1	DISPLAYORDER_1
1	0	0	ALL	0	0	ALL	0	0
2	1	2	ALL	0	0	NONE	1	1
3	1015	11	ALL	0	0	Customer Pick Up	1015	10
4	1017	21	ALL	0	0	Drop Box	1017	20
5	1037	31	ALL	0	0	Storefront	1037	30

<*modelRef*>_PD_CostElement

Table 21.15 *DimCostElement View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
CostElementName	Number		Name	Name

Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
ID	Number			ID values not viewable in the user interface

The following is a sample <*modRef*>_PD_CostElement:

PE_PD_COSTELEMENT

	ID	COSTELEMENTNAME
1	0	All
2	4	EnteredCostElement
3	5	Assignment
4	11	InternalCostElement
5	12	ExternalCostElement

<*modelRef*>_PD_Driver

Table 21.16 *DimDriver View*

Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
ID	Number			ID values not viewable in the user interface
Name	Text		Driver Name	Driver Name

The following is a sample <*modRef*>_PD_Driver:

PE_PD_DRIVER

	ID	DRIVERNAME
1	-2	None
2	0	All
3	1	Evenly Assigned
4	2	Percentage
5	3	Bill of Cost
6	4	Sales volume
7	1085	# of Complaints
8	1086	FTE
9	1087	Allocated Cost
10	1088	Consumption Driv...
11	1089	# of Packages
12	1090	# of Expedite Req...

<*modelRef*>_PD_Module

Table 21.17 *DimModule View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
ModuleName	Text		Module Type	Module

The following is a sample <*modRef*>_PD_Module:

PE_PD_MODULE

	ID	MODULENAME
1	-2	None
2	0	External Unit
3	1	Resource
4	2	Activity
5	3	Cost Object
6	4	Profitability

<*modelRef*>_PD_Period

Table 21.18 *DimPeriod View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
Period_L1 Name	Text			
......				
Period_L*n* Name	Text			

The following is a sample <*modRef*>_PD_Period:

PE_PD_PERIOD

	ID	STARTDATE	YEAR	ID_1	STARTDATE_1	QUARTER	ID_2	STARTDATE_2	MONTH	ID_3	STARTDATE_3	PERIOD_L4	ID_4	STARTDATE_4	PERIOD_L5	ID_5	STARTDATE_5	PERIOD_L6	ID_6	STARTDATE_6	PERIOD_L7	ID_7	STARTDATE_7
1	25	01JAN2009:00:0...	E2...	25	01...	E2...	25	01...	E2...	25	01...	E2...	25	01...	E2...	25	01...	E2...	25	01...	E2...	25	01...
2	20	01JAN2004:00:0...	20...	20	01...	20...	20	01...	20...	20	01...	20...	20	01...	20...	20	01...	20...	20	01...	20...	20	01...
3	26	01JAN2004:00:0...	SC...	26	01...	SC...	26	01...	SC...	26	01...	SC...	26	01...	SC...	26	01...	SC...	26	01...	SC...	26	01...
4	1	01JAN2006:00:0...	20...	1	01...	20...	1	01...	20...	1	01...	20...	1	01...	20...	1	01...	20...	1	01...	20...	1	01...
5	2	01JAN2007:00:0...	20...	2	01...	20...	2	01...	20...	2	01...	20...	2	01...	20...	2	01...	20...	2	01...	20...	2	01...
6	3	01JAN2008:00:0	20	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	01...	20...	3	01...	20...	3	01...	20...	3	01...	20...	3	01...

<*modelRef*>_PD_Scenario

Table 21.19 *DimScenario View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values not viewable in the user interface
Scenario_L1 Name	Text			
......				
Scenario_L*n* Name	Text			

The following is a sample <*modRef*>_PD_Scenario:

PE_PD_SCENARIO

	ID	TOP_SCENARIO	ID_1	SCENARIO_L2	ID_2	PERSONAL_SCENARIO	ID_3	SCENARIO_L4	ID_4	SCENARIO_L5	ID_5	SCENARIO_L6	ID_6	SCENARIO_L7	ID_7	SCENARIO_L8	ID_8	SCENARIO_L9	ID_9	SCENARIO_L10	ID_10
1	23	ETE2_Actual	23	ET...	23	ET...	23	ET...	23	ET...	23	ET...	23	ET...	23	ET...	23	ET...	23	ET...	23
2	25	Plan	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25	Pla...	25
3	20	Planning	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20	Pla...	20
4	24	SCE_04	24	SC...	24	SC...	24	SC...	24	SC...	24	SC...	24	SC...	24	SC...	24	SC...	24	SC...	24
5	1	Actual	1	Act...	1	Act...	1	Act...	1	Act...	1	Act...	1	Act...	1	Act...	1	Act...	1	Act...	1
6	2	Budget	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2	Bu...	2
7	21	Planning	20	Wh...	21	Wh...	21	Wh...	21	Wh...	21	Wh...	21	Wh...	21	Wh...	21	Wh...	21	Wh...	21
8	22	Planning	20	Wh...	22	Wh...	22	Wh...	22	Wh...	22	Wh...	22	Wh...	22	Wh...	22	Wh...	22	Wh...	22

<*modelRef*>_PD_YesNo

Table 21.20 *DimYesNo View*

Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
ID	Number			ID values not viewable in the user interface
YesNoName	Text			Yes or No Value

The following is a sample <*modRef*>_PD_YesNo:

PE_PD_YESNO

	ID	YESNONAME
1	0	No
2	1	Yes

<*modelRef*>_PF_MSC

This public view is created when a fact table is created for a 6.3–compatible multi-stage contribution cube.

Table 21.21 *FactMSC View*

Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
All_PeriodID	Number	Period.ID		ID values not viewable in the user interface
All_ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
Cost	Number		Cost	Cost
OutputQuantity	Number		Output Quantity	Output Quantity
Revenue	Number		Revenue	Revenue
Stage1Dim<*dimRef*>	Text	Join Dim Table		
Stage2Dim<*dimRef*>	Text	Join Dim Table		
Stage3Dim<*dimRef*>	Text	Join Dim Table		
Stage4Dim<*dimRef*>	Text	Join Dim Table		

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
SoldQuantity	Number		Sold Quantity	Sold Quantity

*<modelRef>*_PF_RC

This public view is created when a fact table is created for a resource contribution cube.

Table 21.22 *FactRC View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
All_PeriodID	Number	Period.ID		ID values not viewable in the user interface
All_ReciprocalID	Number			ID values not viewable in the user interface
All_ScenarioID	Number	Scenario.ID		ID values not viewable in the user interface
All_Type	Number			ID values not viewable in the user interface
Cost	Number		Cost	Cost
DriverQuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
Dst_Dim*<dimRef>*	Text	Join Dim Table		
Dst_Dim*<dimRef>*	Text	Join Dim Table		
Dst_DriverID	Number	Driver.ID		ID values not viewable in the user interface
Dst_ModuleID	Number	Module.ID		ID values not viewable in the user interface
OutputQuantity_I	Number		Output Quantity	Output Quantity
Revenue_I	Number		Revenue	Revenue
SoldQuantity_I	Number		Sold Quantity	Sold Quantity
Src_Dim1001	Number	Join Dim Table		ID values not viewable in the user interface
Src_Dim1002	Number	Join Dim Table		ID values not viewable in the user interface

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Src_Dim1004	Number	Join Dim Table		ID values not viewable in the user interface
Src_DriverID	Number	Driver.ID		ID values not viewable in the user interface
Src_ModuleID	Number	Module.ID		ID values not viewable in the user interface

<*modelRef*>_PF_SSC

This public view is created when a fact table is created for a single–stage contribution cube.

Table 21.23 *FactSSC View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Cost	float			
DriverQuantityCalculated	float			
Revenue_I	float			
OutputQuantity_I	float			
SoldQuantity_I	float			
Attribute*xxx*	float			One for each attribute selected
AllPeriodID	Number			
AllScenarioID	Number			
All_Type	Number			
All_ReciprocalID	Number			
Src_ModuleID	Number			
Dst_ModuleId	Number			
Src_Dim<*dimRef*>	Number			One for each dimension
Dst_Dim1<*dimRef*>	Number			One for each dimension

*<modelRef>*_PF_*<cubeConfigRef>*

This is a public view of the corresponding table named <*modelID*>_C<*cubeConfigID*>. This view is created when a fact table is created for a custom multi-stage contribution cube.

Table 21.24 *Custom MSC Cube*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
PeriodID	Number			
ScenarioID	Number			
S1Dim<*dimRef*>	Text			One row for each dimension in the cube
Cost	Number		Cost	
Output	Number			
Revenue	Number			
SoldQuantity	Number			
UsedTotalDriverQuantity	Number			
UserOutputQuantity	Number			
Attribute<*dimID*>	Number			One row for each dimensional attribute in the cube
TotalDriverQuantity	Number			

Chapter 22

Legacy Public Views

Overview

SAS Activity-Based Management 7.2 provides two sets of public views:

- Public views that use model References and dimension References, which don’t change when a model is exported and subsequently re-imported, rather than model IDs and dimension IDs, which can change when a model is exported and subsequently re-imported. For information on these public views, see Chapter 21, “Public Views,” on page 329.
- Legacy public views, which use model IDs and dimension IDs as documented in this chapter, are generated so as to support users who have implemented processes that assume this format. Legacy public views will no longer be generated after SAS Activity-Based Management 7.2.

The following table shows the name of legacy public views and the name of the corresponding public view that uses short references rather than IDs. The legacy public views are supported for SAS Activity-Based Management 7.2, but they will not be supported in subsequent releases.

Legacy Public View	New Public View
M<*modelID*>_PublicAccount	<*modelRef*>_PV_Account
M<*modelID*>_PublicAssignment	<*modelRef*>_PV_Assignment
M<*modelID*>_PublicAssignmentExt	<*modelRef*>_PV_AssignmentExt
did not exist before	<*modelRef*>_PV_Attributes
M<*modelID*>_PublicAttributeValue	<*modelRef*>_PV_AttributeValue
M<*modelID*>_PublicDimension	<*modelRef*>_PV_Dimension
M<*modelID*>_PublicDimMember	<*modelRef*>_PV_DimMember
M<*modelID*>_PublicDriver	<*modelRef*>_PV_Driver
M<*modelID*>_PublicEnteredCE	<*modelRef*>_PV_EnteredCE
M<*modelID*>_PublicDim<*dimID*>	<*modelRef*>_PD_<*dimRef*>
M<*modelID*>_PublicDimCostElement	<*modelRef*>_PD_CostElement
M<*modelID*>_PublicDimDriver	<*modelRef*>_PD_Driver
M<*modelID*>_PublicDimModule	<*modelRef*>_PD_Module
M<*modelID*>_PublicDimPeriod	<*modelRef*>_PD_Period
M<*modelID*>_PublicDimScenario	<*modelRef*>_PD_Scenario
M<*modelID*>_PublicDimYesNo	<*modelRef*>_PD_YesNo
M<*modelID*>_PublicFactMSC	<*modelRef*>_PF_MSC
M<*modelID*>_PublicFactRC	<*modelRef*>_PF_RC
M<*modelID*>_PublicFactSSC	<*modelRef*>_PF_SSC
did not exist before	<*modelRef*>_PF_<*cubeConfigRef*>

Table Column Descriptions

Field Name
The field name that is provided in the public view.

Field Type
The definition of each field type (numeric or text field).

Join Field
The specific ID fields to join between tables. These fields define keys that can be used to link multiple tables for reporting. This column identifies the corresponding join field.

Properties Reference
The specific corresponding property. See "Properties List Alphabetically" in the *SAS Activity Based Management: User's Guide*.

Model View Columns
The specific property that is displayed in the model column view.

To open public views using Microsoft SQL Server Enterprise Manager, open the **Databases** folder and expand the ABM Models database. Under ABM Models, double-click the **Views** icon to display the database views for the various models in the right pane. Right-click the PublicModel view and select **Open View** ⇨ **Return all rows** to display the table view that maps a model ID to a model name. In this example, the Inflight model maps to model ID 1195. The group of views that begin with M1195_Public... (where 1195 is the internal database model ID that is found in the PublicModel view) specifically pertain to the Inflight model.

M<*modelID*>_PublicAccount

Table 22.1 *PublicAccount View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedCost	Number		Assigned Cost	Assigned Cost
AssignedIdleCost	Number		Assigned Idle Cost	Assigned Idle Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
AssignedNonReciprocalCost	Number		Assigned Non-Reciprocal Cost	Assigned Non-Reciprocal Cost
Cost	Number		Cost	Cost
CostReceived	Number		Received Cost	Received Cost
CostReceivedAssigned	Number		Received Assignment Cost	Received Assignment Cost
CostReceivedBOC	Number		Received BOC Cost	Received BOC Cost
Dim10xx	Number			Specific dimensions in the model

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
DrivableCost	Number		Drivable Cost	Drivable Cost
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverId	Number	Driver.Id		ID values - not viewable in the user interface
DriverRate	Number		Driver Rate	Driver Rate
HasAssignments	Number		Has Assignments	Has Assignments
HasBOC	Number		Has BOC	Has BOC
HasEnteredCost	Number		Has Entered Cost	Has Entered Cost
HasIdleCost	Number		Has Idle Cost	Has Idle Cost
HasNotes	Number		Has Notes	Has Notes
HasUsedCost	Number		Has Used Costs	Has Used Costs
IdleCost	Number		Idle Cost	Idle Cost
IdlePercentage	Number		Idle Percentage	Idle Percentage
IdleQuantity	Number		Idle Quantity	Idle Quantity
Measure	Text		Unit of Measure	Unit of Measure
ModuleID	Number			ID values - not viewable in the user interface
ModuleType	Text		Module Type	Module Type
Name	Text		Name	Name
Note	Text		Periodic Note	Periodic Note
OutputQuantity	Number		Output Quantity	Output Quantity
PeriodId	Number	Period.Id		ID values - not viewable in the user interface
Profit	Number		Profit	Profit
PublishName	Text			ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Revenue	Number		Revenue	Revenue
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
SoldQuantity	Number		Sold Quantity	Sold Quantity
TotalDriverQuantity	Number		Total Driver Quantity (TDQ)	Total Driver Quantity (TDQ)
TotalDriverQuantityBasic	Number		Total Driver Quantity Basic (TDQBasic)	Total Driver Quantity Basic (TDQBasic)
TotalDriverQuantityCalculated	Number		Total Driver Quantity Calculated (TDQCalc)	Total Driver Quantity Calculated (TDQCalc)
Type	Text		Type	Type
UnassignedCost	Number		Unassigned Cost	Unassigned Cost
UnassignedQuantity	Number		Unassigned Quantity	Unassigned Quantity
UnitCost	Number		Unit Cost	Unit Cost
UnitCostEntered	Number		Unit Cost	Unit Cost - external units only
UnitProfit	Number		Unit Profit	Unit Profit
UnitRevenue	Number		Unit Revenue	Unit Revenue
UsedCost	Number		Used Cost	Used Cost
UsedQuantity	Number		Used Quantity	Used Quantity
UserOutputQuantity	Number		Output Quantity UE	Output Quantity UE
UserTotalDriverQuantity	Number		Total Driver Quantity UE (TDQUE)	Total Driver Quantity UE (TDQUE)

M<*modelID*>_PublicAssignment

Table 22.2 *PublicAssignment View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationId	Number	Account.ID		ID values - not viewable in the user interface

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
DestinationName	Text		Name	Name
DestinationReference	Text		Reference	Reference
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverId	Number	Driver.Id		ID values - not viewable in the user interface
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost
PeriodId	Number	Period.Id		ID values - not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic
QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioId	Number	Scenario.ID		ID values - not viewable in the user interface
SourceId	Number	Account.ID		ID values - not viewable in the user interface
SourceName	Text		Name	Name
SourceReference	Text		Reference	Reference
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable

M<*modelID*>_PublicAssignmentExt

Table 22.3 *PublicAssignmentExt View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationId	Number	Account.ID		ID values - not viewable in the user interface
DestinationName	Text		Name	Name
DestinationReference	Text		Reference	Reference
DrivenCost	Number		Driven Cost	Driven Cost
DriverId	Number			ID values - not viewable in the user interface
DriverQuantity	Number			Driver Quantity
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost
PeriodID	Number	Period.Id		ID values - not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic
QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
SourceID	Number	Account.ID		ID values - not viewable in the user interface
SourceName	Text		Name	Name
SourceReference	Text		Reference	Reference
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable

M<*modelID*>_PublicAttributeValue

Table 22.4 *AttributeValue View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Attribute	Text		Name	Name
ItemId	Number	Account.Id		ID values - not viewable in the user interface
NumericValue	Number			Specific Attribute Value
PeriodId	Number	Period.Id		ID values - not viewable in the user interface
ScenarioId	Number	Scenario.ID		ID values - not viewable in the user interface
StringValue	Text			Specific attribute value

M<*modelID*>_PublicDim<*dimID*>

Table 22.5 *Dimxxxx View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
All_*nnnnnn*	Text			Specific model dimension (Where *nnnnnn* is the dimension name)
Id	Number			ID values - not viewable in the user interface
Level1	Text		Dimension Level Name	Dimension Level Name
DisplayOrder	float			

M<*modelID*>_PublicDimCostElement

Table 22.6 *DimCostElement View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
CostElementName	Number		Name	Name
ID	Number			ID values - not viewable in the user interface

M<*modelID*>_PublicDimDriver

Table 22.7 *DimDriver View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values - not viewable in the user interface
Name	Text		Driver Name	Driver Name

M<*modelID*>_PublicDimension

Table 22.8 *Dimension View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Id	Number			ID values - not viewable in the user interface
Name	Text		Dimension Name	Dimension Name

M<*modelID*>_PublicDimMember

Table 22.9 *DimMember View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Id	Number			ID values - not viewable in the user interface
LevelId	Number		Dimension Level Number	Dimension Level Number
LevelName	Text		Dimension Level Name	Dimension Level Name
Name	Text		Dimension Member Name	Dimension Member Name
ParentId	Number			ID values - not viewable in the user interface
Reference	Text		Dimension Member Reference	Dimension Member Reference
DisplayOrder	float			

M<*modelID*>_PublicDimModule

Table 22.10 *DimModule View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values - not viewable in the user interface
ModuleName	Text		Module Type	Module

M<*modelID*>_PublicDimPeriod

Table 22.11 *DimPeriod View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values - not viewable in the user interface
Period_L1 Name	Text			
......				
Period_Ln Name	Text			

M<*modelID*>_PublicDimScenario

Table 22.12 *DimScenario View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values - not viewable in the user interface
Scenario_L1 Name	Text			
......				
Scenario_Ln Name	Text			

M<*modelID*>_PublicDimYesNo

Table 22.13 *DimYesNo View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
ID	Number			ID values - not viewable in the user interface
YesNoName	Text			Yes or No value

M<*modelID*>_PublicDriver

Table 22.14 *Driver View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Formula	Text		Driver Formula	Formula
Id	Number	Account.DriverId and Assignment.DriverId		ID values - not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver checkbox: Shared or Unique
RuleFormula	Text		Rule Formula	Rule Formula
UseRuleFormula	Number		Use Rule Formula	Driver checkbox: Use Rule Formula

M<*modelID*>_PublicEnteredCE

Table 22.15 *EnteredCE View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Cost	Number		Cost	Cost
Name	Text		Name	Name
ParentId	Number	Account.ID		ID values - not viewable in the user interface
PeriodID	Number	Period.Id		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
ScenarioId	Number	Scenario.ID		ID values - not viewable in the user interface

M<*modelID*>_PublicFactMSC

Table 22.16 *FactMSC View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
All_PeriodID	Number	Period.Id		ID values - not viewable in the user interface
All_ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
Cost	Number		Cost	Cost
OutputQuantity	Number		Output Quantity	Output Quantity
Revenue	Number		Revenue	Revenue
Stage1Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage2Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage2Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage3Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage3Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage4Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
Stage4Dim10xx	Number	Join Dim Table		ID values - not viewable in the user interface
SoldQuantity	Number		Sold Quantity	Sold Quantity

M<*modelID*>_PublicFactRC

Table 22.17 *FactRC View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
All_PeriodID	Number	Period.Id		ID values - not viewable in the user interface
All_ReciprocalID	Number			ID values - not viewable in the user interface
All_ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
All_Type	Number			ID values - not viewable in the user interface
Cost	Number		Cost	Cost
DriverQuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
Dst_Dim1005	Number	Join Dim Table		ID values - not viewable in the user interface
Dst_Dim1006	Number	Join Dim Table		ID values - not viewable in the user interface
Dst_DriverId	Number	Driver.Id		ID values - not viewable in the user interface
Dst_ModuleId	Number	Module.Id		ID values - not viewable in the user interface
OutputQuantity_I	Number		Output Quantity	Output Quantity
Revenue_I	Number		Revenue	Revenue
SoldQuantity_I	Number		Sold Quantity	Sold Quantity
Src_Dim1001	Number	Join Dim Table		ID values - not viewable in the user interface
Src_Dim1002	Number	Join Dim Table		ID values - not viewable in the user interface
Src_Dim1004	Number	Join Dim Table		ID values - not viewable in the user interface
Src_DriverID	Number	Driver.Id		ID values - not viewable in the user interface

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Src_ModuleId	Number	Module.Id		ID values - not viewable in the user interface

M<*modelID*>_PublicFactSSC

Table 22.18 *FactSSC View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Cost	float			
DriverQuantityCalculated	float			
Revenue_I	float			
OutputQuantity_I	float			
SoldQuantity_I	float			
Attributexxx	float			1 for each attribute selected
AllPeriodId	Number			
AllScenarioId	Number			
All_Type	Number			
All_ReciprocalId	Number			
Src_ModuleId	Number			
Dst_ModuleId	Number			
Src_Dim10xx	Number			1 for each dimension
Dst_Dim10xx	Number			1 for each dimension

PublicModel

Table 22.19 *Model View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Default*modelID*	Number			ID values - not viewable in the user interface
Name	Text			Model

PublicModelStatus

Table 22.20 *ModelStatus View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
CalculationStatus	Text			Calculation status - not viewable in the user interface
CubeStatus	Text			Cube status - not viewable in the user interface
ModelId	Number			ID values - not viewable in the user interface
Period	Text		Name	Name
Scenario	Text		Name	Name

PublicPeriod

Table 22.21 *Period View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Description	Text			Description
EndDate	Number			EndDate

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Id	Number	PeriodId		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentId	Number	PeriodId		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			StartDate

PublicScenario

Table 22.22 *Scenario View*

Field Name	Field Type	Join Field	Properties Reference	Model View Columns
Description	Text			Description
Id	Number	ScenarioID		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentId	Number	ScenarioID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference

Part 7

Information Maps

Chapter 23
Information Maps

Overview

SAS Activity-Based Management provides two types of information maps: Account information maps and Assignment information maps. The tables in this section describe the schema for the Account information maps and Assignment information maps.

The following table shows the name of legacy information maps and the name of the corresponding information map that uses short references rather than IDs. The legacy information maps are supported for SAS Activity-Based Management 7.2, but they will not be supported in subsequent releases.

Legacy Information Map	New Information Map
M<*modelID*>_<*modelName*>_AccountMap	<*modelRef*>_AccountMap
M<*modelID*>_<*modelName*>_AssignmentMap	<*modelRef*>_AssignmentMap

Information Map — Column Descriptions

Source Public View
The name of the view that is provided in the SAS Activity-Based Management model database and the source for the particular information map property. See Chapter 21, “Public Views,” on page 329.

Field Name
The field name that is provided in the information map.

Field Type
The definition of each field type (numeric or text field).

Join Field
The specific ID fields to join between tables. These fields define keys that can be used to link multiple tables for reporting. This column identifies the corresponding join field for this information map.

Properties Reference
The specific corresponding property. See "Properties List Alphabetically" in the *SAS Activity Based Management: User's Guide*.

Model View Columns
The specific property that is displayed in the model column view.

Account IMAP Schema

The following table contains the schema for the AccountIMAP information map M<*modelRef*>_AccountMap (where <*modelRef*> is the model Short Reference). This name is generated in the Register Metadata step for every model in SAS Activity-Based Management.

***Table 23.1** <modelRef>_AccountMap Schema*

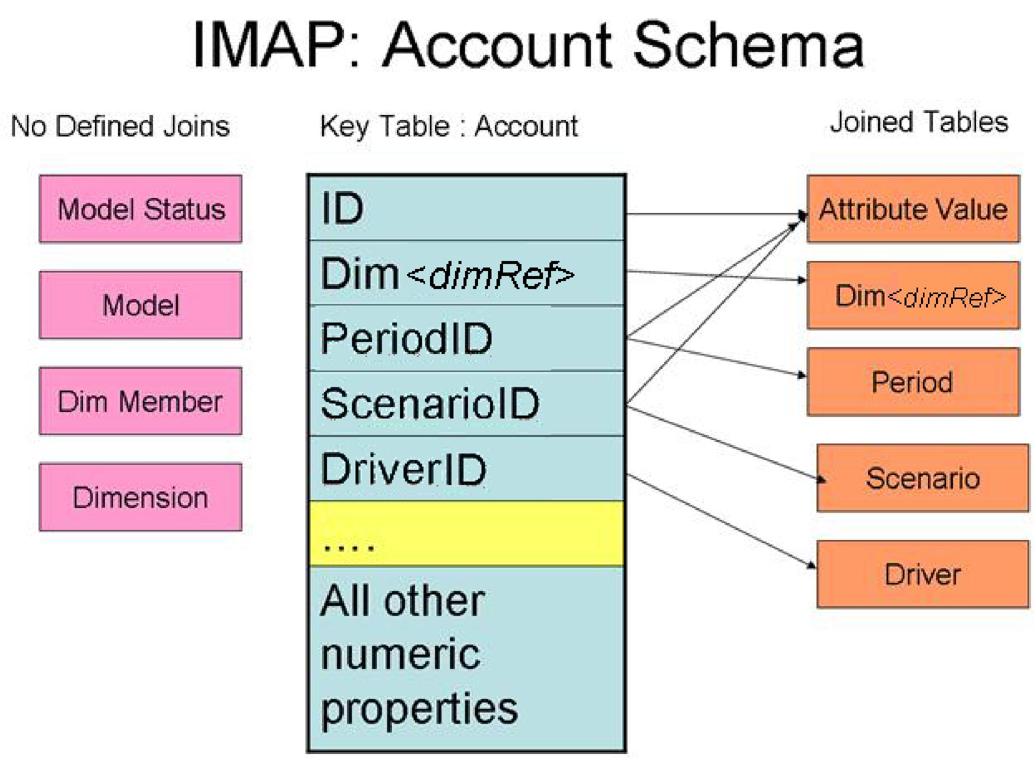
Source Public View: Period See "PublicPeriod " on page 334.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
ID	Number	Period.ID		ID values - not viewable in the user interface
Description	Text			Description
EndDate	Number			EndDate
Name	Text		Name	Name
ParentID	Number	Period.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			StartDate
Source Public View: Scenario See "PublicScenario " on page 334.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
ID	Number	Scenario.ID		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentID	Number	Scenario.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Source Public View: Account See "<modelRef>_PV_Account " on page 335.				

Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedCost	Number		Assigned Cost	Assigned Cost
AssignedIdleCost	Number		Assigned Idle Cost	Assigned Idle Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
AssignedNonReciprocalCost	Number		Assigned Non-Reciprocal Cost	Assigned Non-Reciprocal Cost
Cost	Number		Cost	Cost
CostReceived	Number		Received Cost	Received Cost
CostReceivedAssigned	Number		Received Assignment Cost	Received Assignment Cost
CostReceivedBOC	Number		Received BOC Cost	Received BOC Cost
Dim<*dimShortRef*>	Number			One column for each dimension in the model
DrivableCost	Number		Drivable Cost	Drivable Cost
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverID	Number	Driver.ID		ID values - not viewable in the user interface
DriverRate	Number		Driver Rate	Driver Rate
HasAssignments	Number		Has Assignments	Has Assignments
HasBOC	Number		Has BOC	Has BOC
HasEnteredCost	Number		Has Entered Cost	Has Entered Cost
HasIdleCost	Number		Has Idle Cost	Has Idle Cost
HasNotes	Number		Has Notes	Has Notes
HasUsedCost	Number		Has Used Costs	Has Used Costs
ID	Number			ID values - not viewable in the user interface
IdleCost	Number		Idle Cost	Idle Cost
IdlePercentage	Number		Idle Percentage	Idle Percentage
IdleQuantity	Number		Idle Quantity	Idle Quantity
Measure	Text		Unit of Measure	Unit of measure

ModuleID	Number			ID values - not viewable in the user interface
ModuleType	Text		Module Type	Module Type
Name	Text		Name	Name
Note	Text		Periodic Note	Periodic Note
OutputQuantity	Number		Output Quantity	Output Quantity
PeriodID	Number	Period.ID		ID values - not viewable in the user interface
Profit	Number		Profit	Profit
PublishName	Text			ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Revenue	Number		Revenue	Revenue
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
SoldQuantity	Number		Sold Quantity	Sold Quantity
TotalDriverQuantity	Number		Total Driver Quantity (TDQ)	Total Driver Quantity (TDQ)
TotalDriverQuantityBasic	Number		Total Driver Quantity Basic (TDQBasic)	Total Driver Quantity Basic (TDQBasic)
TotalDriverQuantityCalculated	Number		Total Driver Quantity Calculated (TDQCalc)	Total Driver Quantity Calculated (TDQCalc)
Type	Text		Type	Type
UnassignedCost	Number		Unassigned Cost	Unassigned Cost
UnassignedQuantity	Number		Unassigned Quantity	Unassigned Quantity
UnitCost	Number		Unit Cost	Unit Cost
UnitCostEntered	Number		Unit Cost	Unit Cost - external units only
UnitProfit	Number		Unit Profit	Unit Profit
UnitRevenue	Number		Unit Revenue	Unit Revenue
UsedCost	Number		Used Cost	Used Cost
UsedQuantity	Number		Used Quantity	Used Quantity
UserOutputQuantity	Number		Output Quantity UE	Output Quantity UE
UserTotalDriverQuantity	Number		Total Driver Quantity UE (TDQUE)	Total Driver Quantity UE (TDQUE)

Source Public View: Driver See “<modelRef>_PV_Driver ” on page 344.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Formula	Text		Driver Formula	Formula
ID	Number	Driver.ID		ID values - not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver Checkbox: Shared or Unique
Source Public View: AttributeValue See “<modelRef>_PV_AttributeValue ” on page 342.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Attribute	Text		Name	Name
ItemID	Number	Account.ID		ID values - not viewable in the user interface
NumericValue	Number			Specific Attribute Value
PeriodID	Number	Period.ID		ID values - not viewable in the user interface
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
StringValue	Text			Specific Attribute Value
Source Public View: *<modelRef>*_PD_*<DimRef>* See “<modelRef>_PD_<DimRef> ” on page 346.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Act_L1	Text			Specific Model Dimension (Example: Activity)
Act_L2	Text			Specific Model Dimension (Example: ProcessOrders)
All_*dim_name*	Text			Specific Model Dimension (Example - Activity)
Displayorder	Number			Implicit in the order of dimension member. It is not a specific column in the user interface..
ID	Number	Account.Dim1000		ID values - not viewable in the user interface

Level1	Text		Dimension Level Name	Dimension Level Name



Assignment IMAP Schema

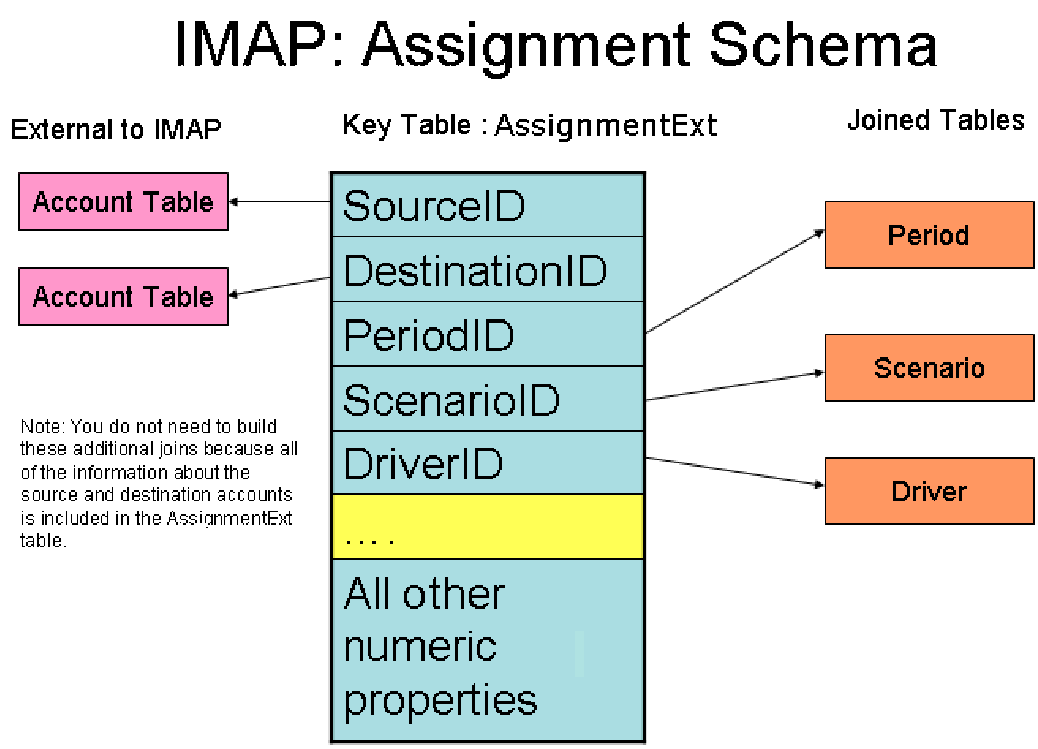
The following table contains the schema for the AssignmentIMAP information map M<*modelRef*>_AssignmentMap (where <*modelRef*> is the model Short Reference). This name is generated in the Register Metadata step for every model in SAS Activity-Based Management.

***Table 23.2** <modelRef>_AssignmentMap schema*

Source Public View: Period See “PublicPeriod ” on page 334.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
EndDate	Number			End Date
ID	Number	Period.ID		ID values - not viewable in the user interface
Name	Text		Name	Name

ParentID	Number	Period.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			Start Date
Source Public View: Scenario See “PublicScenario ” on page 334				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
ID	Number	Scenario.ID		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentID	Number	Scenario.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Source Public View: AssignmentExt See “<modelRef>_PV_AssignmentExt ” on page 340.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationID	Number	Account.ID		ID values - not viewable in the user interface
DestinationName	Text		Name	Name
DestinationReference	Text		Reference	Reference
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverID	Number	Driver.ID		ID values - not viewable in the user interface
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost
PeriodID	Number	Period.ID		ID values - not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic

QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
SourceID	Number	Account.ID		ID values - not viewable in the user interface
SourceName	Text		Name	Name
SourceReference	Text		Reference	Reference
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable
Source Public View: Driver See "<modelRef>_PV_Driver " on page 344.				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Formula	Text		Driver Formula	Formula
ID	Number	Driver.ID		ID values - not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver Checkbox: Shared or Unique


IMAP: Assignment Schema
External to IMAP
Key Table : AssignmentExt
Joined Tables
Account Table
Account Table
SourceID
DestinationID
PeriodID
ScenarioID
DriverID
....
All other numeric properties
Period
Scenario
Driver
Note: You do not need to build these additional joins because all of the information about the source and destination accounts is included in the AssignmentExt table.

Chapter 24

Legacy Information Maps

Overview

SAS Activity-Based Management 7.2 provides two sets of information maps:

- Information maps that use dimension References, which don't change when a model is exported and subsequently re-imported, rather than dimension IDs, which can change when a model is exported and subsequently re-imported. For information on these information maps, see Chapter 23, "Information Maps," on page 375.
- Legacy information maps, which use dimension IDs as documented in this chapter, are generated so as to support users who have implemented processes that assume this format. Legacy information maps will no longer be generated after SAS Activity-Based Management 7.2.

The following table shows the name of legacy information maps and the name of the corresponding information map that uses short references rather than IDs. The legacy information maps are supported for SAS Activity-Based Management 7.2, but they will not be supported in subsequent releases.

Legacy Information Map	New Information Map
M<*modelID*>_<*modelName*>_AccountMap	<*modelRef*>_AccountMap
M<*modelID*>_<*modelName*>_AssignmentMap	<*modelRef*>_AssignmentMap

Information Map — Column Descriptions

Source Public View
The name of the view that is provided in the SAS Activity-Based Management model database and the source for the particular information map property. See Chapter 21, "Public Views," on page 329.

Field Name
The field name that is provided in the information map.

Field Type
The definition of each field type (numeric or text field).

Join Field
The specific ID fields to join between tables. These fields define keys that can be used to link multiple tables for reporting. This column identifies the corresponding join field for this information map.

Properties Reference
The specific corresponding property. See "Properties List Alphabetically" in the *SAS Activity Based Management: User's Guide*.

Model View Columns
The specific property that is displayed in the model column view.

Account IMAP Schema

The following table contains the schema for the AccountIMAP information map. M<ModelID>_<ModelName>_AccountMap (where <ModelID> is the internal model database ID and <ModelName> is the name of the model) is the name of the information map that is provided in SAS Management Console -- for example, M1195_Inflight_AccountMap. This name is generated in the Register Metadata step for every model in SAS Activity-Based Management.

Table 24.1 *Account IMAP schema*

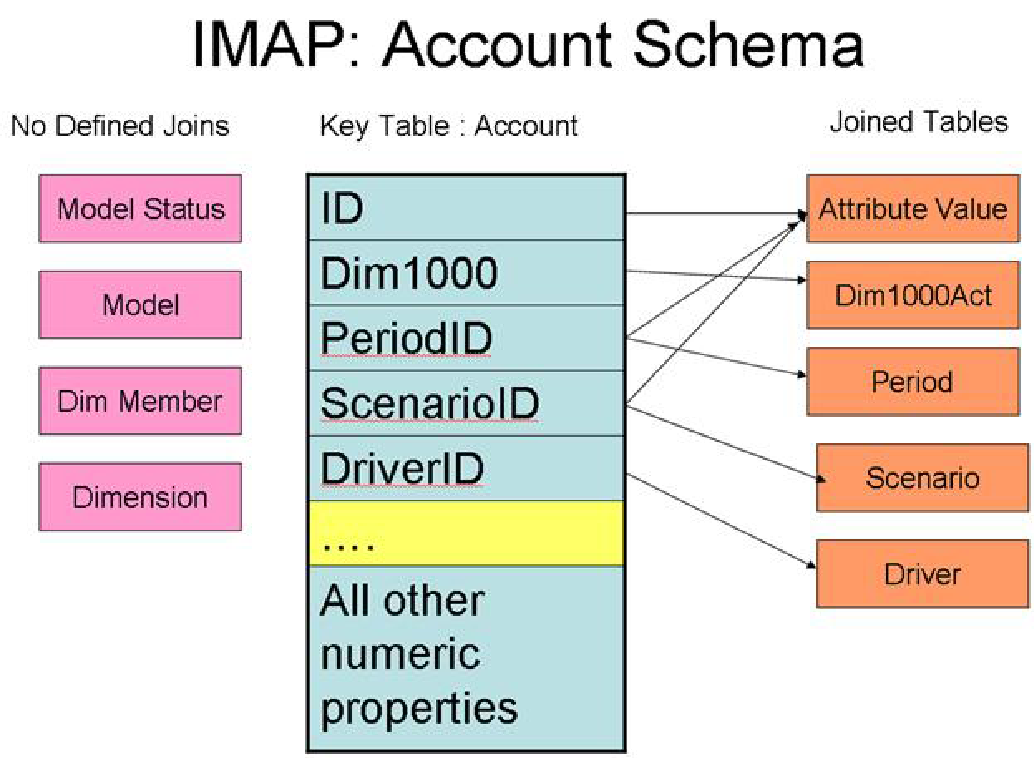
Source Public View: Period				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Id	Number	Period.Id		ID values - not viewable in the user interface
Description	Text			Description
EndDate	Number			EndDate
Name	Text		Name	Name
ParentId	Number	Period.Id		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			StartDate
Source Public View: Scenario				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
Id	Number	Scenario.ID		ID values - not viewable in the user interface

Name	Text		Name	Name
ParentId	Number	Scenario.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Source Public View: Account				
`Field Name`	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedCost	Number		Assigned Cost	Assigned Cost
AssignedIdleCost	Number		Assigned Idle Cost	Assigned Idle Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
AssignedNonReciprocalCost	Number		Assigned Non-Reciprocal Cost	Assigned Non-Reciprocal Cost
Cost	Number		Cost	Cost
CostReceived	Number		Received Cost	Received Cost
CostReceivedAssigned	Number		Received Assignment Cost	Received Assignment Cost
CostReceivedBOC	Number		Received BOC Cost	Received BOC Cost
Dim1000	Number			Specific dimensions in the model
DrivableCost	Number		Drivable Cost	Drivable Cost
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverId	Number	Driver.Id		ID values - not viewable in the user interface
DriverRate	Number		Driver Rate	Driver Rate
HasAssignments	Number		Has Assignments	Has assignments
HasBOC	Number		Has BOC	Has BOC
HasEnteredCost	Number		Has Entered Cost	Has Entered Cost
HasIdleCost	Number		Has Idle Cost	Has Idle Cost
HasNotes	Number		Has Notes	Has notes
HasUsedCost	Number		Has Used Costs	Has Used Costs

Id	Number			ID values - not viewable in the user interface
IdleCost	Number		Idle Cost	Idle Cost
IdlePercentage	Number		Idle Percentage	Idle Percentage
IdleQuantity	Number		Idle Quantity	Idle Quantity
Measure	Text		Unit of Measure	Unit of measure
ModuleId	Number			ID values - not viewable in the user interface
ModuleType	Text		Module Type	Module Type
Name	Text		Name	Name
Note	Text		Periodic Note	Periodic Note
OutputQuantity	Number		Output Quantity	Output Quantity
PeriodId	Number	Period.Id		ID values - not viewable in the user interface
Profit	Number		Profit	Profit
PublishName	Text			ID values - not viewable in the user interface
Reference	Text		Reference	Reference
Revenue	Number		Revenue	Revenue
ScenarioID	Number	Scenario.ID		ID values - not viewable in the user interface
SoldQuantity	Number		Sold Quantity	Sold Quantity
TotalDriverQuantity	Number		Total Driver Quantity (TDQ)	Total Driver Quantity (TDQ)
TotalDriverQuantityBasic	Number		Total Driver Quantity Basic (TDQBasic)	Total Driver Quantity Basic (TDQBasic)
TotalDriverQuantityCalculated	Number		Total Driver Quantity Calculated (TDQCalc)	Total Driver Quantity Calculated (TDQCalc)
Type	Text		Type	Type
UnassignedCost	Number		Unassigned Cost	Unassigned Cost
UnassignedQuantity	Number		Unassigned Quantity	Unassigned Quantity
UnitCost	Number		Unit Cost	Unit Cost
UnitCostEntered	Number		Unit Cost	Unit Cost - external units only
UnitProfit	Number		Unit Profit	Unit Profit

UnitRevenue	Number		Unit Revenue	Unit Revenue
UsedCost	Number		Used Cost	Used Cost
UsedQuantity	Number		Used Quantity	Used Quantity
UserOutputQuantity	Number		Output Quantity UE	Output Quantity UE
UserTotalDriverQuantity	Number		Total Driver Quantity UE (TDQUE)	Total Driver Quantity UE (TDQUE)
Source Public View: Driver				
`Field Name`	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Formula	Text		Driver Formula	Formula
Id	Number	Driver.Id		ID values - not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver checkbox: Shared or Unique
Source Public View: AttributeValue				
`Field Name`	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Attribute	Text		Name	Name
ItemId	Number	Account.Id		ID values - not viewable in the user interface
NumericValue	Number			Specific Attribute Value
PeriodId	Number	Period.Id		ID values - not viewable in the user interface
ScenarioId	Number	Scenario.ID		ID values - not viewable in the user interface
StringValue	Text			Specific attribute value
Source Public View: Dimxxx				
`Field Name`	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Act_L2	Text			Specific model dimension (Example: Activity)
All_*dim_name*	Text			Specific model dimension (Example - Activity)

Id	Number	Account.Dim1000		ID values - not viewable in the user interface
Level1	Text		Dimension Level Name	Dimension Level Name



Assignment IMAP Schema

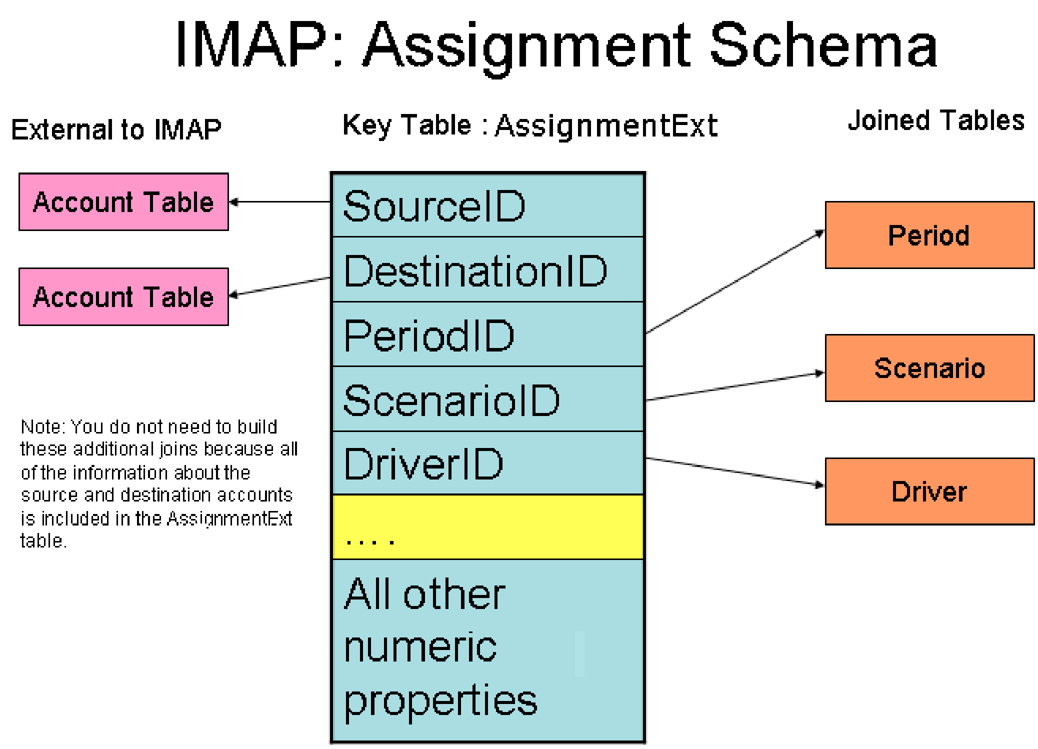
The following table contains the schema for the AssignmentIMAP information map. M<ModelID>_<ModelName>_AssignmentMap (where <ModelID> is the internal model database ID and <ModelName> is the name of the model) is the name of the information map that is provided in SAS Management Console -- for example, M1195_Inflight_AssignmentMap. This name is generated in the **Register Metadata**step for every model in SAS Activity-Based Management.

Table 24.2 *Assignment IMAP schema*

Source Public View: Period				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
EndDate	Number			EndDate

Id	Number	Period.Id		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentId	Number	Period.Id		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
StartDate	Number			StartDate
		Source Public View: Scenario		
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Description	Text			Description
Id	Number	Scenario.ID		ID values - not viewable in the user interface
Name	Text		Name	Name
ParentId	Number	Scenario.ID		ID values - not viewable in the user interface
Reference	Text		Reference	Reference
		Source Public View: AssignmentExt		
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
AllocatedCost	Number		Allocated Cost	Allocated Cost
AssignedIdleQuantity	Number		Assigned Idle Quantity	Assigned Idle Quantity
Cost	Number		Cost	Cost
DestinationId	Number	Account.ID		ID values - not viewable in the user interface
DestinationName	Text		Name	Name
DestinationReference	Text		Reference	Reference
DrivenCost	Number		Driven Cost	Driven Cost
DrivenQuantity	Number		Driven Quantity	Driven Quantity
DriverId	Number	Driver.Id		ID values - not viewable in the user interface
FixedQuantity	Number		Driver Quantity Fixed	Driver Quantity Fixed
FixedWeight	Number		Driver Weight Fixed	Driver Weight Fixed
IdleCost	Number		Idle Cost	Idle Cost

PeriodId	Number	Period.Id		ID values - not viewable in the user interface
QuantityBasic	Number		Driver Quantity Basic	Driver Quantity Basic
QuantityCalculated	Number		Driver Quantity Calculated	Driver Quantity Calculated
ScenarioId	Number	Scenario.ID		ID values - not viewable in the user interface
SourceId	Number	Account.ID		ID values - not viewable in the user interface
SourceName	Text		Name	Name
SourceReference	Text		Reference	Reference
UsedCost	Number		Used Cost	Used Cost
UserIdleQuantity	Number		Idle Driver Quantity UE	Idle Driver Quantity UE
VariableQuantity	Number		Driver Quantity Variable	Driver Quantity Variable
VariableWeight	Number		Driver Weight Variable	Driver Weight Variable
Source Public View: Driver				
Field Name	**Field Type**	**Join Field**	**Properties Reference**	**Model View Columns**
Formula	Text		Driver Formula	Formula
Id	Number	Driver.Id		ID values - not viewable in the user interface
ImplementationType	Text		Driver Type	Driver Type
Name	Text		Driver Name	Driver Name
QuantityType	Text			Driver checkbox: Shared or Unique


IMAP: Assignment Schema
External to IMAP
Key Table : AssignmentExt
Joined Tables
Account Table
Account Table
SourceID
DestinationID
PeriodID
ScenarioID
DriverID
. . . .
All other numeric properties
Period
Scenario
Driver
Note: You do not need to build these additional joins because all of the information about the source and destination accounts is included in the AssignmentExt table.

Part 8

Working with Other SAS Programs

Chapter 25
Using Information Maps

Create Information Maps (Register Metadata)

Information maps are the vehicle for sharing information among SAS programs such as SAS Information Map Studio, SAS Web Report Studio, and SAS OLAP Cube Studio. For example, you can use SAS Web Report Studio to generate account-based reports and assignment-based reports.

Perform the following steps to create information maps:

1. Open the model for which you want to create information maps.
2. Select **Model** ⇨ **Register Metadata**. The Register Metadata dialog box opens.

   Note: The model must have been calculated.

3. If you have not already specified the folder to use on the Metadata Server to use, click **Configure**. The Metadata Server Options dialog box appears.

   Note: You can also open the Metadata Server Options dialog box by selecting **Tools** ⇨ **Metadata Server Options**.

4. Under **Create information maps for reporting**, select the type of map(s) that you want to create.

   - **Account Map**: This option creates an information map from model-specific public database views. This information map contains account-related data that can be used by SAS Web Report Studio to create account-based reports.

- **Assignment Map**: This option creates an information map from model-specific public database views. This information map contains assignment-related data that can be used by SAS Web Report Studio to create assignment-based reports.

5. Click **Create**. Before information maps are created, specific public database views that are associated with the current model are copied to the SAS data library that you specified in the Metadata Server Options dialog box. These public database views are created during model calculation, and they are used to create information maps.

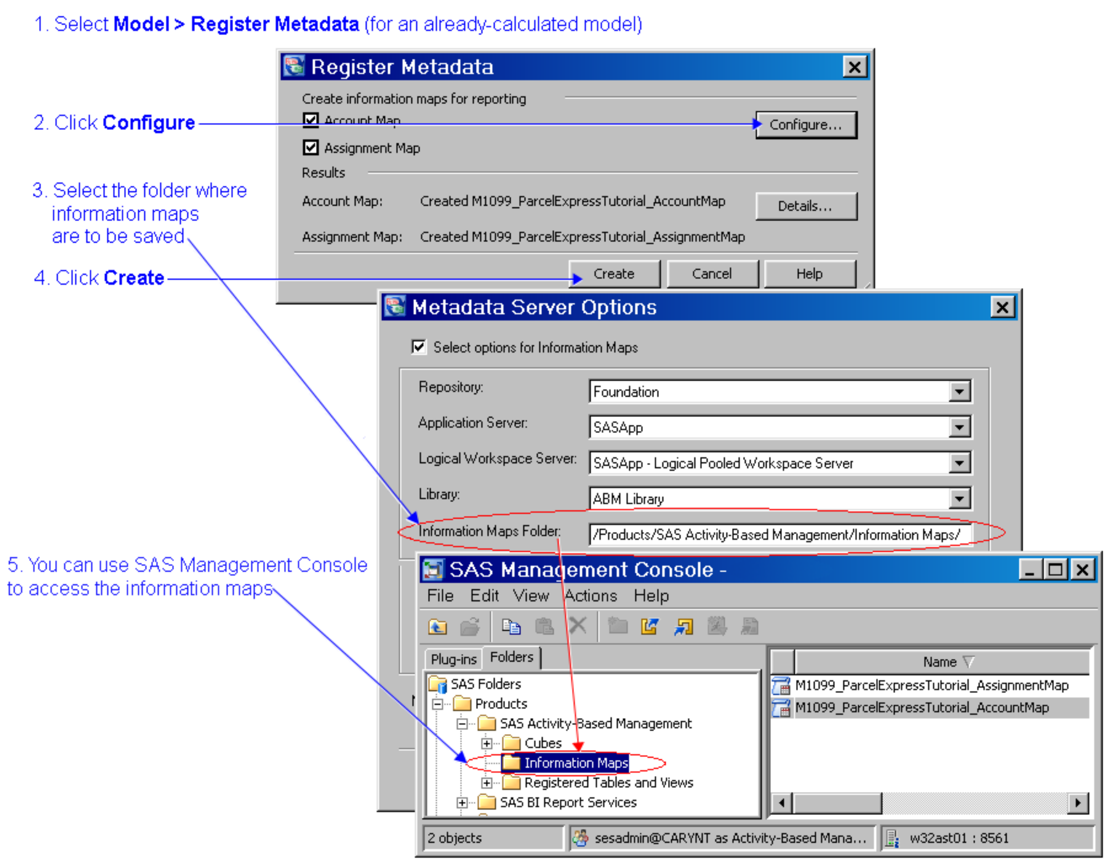
   Map names are automatically generated in the following formats:

   - Account maps: M<InternalModelID>_<ModelName>_AccountMap
   - Assignment maps: M<InternalModelID>_< ModelName >_AssignmentMap

   For example, an account map for a SAS Activity-Based Management model with an internal model ID of 1079 and a model name of Parcel Express is generated as M1079_ParcelExpress_AccountMap

   Note: Each time that you create an information map, the existing information map is deleted and replaced by the new information map.

6. Click **Details** to view the log details of the results. The Register Metadata Results dialog box appears. If the tables are successfully imported, then the information maps are created.



See Also

- "Register Metadata / Metadata Server Options" on page 423
- "Work with Information maps in SAS Information Map Studio" on page 399

- "Work with Information Maps in SAS Web Report Studio" on page 401
- "Use SAS Management Console to Configure for Information Maps" on page 405
- "Change the SAS Metadata Server Connection" on page 410

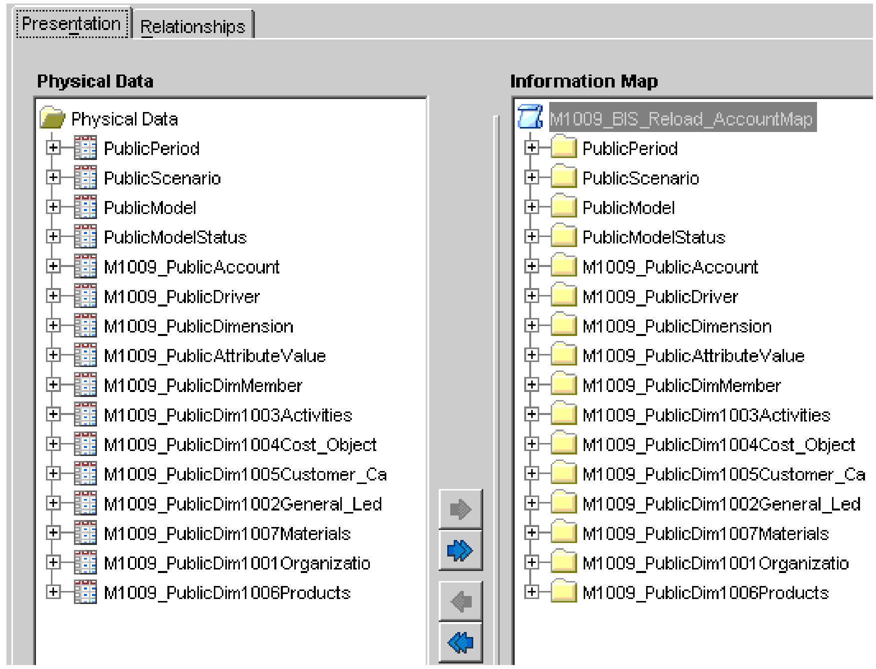
Work with Information maps in SAS Information Map Studio

You can use SAS Information Map Studio to open the predefined account maps and assignment maps that you create in SAS Activity-Based Management. To use these predefined SAS Activity-Based Management information maps in SAS Web Report Studio, you are not required to take any specific action within SAS Information Map Studio. However, you should become familiar with the available content from the account maps and assignment maps because you can define custom SAS Web Report Studio reports based on your SAS Activity-Based Management content.

For information about creating custom information maps, see the SAS Information Map Studio documentation.

Perform the following steps to use SAS Information Map Studio to open the account maps and assignment maps that you create in SAS Activity-Based Management.

1. Open SAS Information Map Studio and create a Metadata Profile by following the steps in the wizard (or open an existing Metadata Profile). You must provide the following information:
   - Name for your Metadata Profile
   - Connection information
   - Repository - Select the Foundation repository (default location for SAS Activity-Based Management information maps)
   - Server, port, user ID, and password
2. In SAS Information Map Studio, specify the data source that contains your SAS Activity-Based Management information map(s):
   a. Select **View ⇨ Metadata Repository Pane** to display the tree view.
   b. Select the Repository specified in the Metadata Server Options dialog box.
   c. Select the Information Maps Folder specified in the Metadata Server Options dialog box. For example: **`/Products/SAS Activity-Based Management/Information Maps/`**.
   d. Select the account map or assignment map that you created in SAS Activity-Based Management. For example: M1079_ParcelExpress_AccountMap. The contents of the information map display.



3. In SAS Information Map Studio, review the table and field contents of the selected map by opening the account map and assignment map folders.

   Measure values (numerical properties from SAS Activity-Based Management) are displayed with ruler icons. Dimensional values (text content from SAS Activity-Based Management) are displayed with the sheets icons.

   Each public source table in SAS Activity-Based Management has both numeric properties and text content, which can ultimately be used in a SAS Web Report Studio report.

4. In SAS Information Map Studio, click the Relationships tab to review the relationship map, which defines how all of the tables are joined into a single information map. All of the relationships are predefined when you register the metadata from SAS Activity-Based Management.

   - In the account map, there are 16 different tables that are joined together to define the single information map. You should not customize these joins.
   - In the assignment map, the process view in SAS Information Map Studio is simpler because all of the necessary joins have been defined and implemented in SAS Activity-Based Management to provide a single contribution table for SAS Information Map Studio to support assignment analysis.

Note: The relationships for the account maps and assignment maps that you create using SAS Activity-Based Management are defined based on the SAS Activity-Based Management model schema and logic. Relationships should not be modified in any way. If you modify these relationships in SAS Information Map Studio, your SAS Web Report Studio reports might return unexpected results.

For information about creating custom information maps based on SAS Activity-Based Management content, see the SAS Information Map Studio documentation at `http://support.sas.com/documentation/onlinedoc/ims/index.html`.

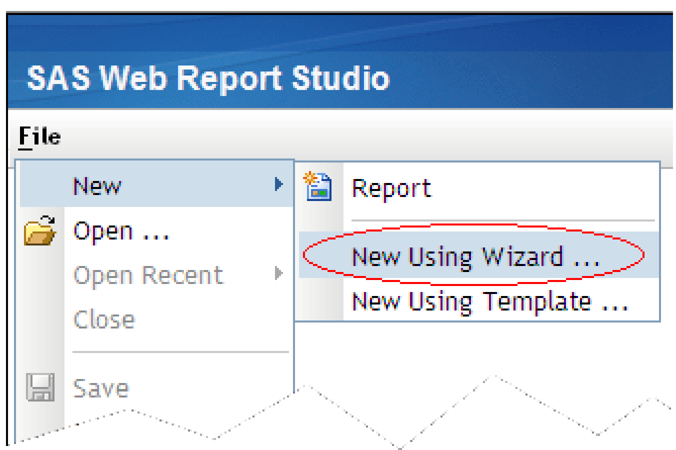
Work with Information Maps in SAS Web Report Studio

You can use SAS Web Report Studio to create reports that are based on the account maps and assignment maps that you create in SAS Activity-Based Management. For information about creating reports, see the **SAS Web Report Studio documentation**.

Perform the following steps to use SAS Web Report Studio:

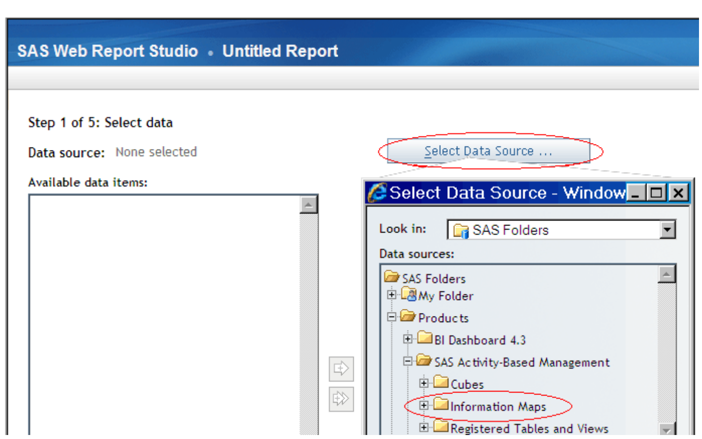
1. Log in to SAS Web Report Studio. Use a login account that is defined in SAS Management Console and that has access to the information maps from the registered metadata.

2. Select **File** ⇨ **New Using Wizard** to launch the SAS Web Report Studio wizard. This wizard will walk you through the basic definition process for building a simple report from your SAS Activity-Based Management information maps.



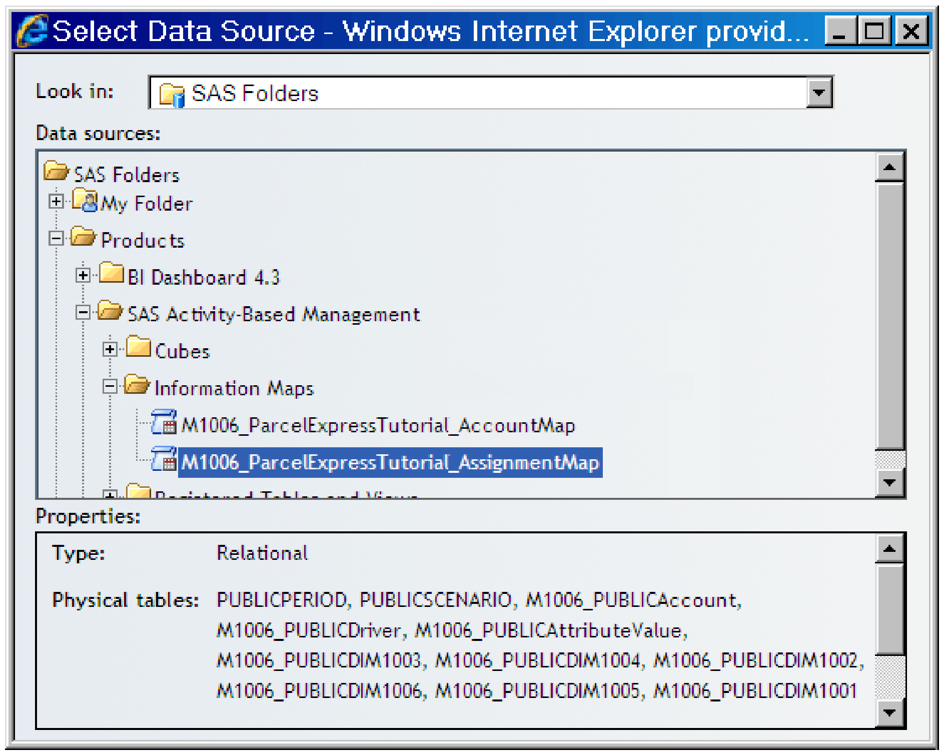
The wizard opens.

3. Click **Select Data Source**.



4. Select the information map that you want to use to build your report. Each SAS Activity-Based Management models information map is noted by the model ID and the model name. A simple report can be based on a single information map.

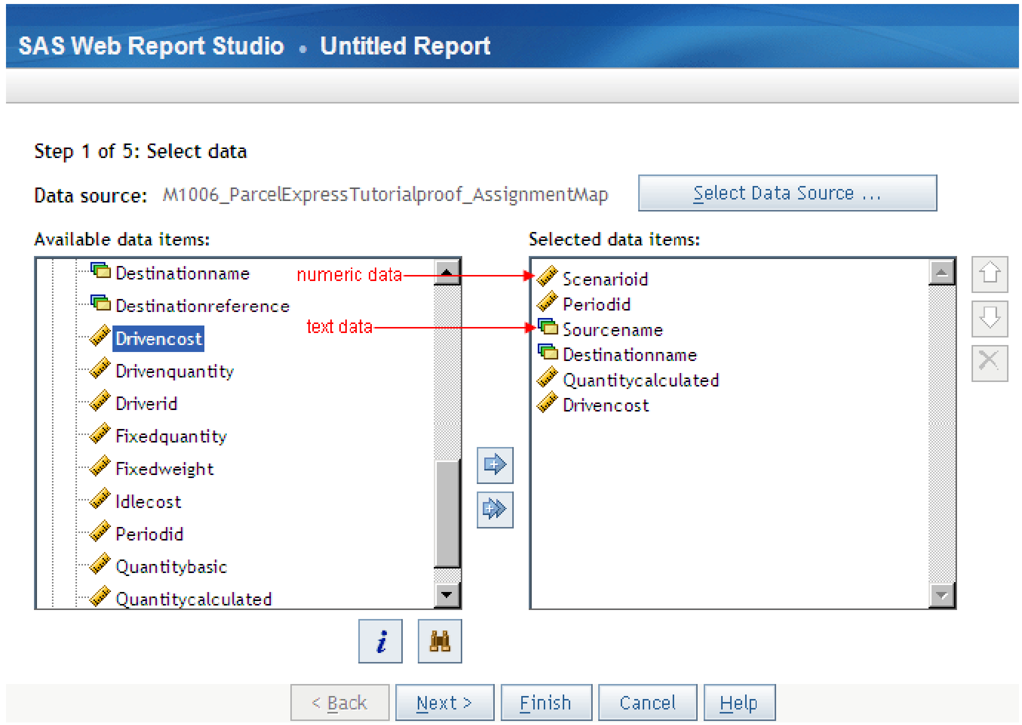
   Note: When you create information maps in SAS Activity-Based Management, the Register Metadata Results dialog box notes the unique names of the information maps that are created. For example, notice the account information map (M1006_ParcelExpressTutorial_AccountMap) and the assignment information map (M1006_ParcelExpressTutorial_AssignmentMap) in the following dialog box.



   Click **Next**.

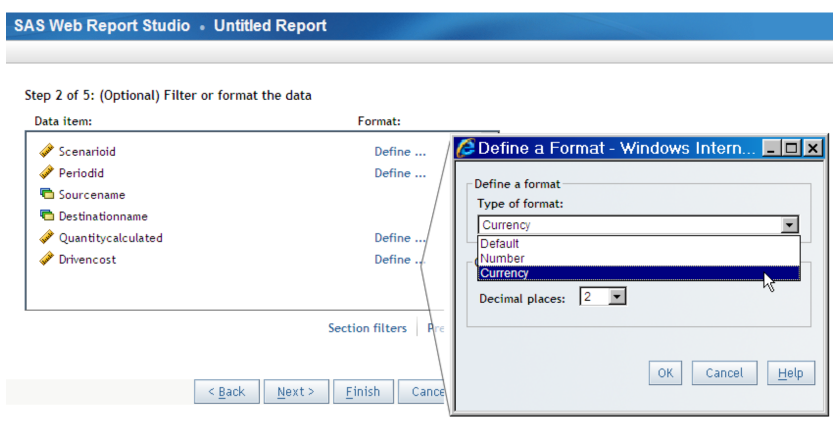
5. Select the fields that you want to display on the report from the Available data items box. Numeric properties are displayed with a ruler icon and text content is displayed with the sheets icon.

   Note: The order in which you select the data items determines how the fields are presented in the final report. When you select your data items, consider your grouping needs for subtotals and page breaks and select the data items that you want to display in groups accordingly.



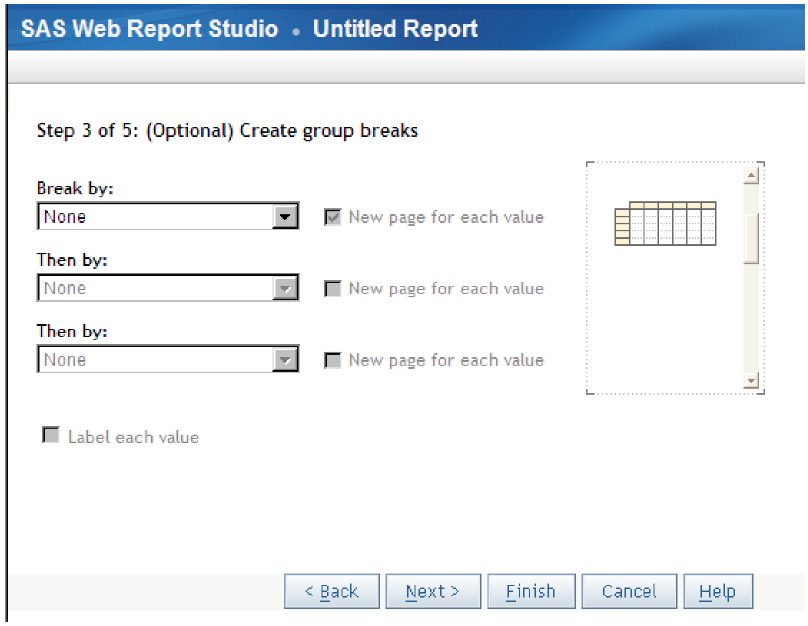
Click **Next**.

6. Specify how you want the data to display and optionally define filters.



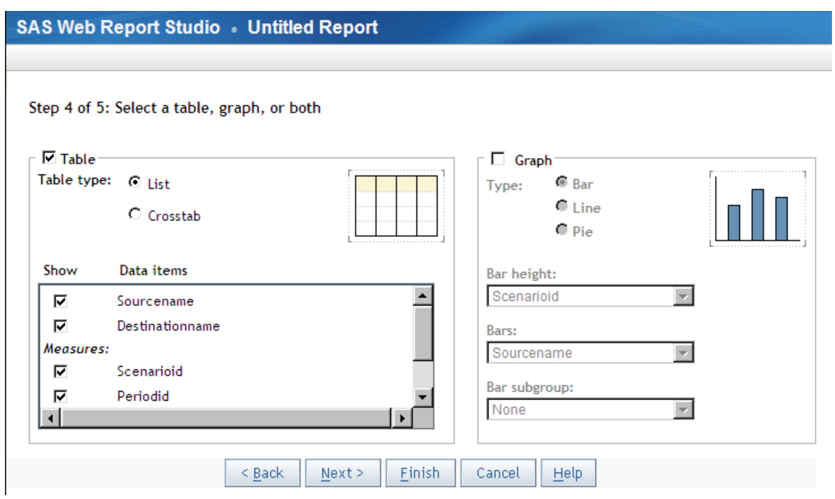
Click **Next**.

7. Create group breaks for data items in the report. Creating group breaks enhances the readability of the final report.



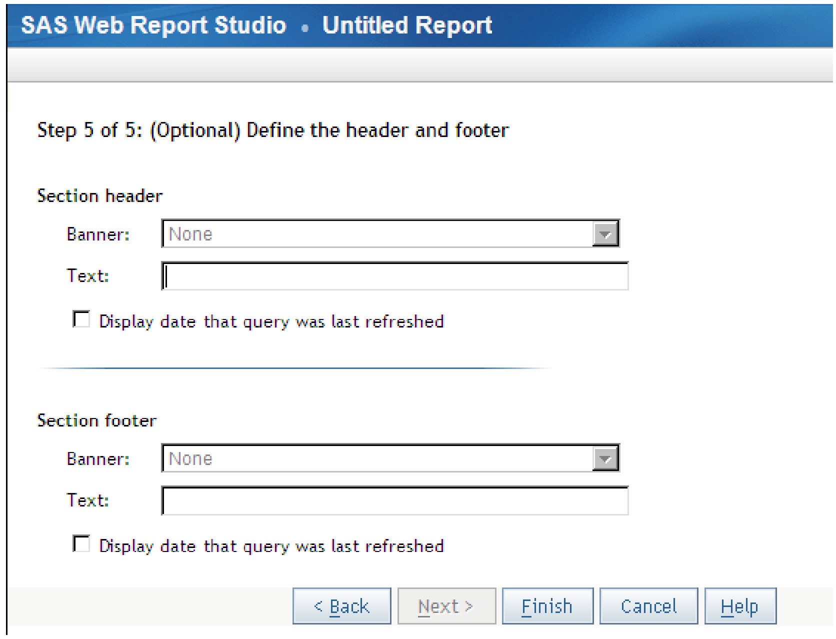
Click **Next**.

8. Specify whether you want you report to display as a Table or as a Graph.



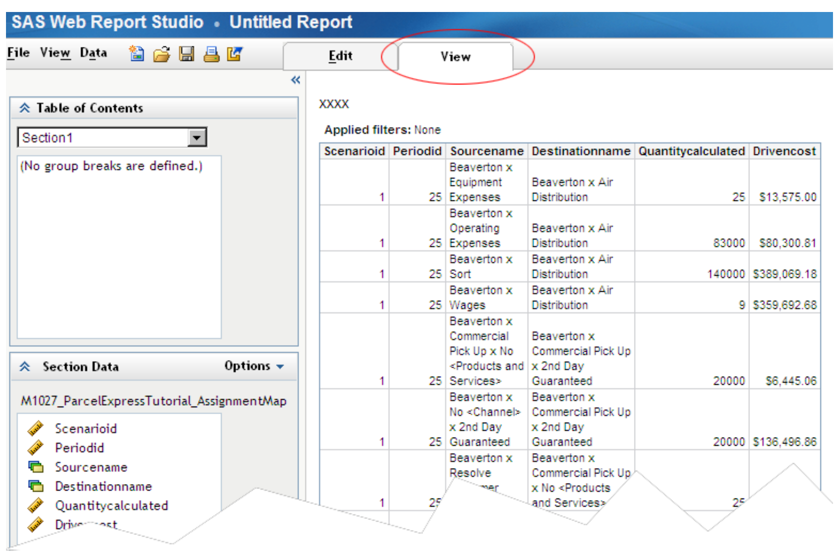
Click **Next**.

9. Define the header and the footer of the report and select Display date that query was last refreshed if you want to display a refresh date on the query.



Click **Finish**.

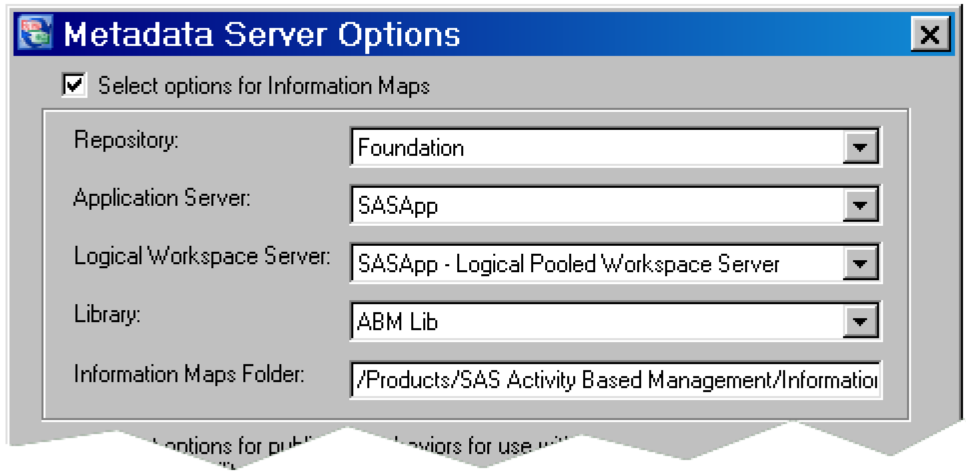
10. Click the **View** tab to view the final report.



11. Save and export the final report to make it available for other users.

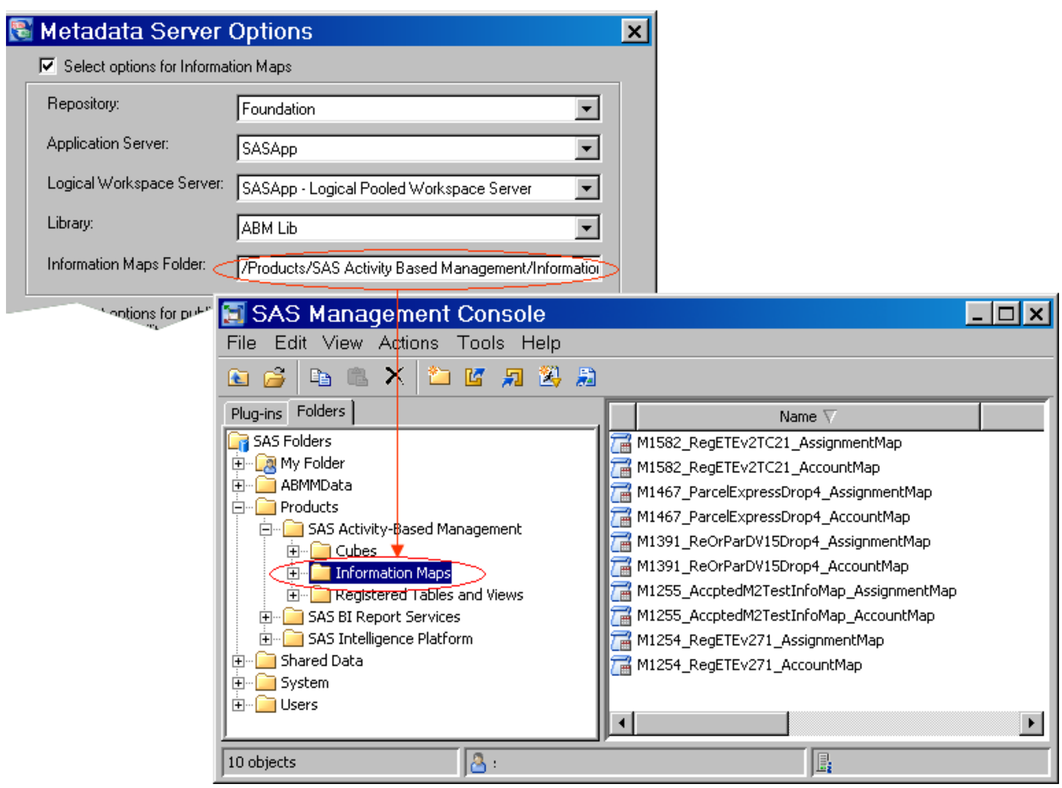
Use SAS Management Console to Configure for Information Maps

The following picture shows an example of options in the Metadata Server Options dialog box for creating information maps.



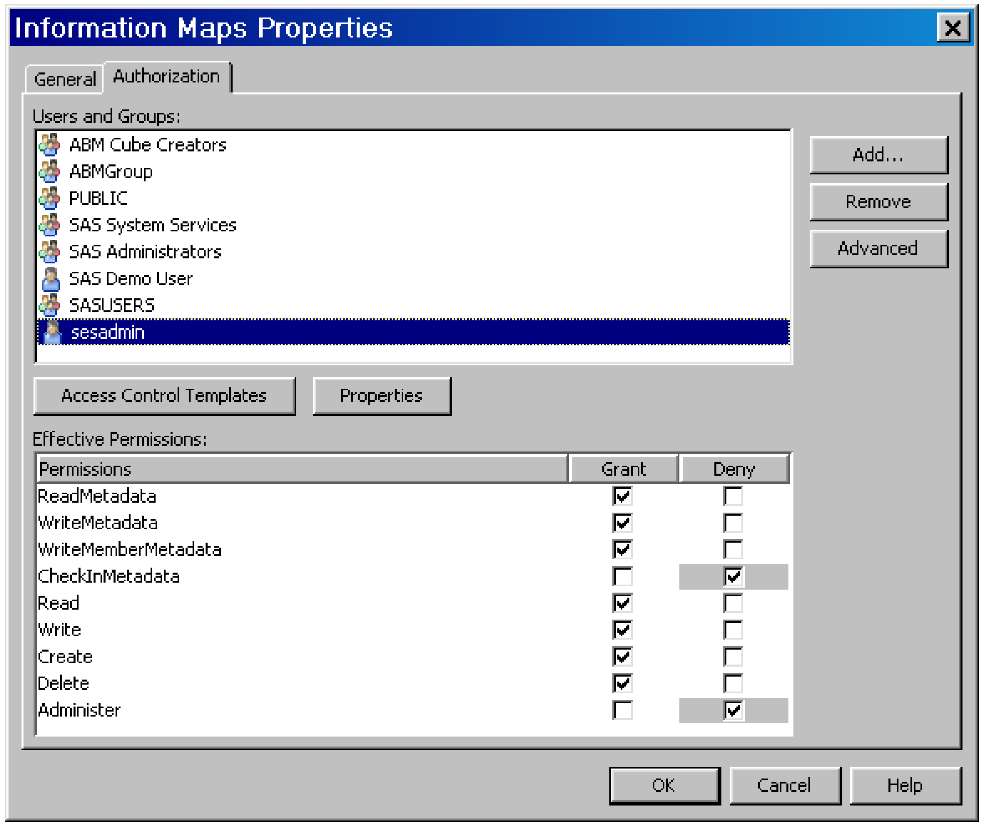
The following steps describe in more detail using SAS Management Console to configure the information map options.

1. Make sure that the default **`Information Maps`** folder exists in the Metadata Server (The folder is created during installation of SAS Activity-Based Management).



2. The user (sesadmin in this example) or group who will be generating information maps should have following permissions for the **`Information Maps`** folder.

   Note: To set the permissions, right-click the **`Information Maps`** folder in SAS Management Console and select **Properties**. Then click the **Authorization** tab.



An error results if the user who generates information maps doesn't have permissions on the target folder.

3. The user who creates information maps should belong to ABMGroup.

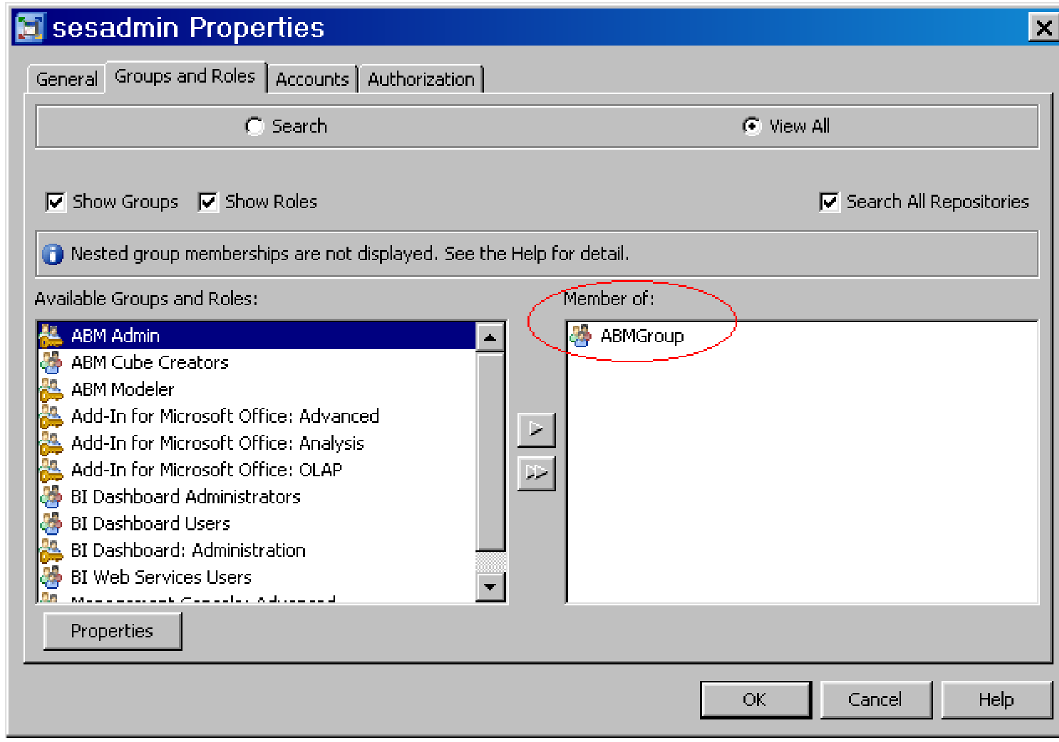
   To add the user to ABMGroup:

   a. Click **User Manager** (in the Plug-ins tab of SAS Management Console).

   b. Right click the user (sesadmin in this example) and select **Properties**.

   c. Click the **Groups and Roles** tab.

   d. Double-click **ABMGroup**.



4. Assign permissions to the library's metadata folder.

   Note: The library should be associated to the ABM database

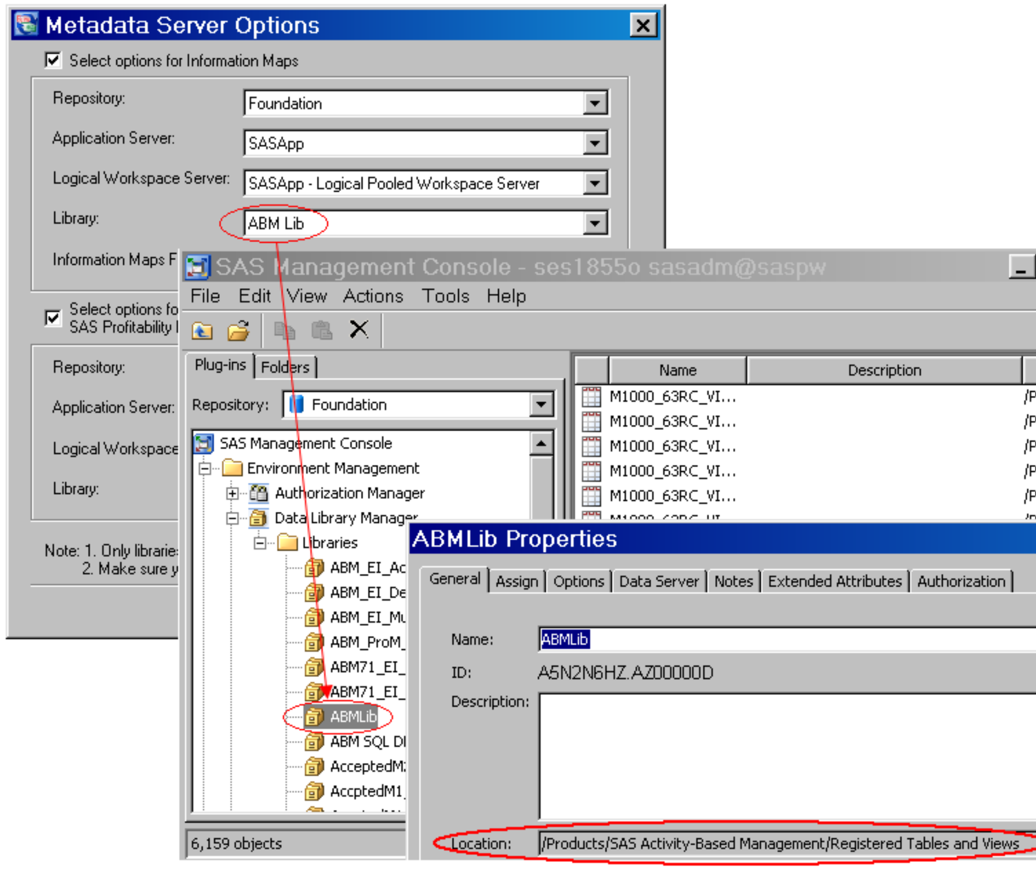
   To determine the library's metadata folder, do the following:

   a. From the SAS Activity-Based Management menu, select **Tools** ⇨ **Metadata Server Options** and note the name of the library.

   b. In the **Plug-ins** tab of SAS Management Console, navigate to the library under **Data Library Manager** ⇨ **Libraries**.

   c. Right-click the library and select **Properties**.

   d. On the **General** tab, note the Location. This is the library's metadata folder.



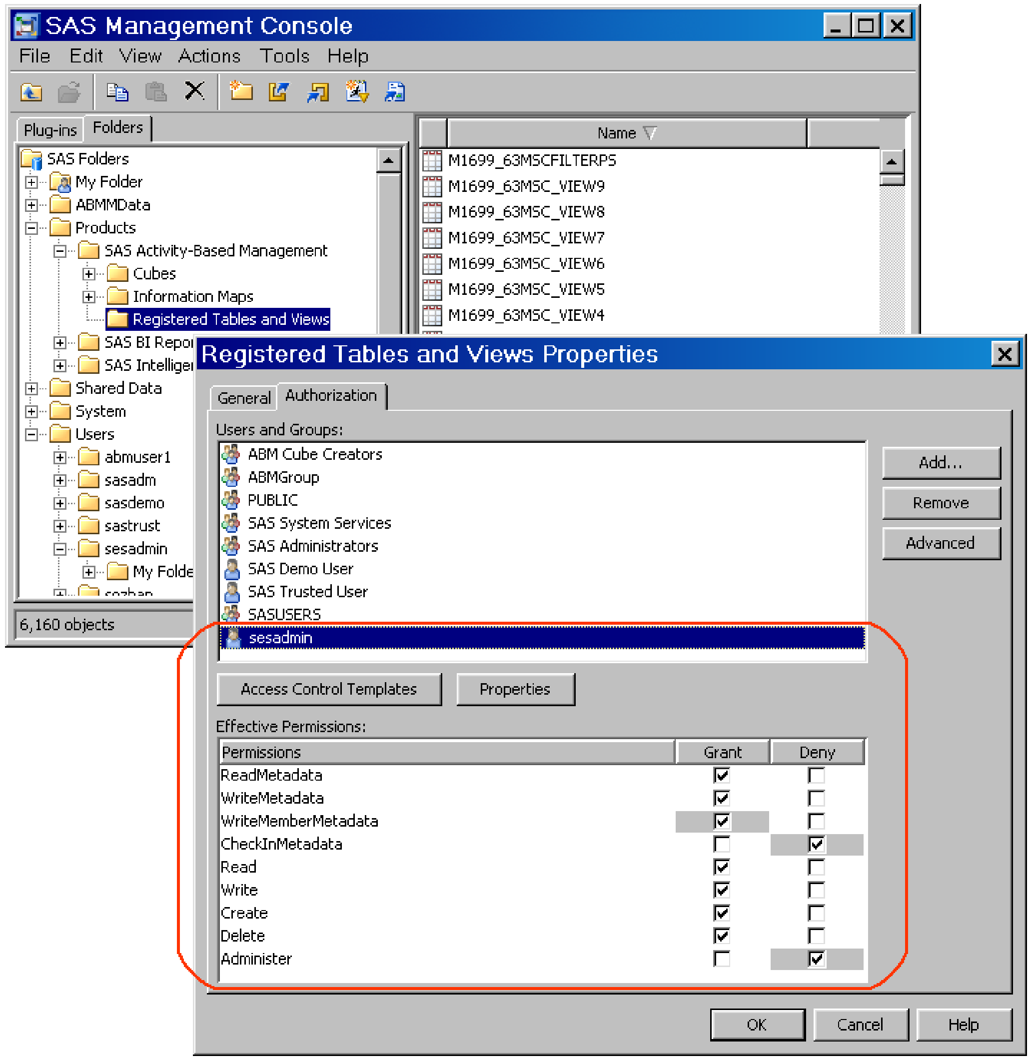
a. In the **Folders** tab of SAS Management Console, navigate to the library's metadata folder.

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

c. Select the **Authorization** tab.

d. Select the user (sesadmin in this example).

e. Set the properties as shown in the following picture.

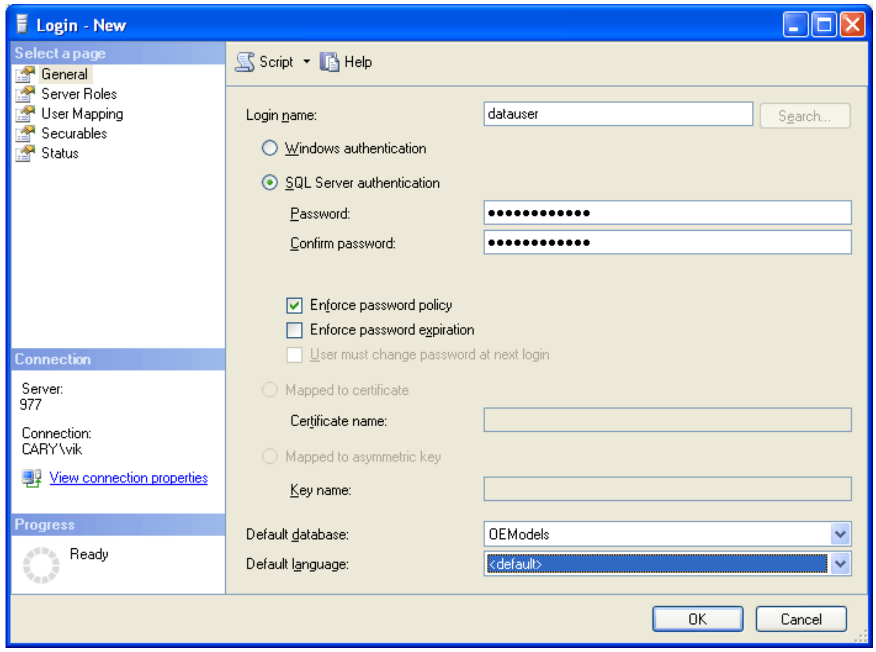


Change the SAS Metadata Server Connection

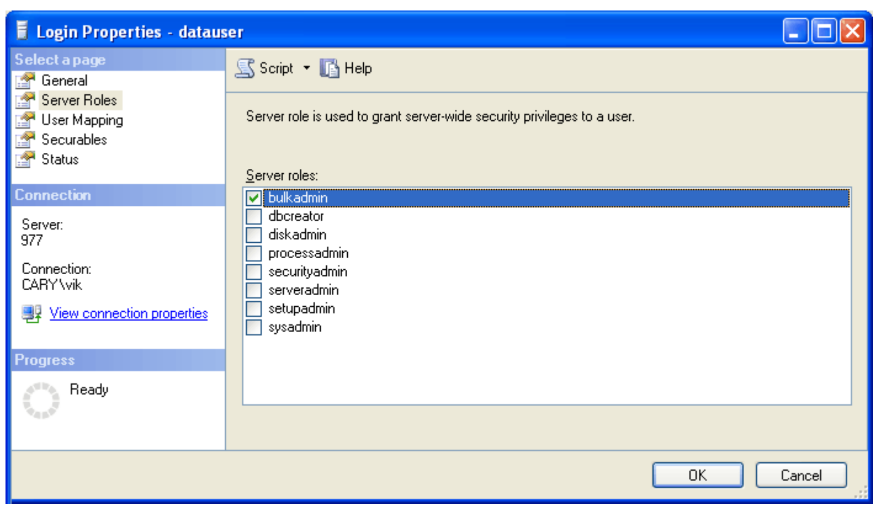
The SAS Metadata Server connection is established during installation of SAS Activity-Based Management. By default, SAS Activity-Based Management uses ODBC to connect to your database containing model information. This help topic describes how to change that connection to use OLE DB if you so choose subsequent to installation.

Note: The following steps assume that you are using Microsoft SQL Server as your SAS Activity-Based Management database.

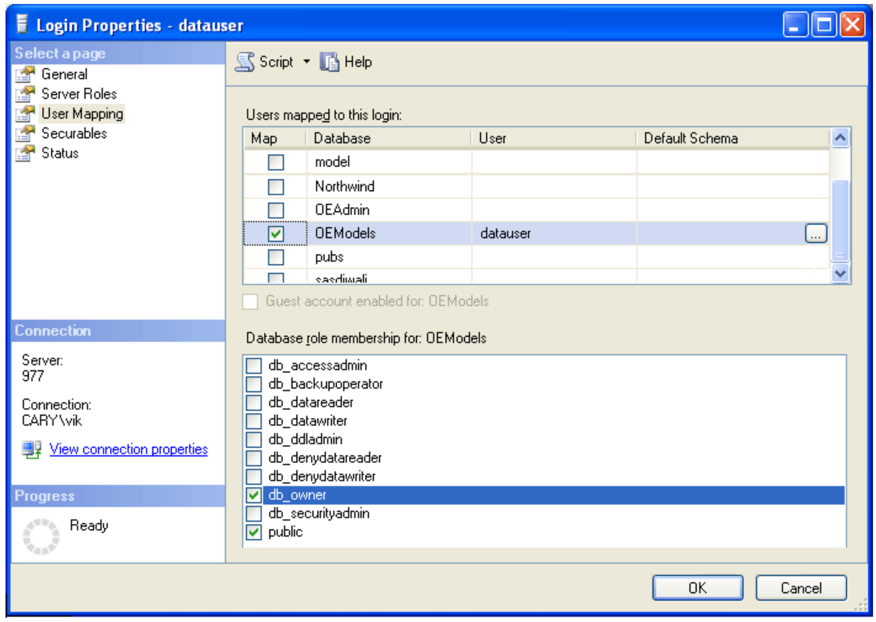
1. Create a new Microsoft SQL Server user. This user should have bulkadmin and db_owner rights to the OEModels database.
   a. In Microsoft SQL Server, add a new user named datauser on the **General** tab, using the SQL Server Authentication information and OEModels as the default Database.



b. Click the **Server Roles** tab and select **bulkadmin**.



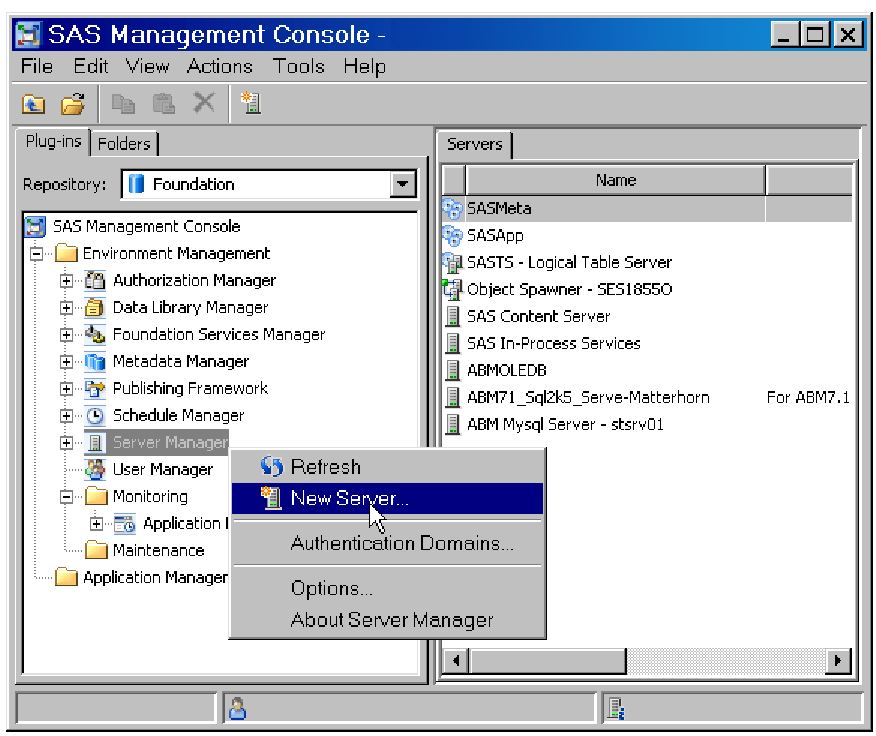
c. On the **User Mapping** tab, select **OEModels** to permit access to that database and select the **db_owner** role.



d. After completing all of the steps of the New Login Wizard, ensure that the db_owner user is listed with the proper permissions.

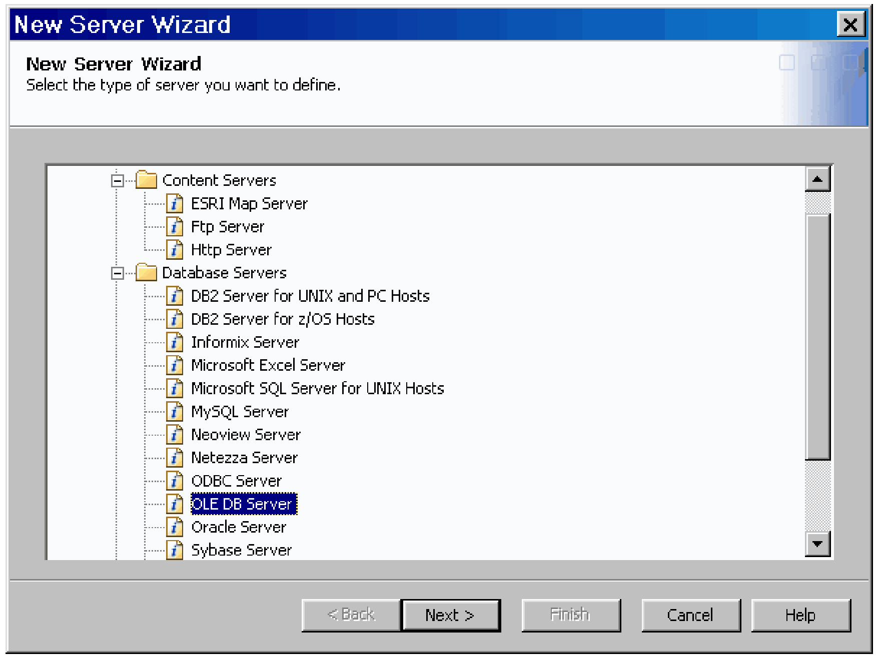
2. Define your SAS Activity-Based Management database server in the metadata on the SAS Metadata Server:

   a. Log in to SAS Management Console as the sasadm user (unrestricted user) and right-click **Server Manager** in the tree view.

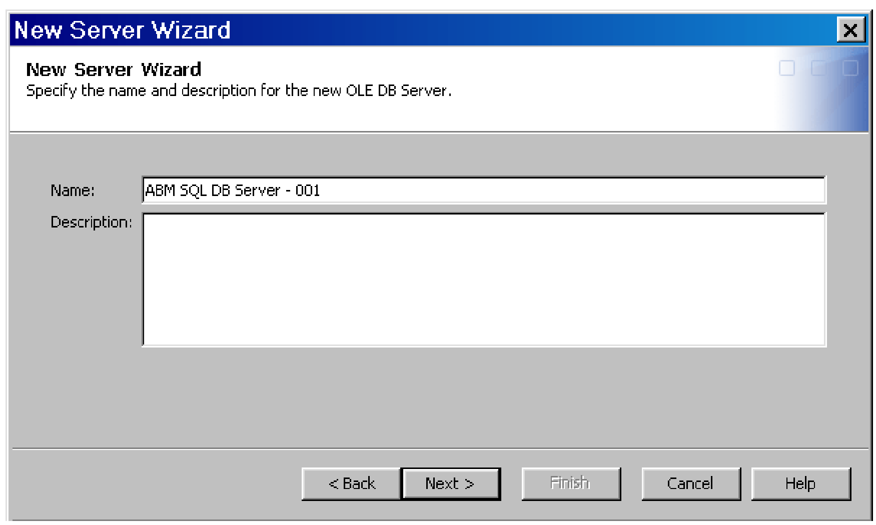


   b. Select **New Server**. The New Server Wizard appears.

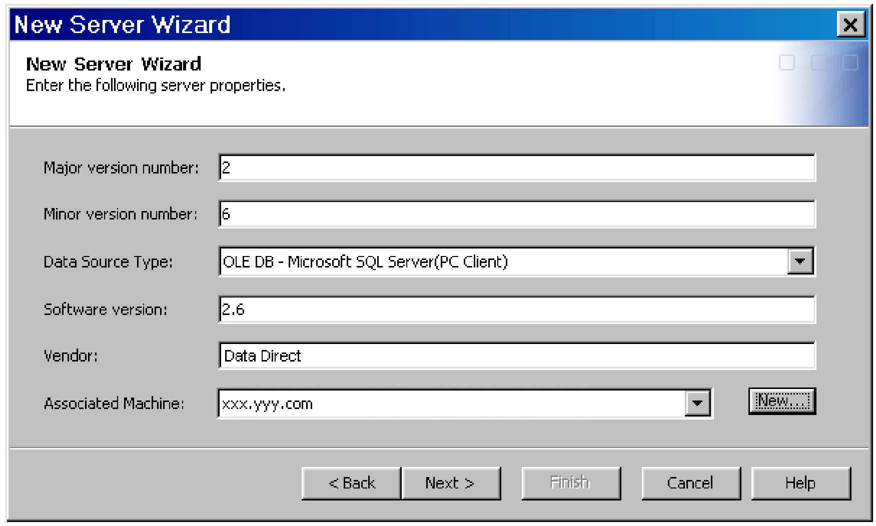
   c. Select **OLE DB Server** in the Database Servers folder and click **Next**.



d. In the **Name** field, type ABM SQL DB Server - 001 to name your SAS Activity-Based Management database server and click **Next**.

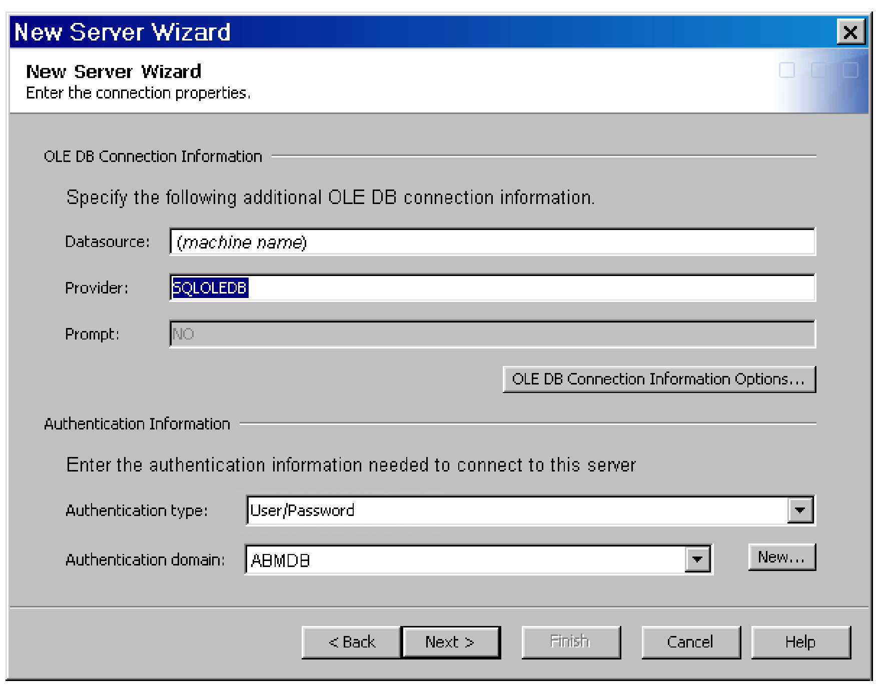


e. Specify the information concerning your database as shown in the picture below, and then click **Next**.



f. Enter the connection properties for the new server:

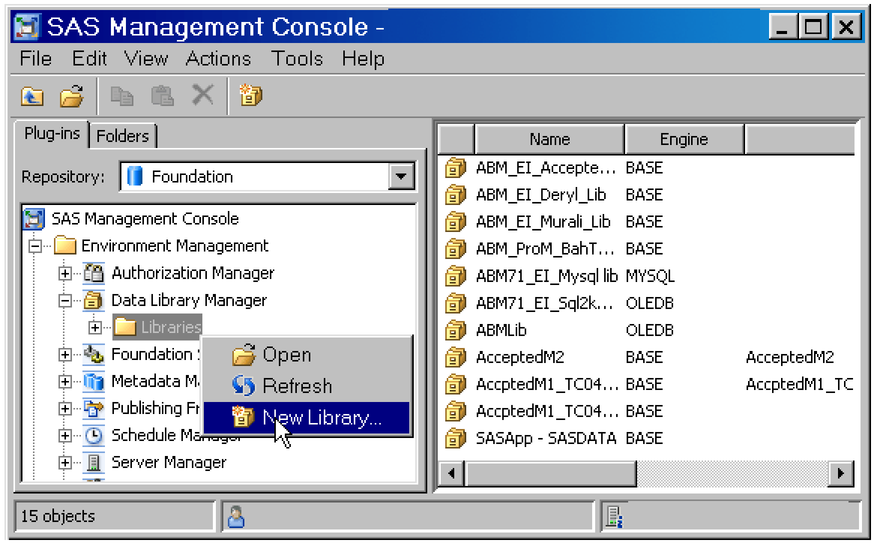
- In the **Datasource** field, type the name of your SAS Activity-Based Management host machine.
- In the **Provider** field, type SQLOLEDB.
- Click **New** and add datauser ABMSQL as the Authentication Domain.
- Click **Next** and review your settings.
- Click **Finish** to create the new server. You should now have a database server named ABM SQL DB Server - 001 on the SAS Metadata Server.



3. Define a new database library in the metadata on the SAS Metadata Server:

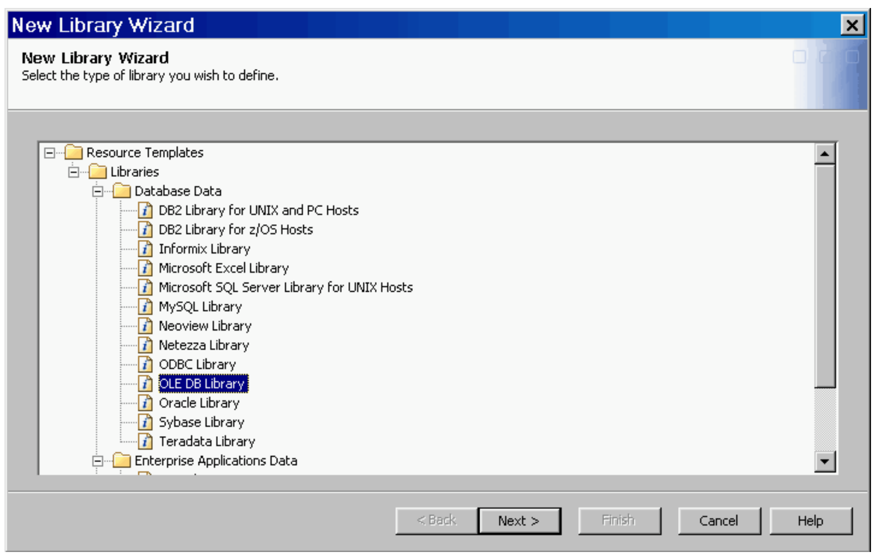
Note: On installation, SAS Activity-Based Management creates an ODBC database library named ABM Library. Because you cannot change the engine from ODBC to OLE DB, you should delete this library and create a new library with the same name but using OLE DB instead of ODBC.

a. Log in to SAS Management Console as the sasadm user (unrestricted user) and right-click **SAS Libraries** under Data Library Manager.



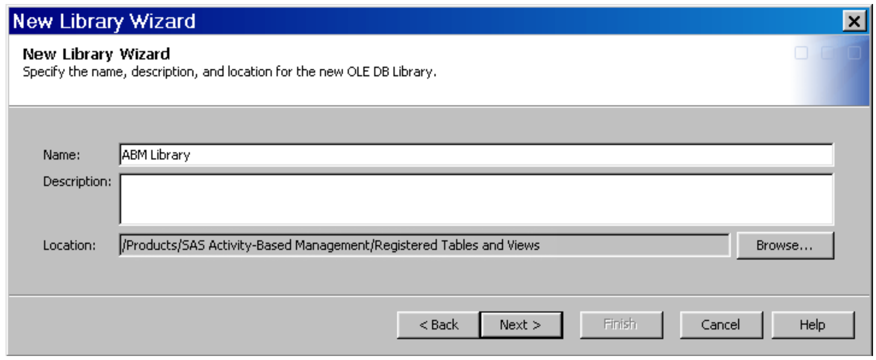
b. Select **New Library**. The New Library Wizard appears.

c. Select **OLE DB Library** in the `Database Data` folder and click **Next**.

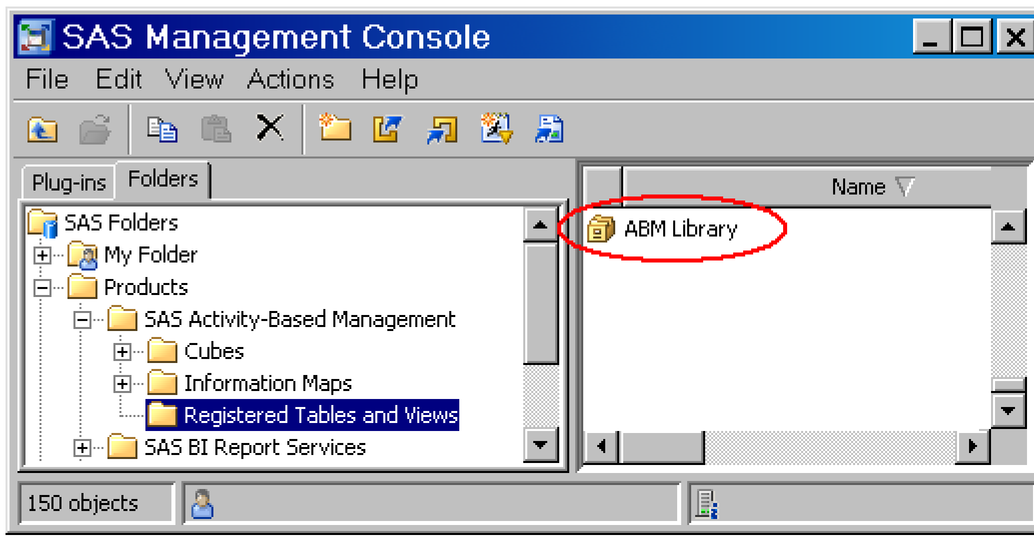


d. In the **Name** field, type ABM Library.

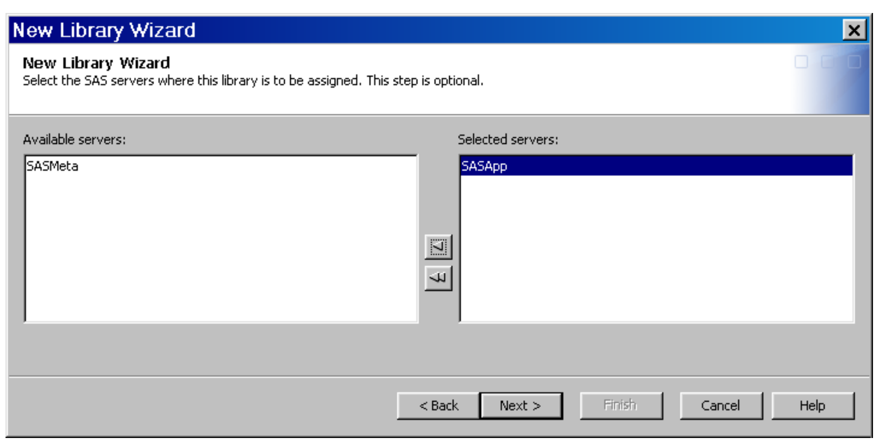
In the **Location** field, type or browse to **`/Products/SAS Activity-Based Management/Registered Tables and Views`**, and then click **Next**.



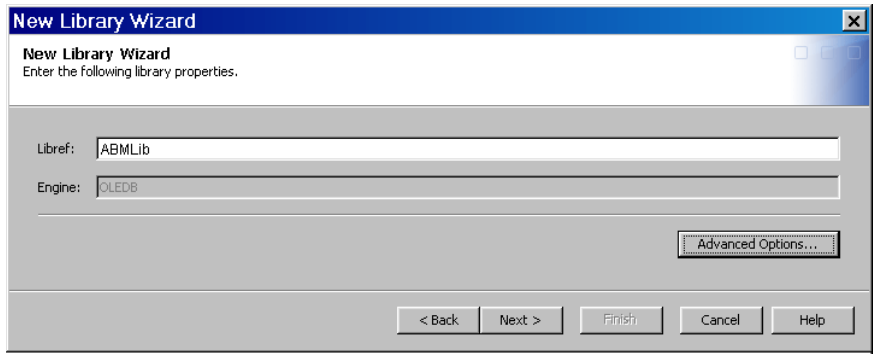
Note: The **Location** field specifies the folder in the Folders tab that SAS Management Console uses to store information about your library.



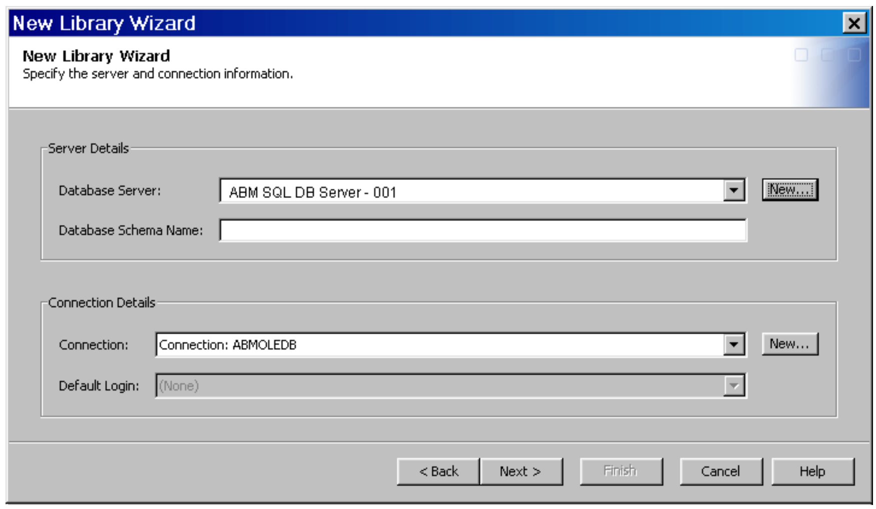
e. Select SASApp as the SAS server, and then click **Next**.



f. In the **Libref** field, type ABMLib as the libref of your database library and click Next.



g. Select the Database Server, ABM SQL DB Server - 001, that you defined in Step 2 and click **Next**.



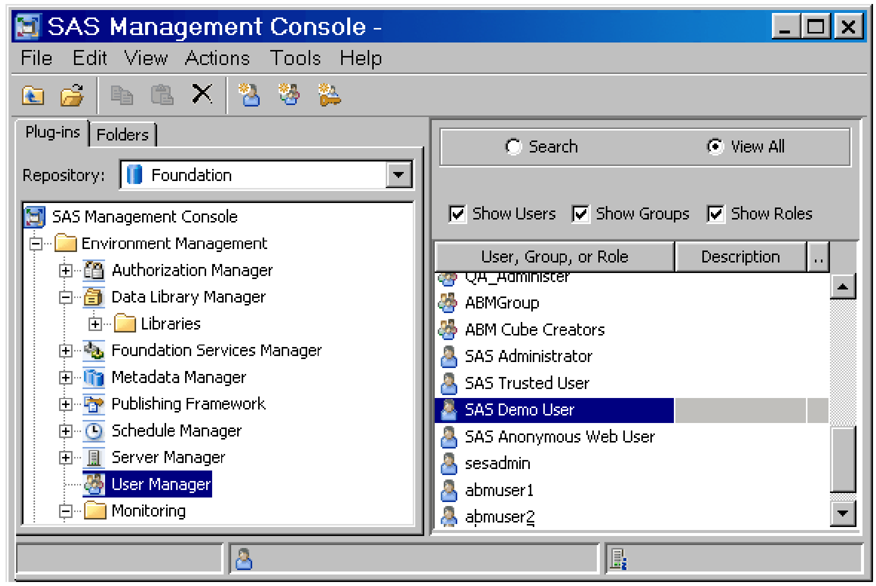
h. Review your settings and click **Finish** to create the new database library.

At this point, you must have available a metadata user who has access to the SAS Activity-Based Management database (Microsoft SQL Server in this case). The following steps use SAS Demo User with the ID, sasdemo, as a working example. SAS Demo User is used by the SAS Activity-Based Management client application to access the SAS Metadata Server, and SAS Demo User must now also access the SAS Activity-Based Management database (Microsoft SQL Server) -- this means that SAS Demo User requires two login accounts.

Note: You should not modify the SAS Demo User unless you are certain that you will not need it for other uses. The following instructions recommend changes that alter SAS Demo User out of its default state.

4. Add the Microsoft SQL Server login information to SAS Demo User:

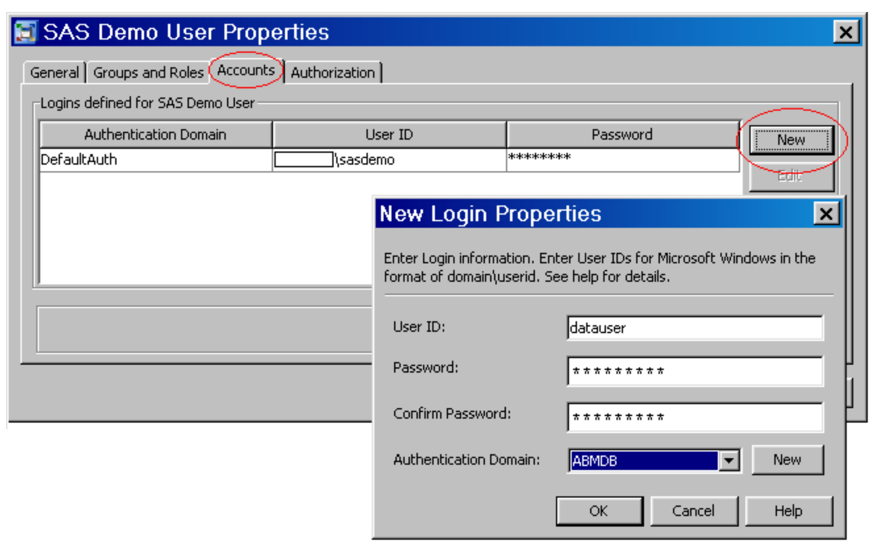
a. Log in to SAS Management Console as the sasadm user (unrestricted user) and selectUser Manager in the tree view.



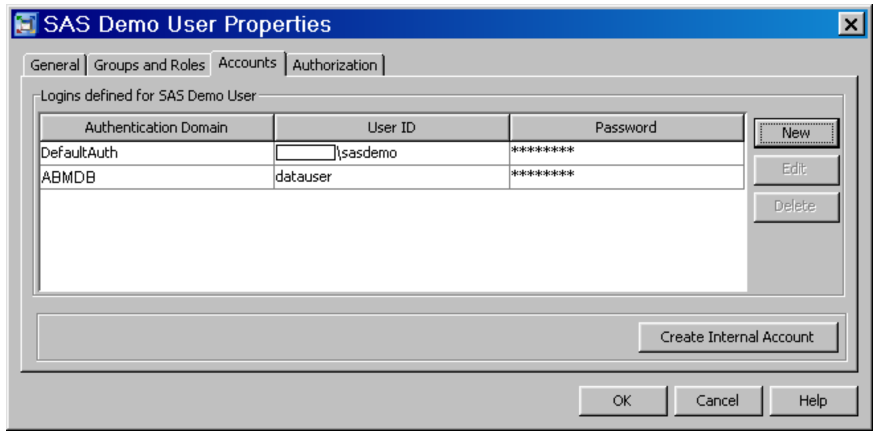
b. Double-click **SAS Demo User** and click the **Accounts** tab.

c. Click **New** and add the login information for SAS Demo User:

- Type the Microsoft SQL Server User ID (datauser) and Password.
- Select the Authentication Domain that you associated with your ABM SQL DB Server - 001 server in Step 2F.
- Click **OK** to close the New Login Properties dialog box.



- Review the two login accounts for SAS Demo User and click **OK** to close the SAS Demo User Properties dialog box.

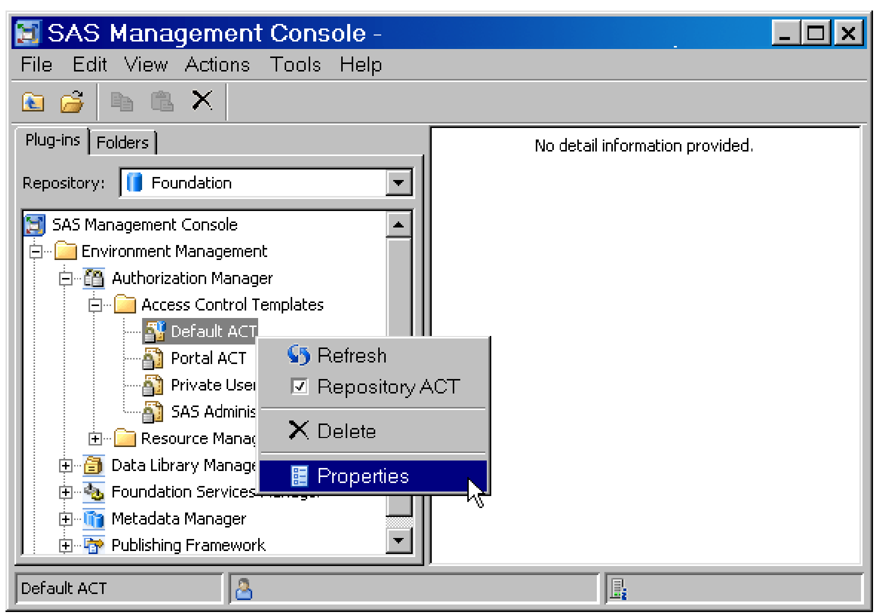


5. Set up the database library.

   Note: The SAS Demo User requires permission to write metadata to the Foundation repository and requires permission to create data in the database library as an application user.

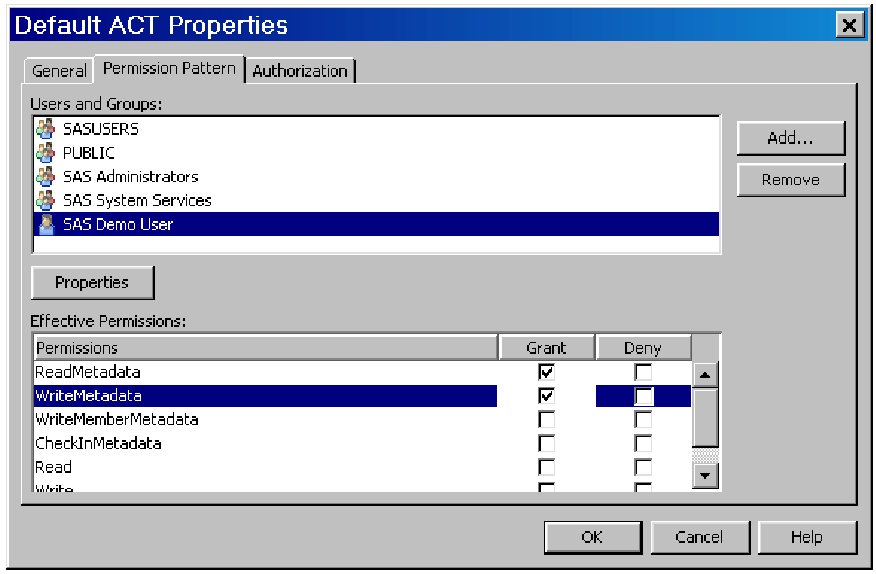
   a. Log in to SAS Management Console as the sasadm user (unrestricted user).

   b. Navigate to **Environment Management** ⇨ **Authorization Manager** ⇨ **Access Control Templates** in the tree view and right-click **Default ACT**.

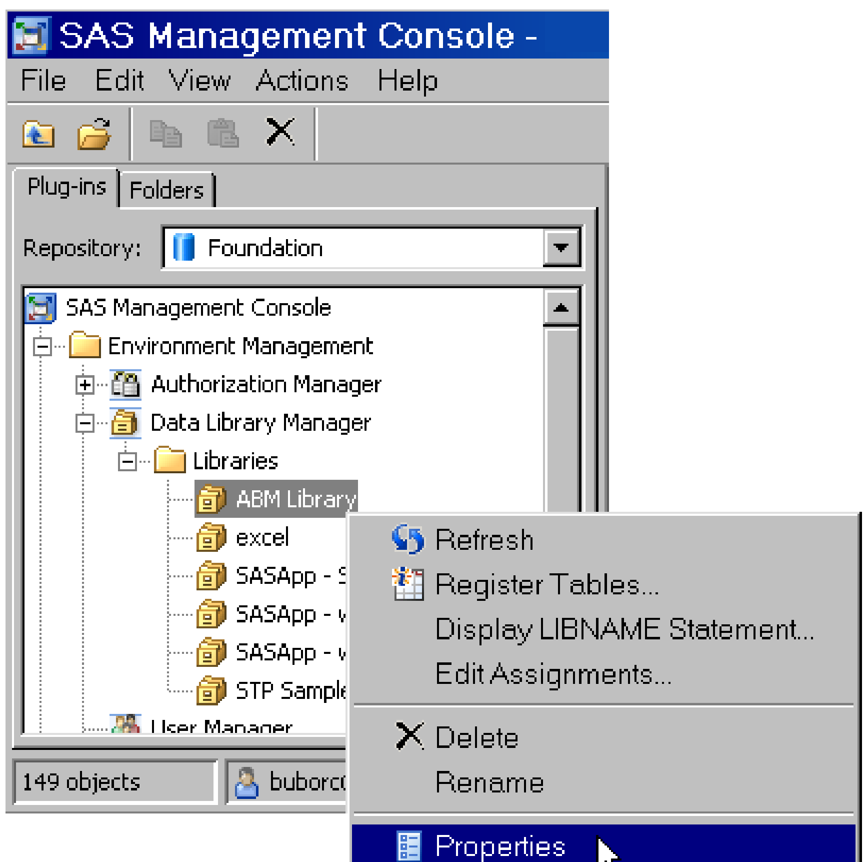


   c. Select **Properties** to open the Default ACT Properties dialog box and do the following:

   - Click the **Permission Pattern** tab and add SAS Demo User to the Users and Groups list.
   - Select the **Grant** check box next to the ReadMetadata and WriteMetadata permissions items.
   - Click **OK** to close the Default ACT Properties dialog box.

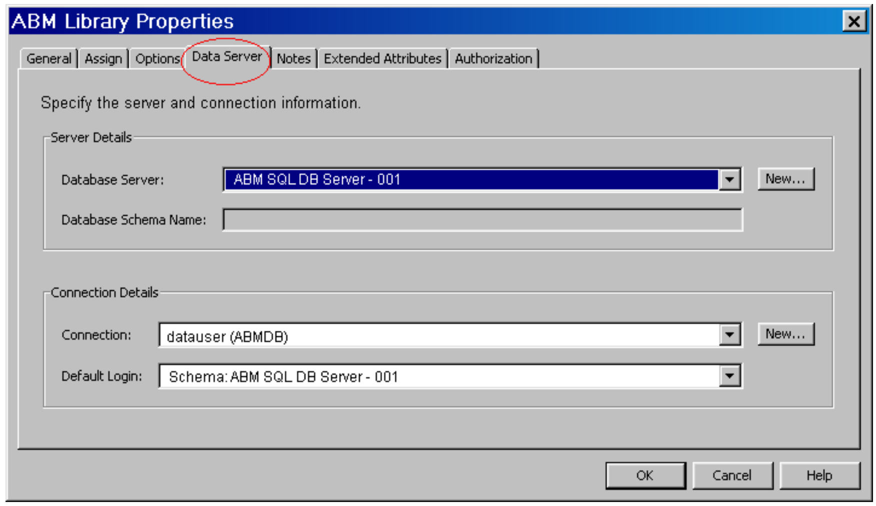


d. In the tree view of the **Plug-ins** tab, right-click the **ABM Library** database library (under **Data Library Manager** ⇨ **Libraries**) that you created in Step 3D and select **Properties**.



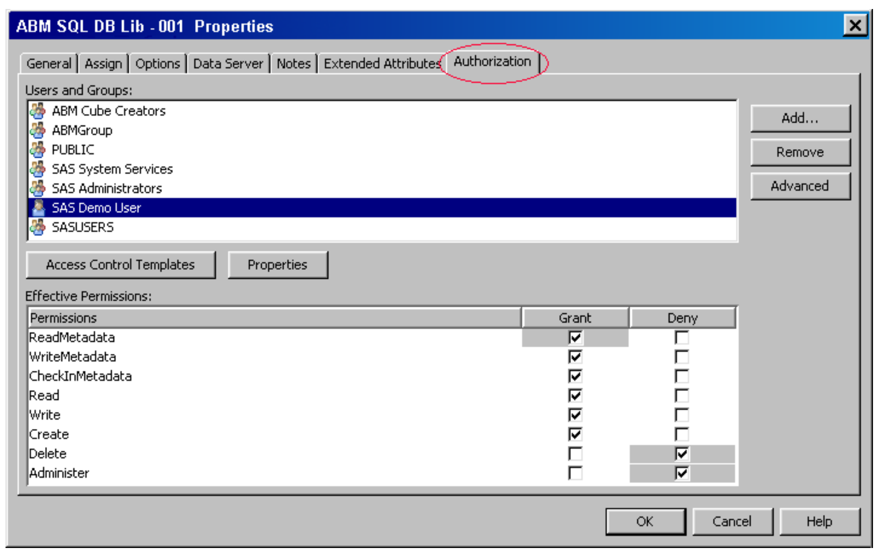
The ABM Library Properties dialog box opens.

e. Click the **Data Server** tab and select the Microsoft SQL Server user that you specified in Step 4C in the Default Login field on the **Data Server** tab.



f. Click the **Authorization** tab and do the following:

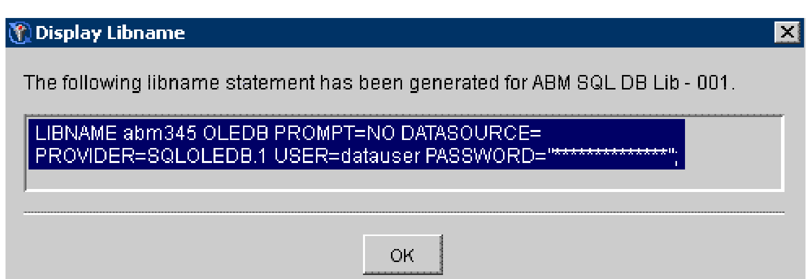
- Add SAS Demo User to the Names list.
- Select the **Grant** check box next to the ReadMetadata, WriteMetadata, CheckInMetadata, Read, Write, and Create permissions items.
- Click **OK** to close the ABM SQL DB Lib - 001 Properties dialog box.



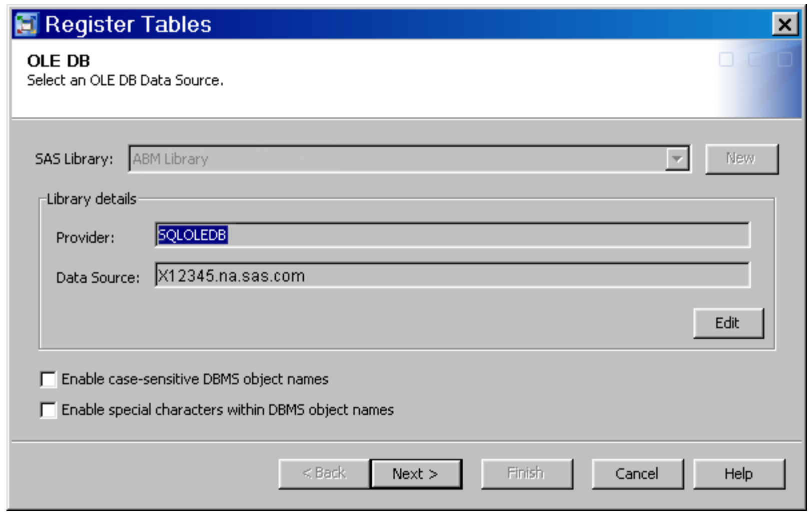
6. Test the database library.

Note: To test the database library, you must log off of SAS Management Console as the sasadm user (unrestricted user) and log in as SAS Demo User (sasdemo).

a. Right-click the database library and select **Display Libname**. Review the LIBNAME statement that includes your settings and click **OK**.

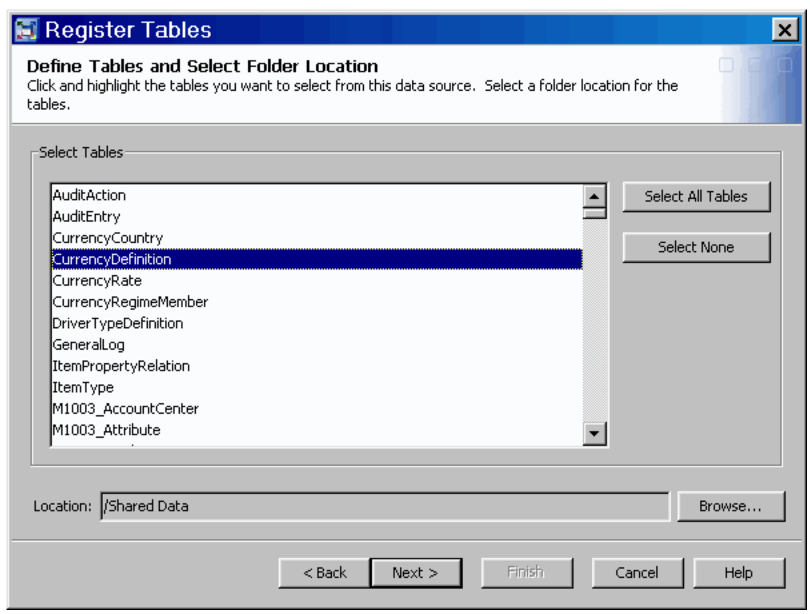


b. Right-click the database library and select **Register Tables**.

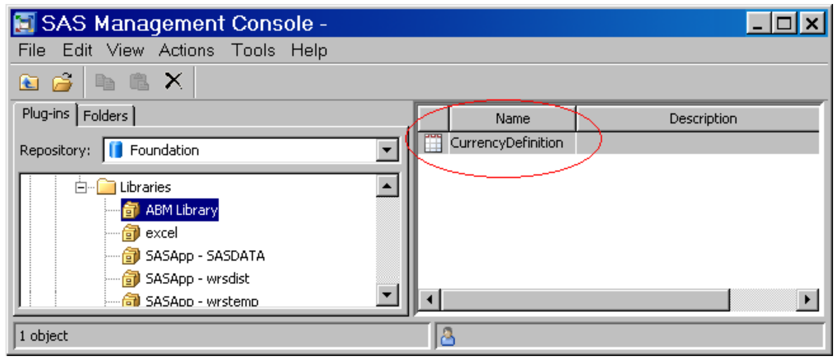


Then click **Next**. You should now be connected to the SAS Activity-Based Management Microsoft SQL Server database.

c. Select a table, such as CurrencyDefinition, from the **Select Tables** list and click **Next** .



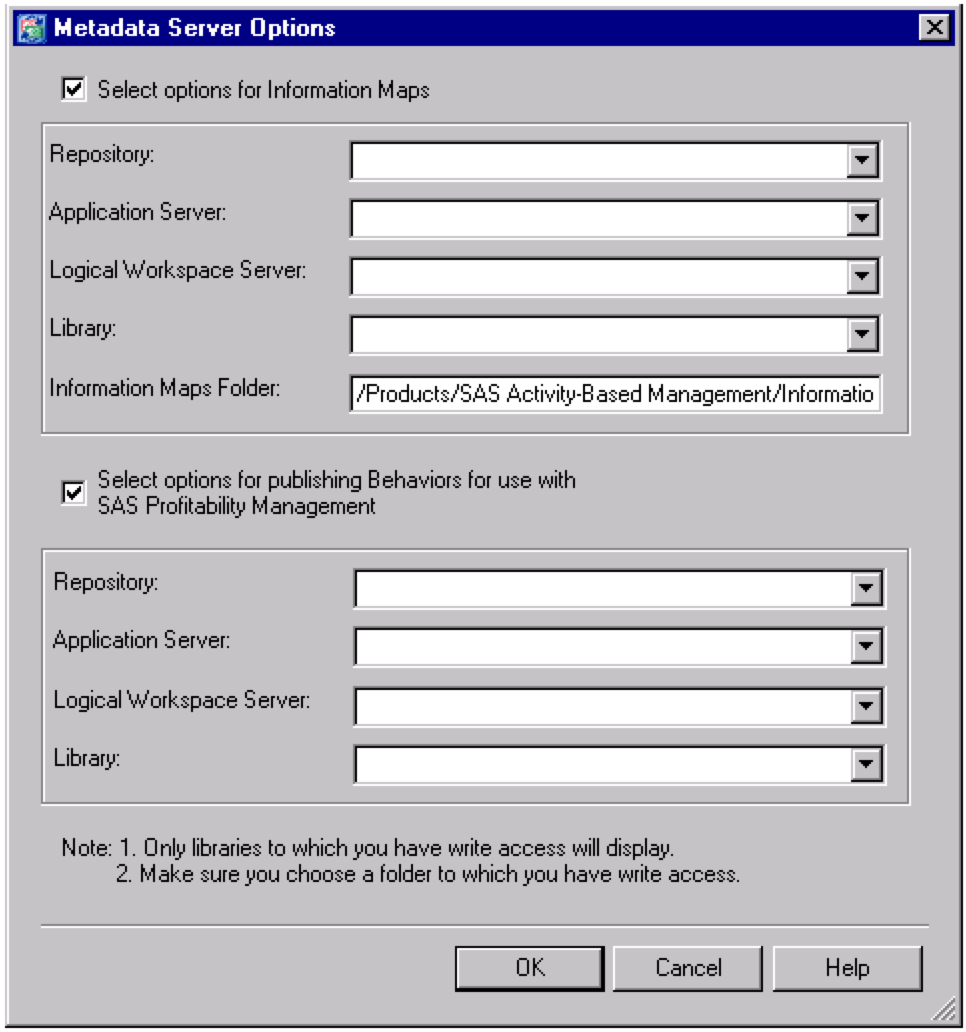
d. Click **Finish**. You should see a new entry named CurrencyDefinition in the database library, which indicates that the database library is valid and in working condition.



Register Metadata / Metadata Server Options

About the Register Metadata and Metadata Server Options Functions

The availability of these features depends on your permissions.



For more information on SAS Information Maps and on SAS Profitability Management, see the chapter on “Working with Other SAS Programs” in the *SAS Activity-Based Management Data Administration Guide* available from the Help menu and also from `http://support.sas.com/documentation/onlinedoc/abm/`.

Register Metadata

About Registering Metadata

Use the Register Metadata dialog box to create information maps for a calculated model.

How to Access the Register Metadata Dialog Box

Select **Model** ⇨ **Register Metadata**.

Note: The model must have been calculated and must currently be open.

Metadata Server Options

About the Metadata Server Options

Use the Metadata Server Options dialog box to specify where on the Metadata Server information maps and SAS Profitability Management behaviors are stored — and to specify what programs are used to create them.

Note: The settings displayed in the Metadata Server Options dialog box are determined during installation of SAS Activity-Based Management. However, you can use the Metadata Server Options dialog box to change these settings after installation.

How to Access the Metadata Server Options Dialog Box

Do either of the following:

- Select **Tools** ⇨ **Metadata Server Options**.
- Click **Configure** on the Register Metadata window.

Chapter 26
SAS Profitability Management

Specify the SAS Profitability Management Input Library

The Profitability Management input library is the library on the Metadata Server where the behavior table is created when you publish behaviors. A SAS Profitability Management input library contains the model's behavior table, dimension tables, transaction tables, report hierarchies, and report layouts.

The behavior table contains the accounts that you have marked as behaviors with the isBehavior attribute. After publishing behaviors, you can open SAS Profitability Management and use with a Profitability Management model the behavior table that you created.

Note: Do not publish to the SAS Profitability Management data library. The data library contains a list of all the models in a SAS Profitability Management installation. And, it stores pointers to information residing in each model's input library.

Note: For more information on creating an input library, see “Create a SAS Profitability Management Input Library” on page 436.

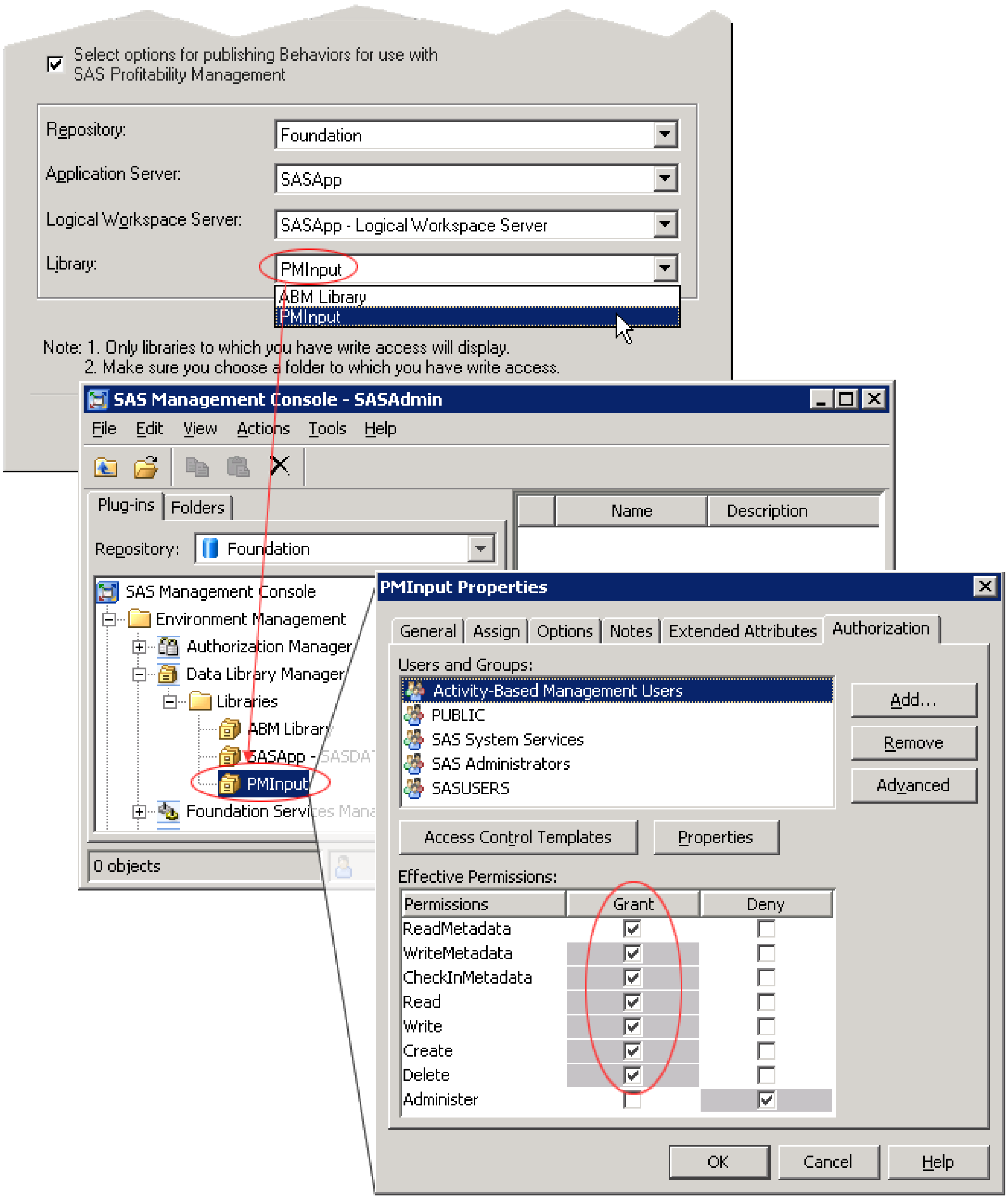
To specify the SAS Profitability Management input library:

1. Select **Tools** ⇨ **Metadata Server Options** .

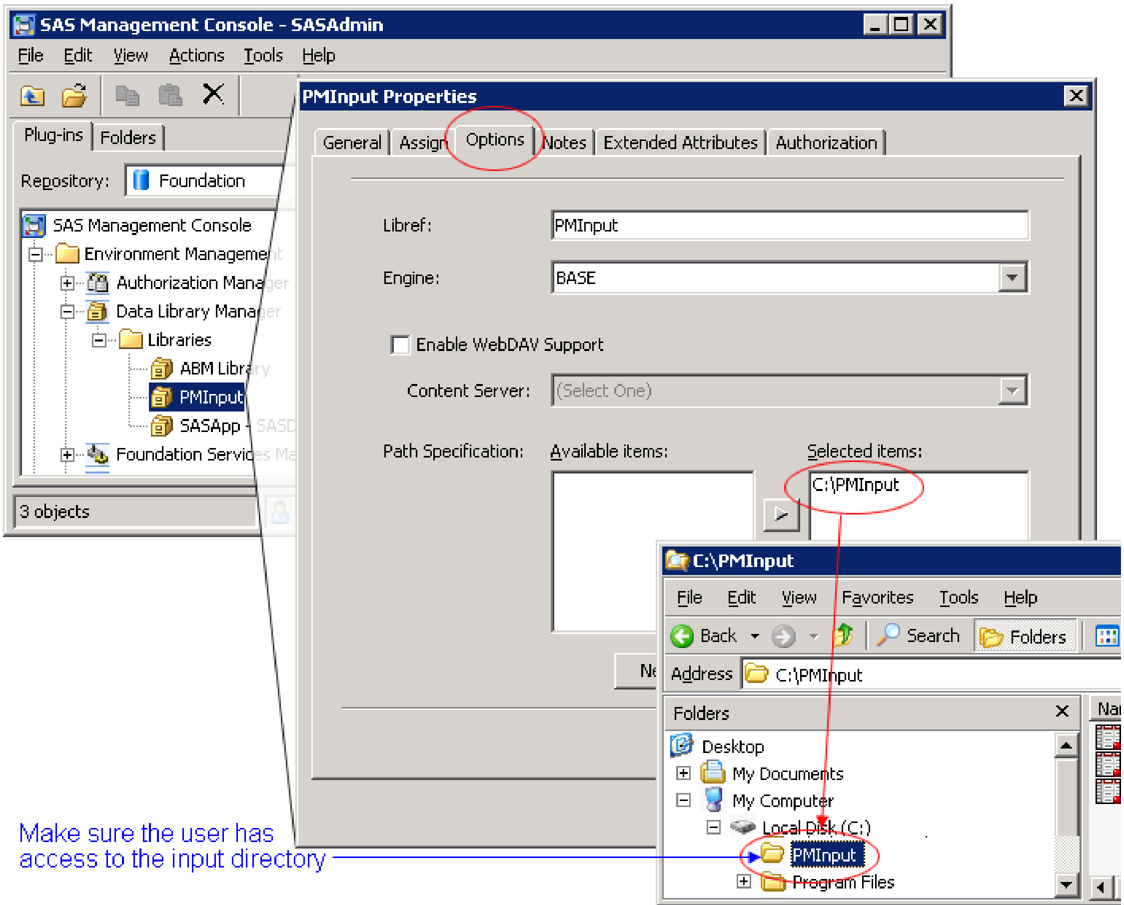
   The Metadata Server Options dialog box opens.

2. Specify a Repository, Application Server, and Logical Workspace Server.
3. Select an input Library from the drop-down list of libraries.

Note: The SAS Activity-Based Management user who publishes behaviors must have write access to the SAS Profitability Management input library to which the behaviors are written. The drop-down list displays for selection only those libraries to which the user has access.



Note: The SAS Activity-Based Management user who publishes behaviors must also have write access to the library's directory on the machine where the behaviors are published. The following picture shows a SAS Profitabiliity Management input library named PMInput whose machine directory is C:\PMInput. In this example, the SAS Activity-Based Management user must have write access to the C:\PMInput directory.



4. Click **OK**.

Use Account Data with SAS Profitability Management

You can use the data in SAS Activity-Based Management accounts with SAS Profitability Management. The process involves two steps:

1. Mark accounts as behaviors
2. Publish the behaviors to SAS Profitability Management

Publishing behaviors means that they are written to a behavior table for use with SAS Profitability Management. A behavior table contains source items with a transaction cost. A behavior table has the following required columns:

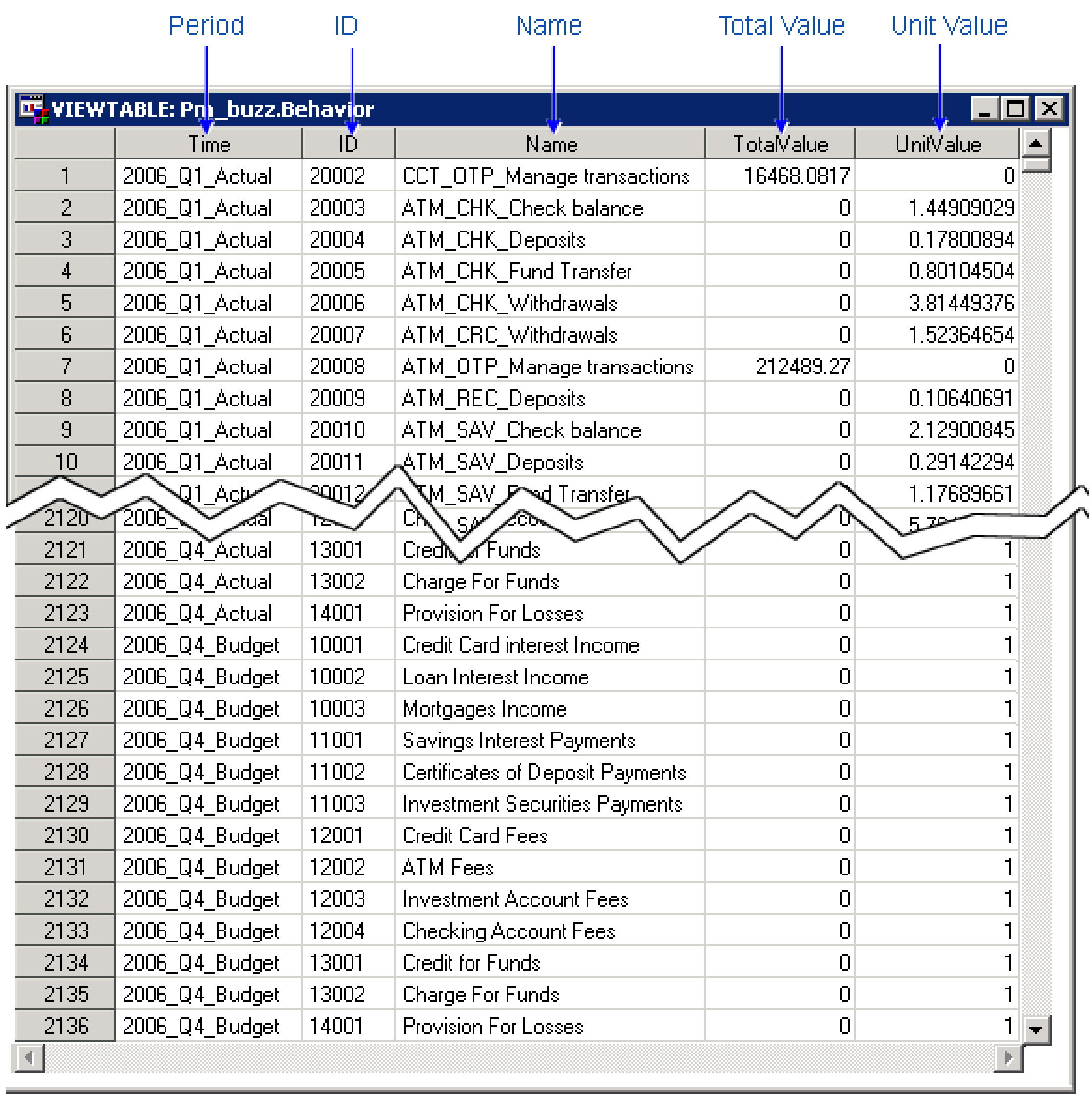
Position	Name	Maximum Length	Description
1	Time	Char 32	Defines the period for the costs
2	ID	Char 32	The identifying reference for the behavior
3	Name	Char 32	The name of the behavior

Position	Name	Maximum Length	Description
4	Total Value	Numeric 8	The total source amount that will be spread
5	Unit Value	Numeric 8	The unit cost for each transaction with this source

- The columns must appear in the order shown.
- Each column must have the length shown.
- The name of the column is arbitrary.

Note: Only one of the Unit-Value and Total-Value fields may contain a non-zero value for any row of the table.

The following picture shows a sample behavior table.



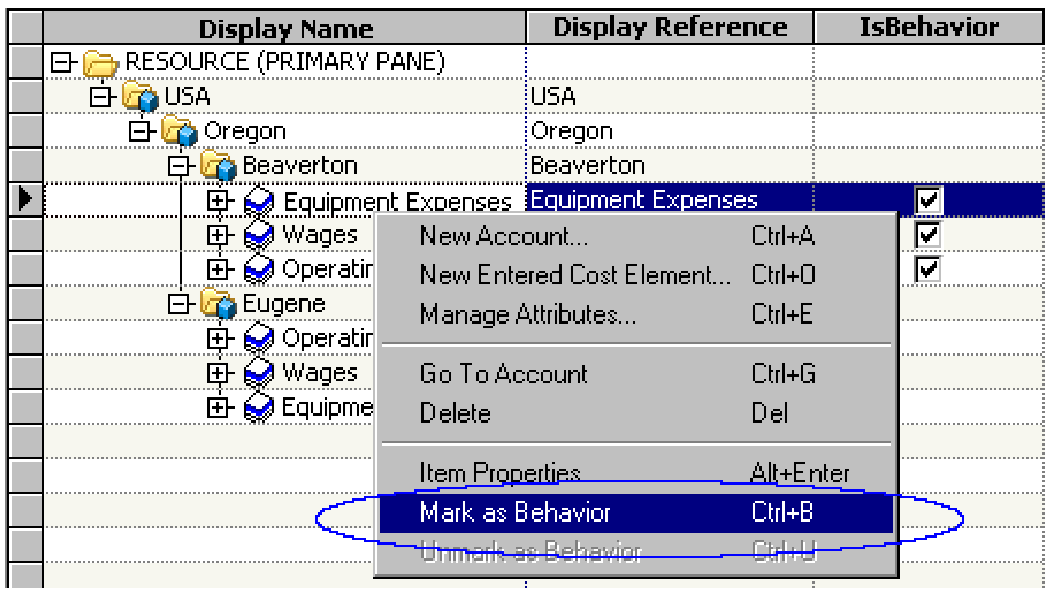
	Time	ID	Name	TotalValue	UnitValue
1	2006_Q1_Actual	20002	CCT_OTP_Manage transactions	16468.0817	0
2	2006_Q1_Actual	20003	ATM_CHK_Check balance	0	1.44909029
3	2006_Q1_Actual	20004	ATM_CHK_Deposits	0	0.17800894
4	2006_Q1_Actual	20005	ATM_CHK_Fund Transfer	0	0.80104504
5	2006_Q1_Actual	20006	ATM_CHK_Withdrawals	0	3.81449376
6	2006_Q1_Actual	20007	ATM_CRC_Withdrawals	0	1.52364654
7	2006_Q1_Actual	20008	ATM_OTP_Manage transactions	212489.27	0
8	2006_Q1_Actual	20009	ATM_REC_Deposits	0	0.10640691
9	2006_Q1_Actual	20010	ATM_SAV_Check balance	0	2.12900845
10	2006_Q1_Actual	20011	ATM_SAV_Deposits	0	0.29142294
[illegible]	[illegible]Q1_Act[illegible]	20012	[illegible]M_SAV_F[illegible]d Transfer	[illegible]	1.17689661
2120	2006[illegible]	12[illegible]	Ch[illegible] SA[illegible]	0	5.7[illegible]
2121	2006_Q4_Actual	13001	Cred[illegible] Funds	0	1
2122	2006_Q4_Actual	13002	Charge For Funds	0	1
2123	2006_Q4_Actual	14001	Provision For Losses	0	1
2124	2006_Q4_Budget	10001	Credit Card interest Income	0	1
2125	2006_Q4_Budget	10002	Loan Interest Income	0	1
2126	2006_Q4_Budget	10003	Mortgages Income	0	1
2127	2006_Q4_Budget	11001	Savings Interest Payments	0	1
2128	2006_Q4_Budget	11002	Certificates of Deposit Payments	0	1
2129	2006_Q4_Budget	11003	Investment Securities Payments	0	1
2130	2006_Q4_Budget	12001	Credit Card Fees	0	1
2131	2006_Q4_Budget	12002	ATM Fees	0	1
2132	2006_Q4_Budget	12003	Investment Account Fees	0	1
2133	2006_Q4_Budget	12004	Checking Account Fees	0	1
2134	2006_Q4_Budget	13001	Credit for Funds	0	1
2135	2006_Q4_Budget	13002	Charge For Funds	0	1
2136	2006_Q4_Budget	14001	Provision For Losses	0	1

Mark Accounts as Behaviors

Mark accounts as behaviors before publishing them to SAS Profitability Management. You can mark an individual account or multiple accounts.

Mark an Individual Account

1. Select the account.
2. Right-click the account.
3. Select Mark as Behavior.



Mark Multiple Accounts

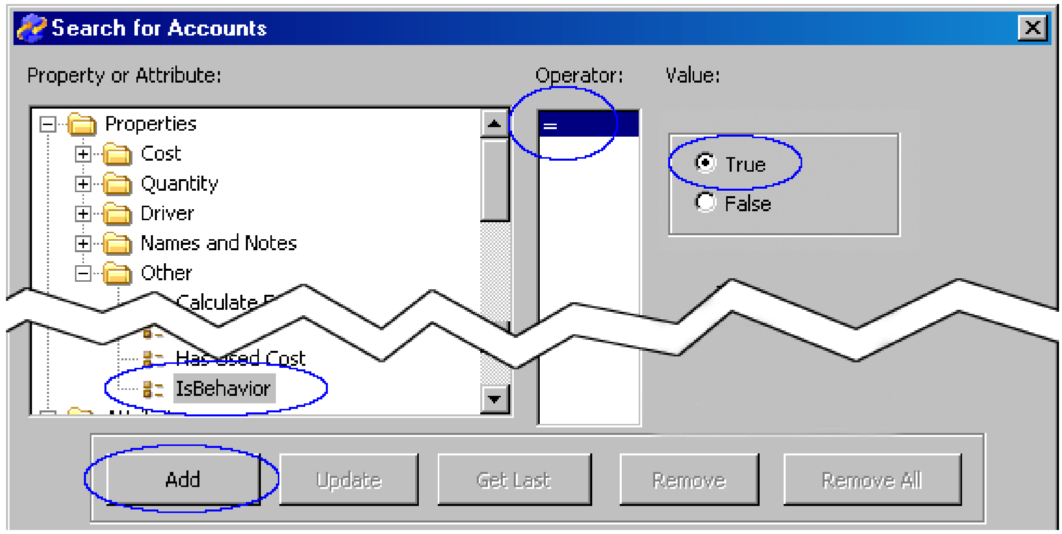
1. Open the Search for Accounts dialog box.
2. Search for the accounts that you want to mark as behaviors.
3. From the list of accounts that are found, select the ones to be marked.
4. Click Actions, and then select Mark as Behaviors.

Search for Accounts Marked as Behaviors

1. Open the Search for Accounts dialog box.
2. Select the **IsBehavior** property (under **Properties** ⇨ **Other**).
3. Select the = operator.
4. Select **True**.
5. Click **Add**.

6. Click **Search**.

   Accounts that have been marked as behaviors are listed.

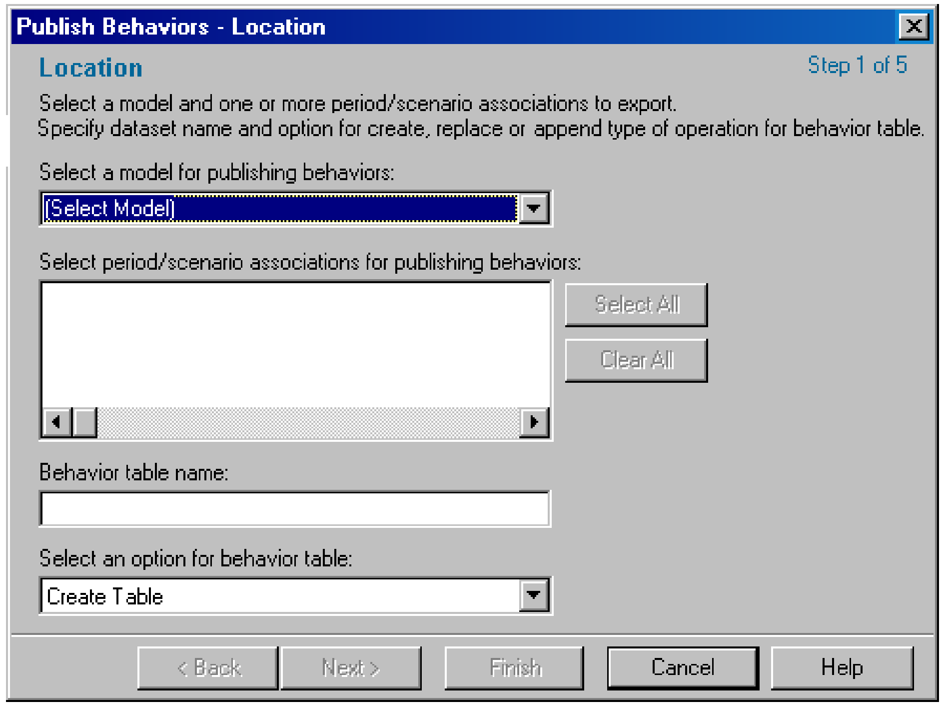


Publish Behaviors to SAS Profitability Management

Select **File** ⇨ **Publish** ⇨ **Behaviors**. The Publish Behaviors wizard opens.

Note: Before you can publish behaviors, you must specify the SAS Profitability Management input library so that SAS Activity-Based Management knows where to store the published behaviors. And, you must have marked some accounts as behaviors.

1. Select a model and specify the name of the behavior table to be created.



Model name
: Select the model whose accounts you want to mark as behaviors.

Period/Scenario associations
: Select the period/scenario associations for which you want to publish the data.

Behavior table name
: Specify a name for the behavior table.

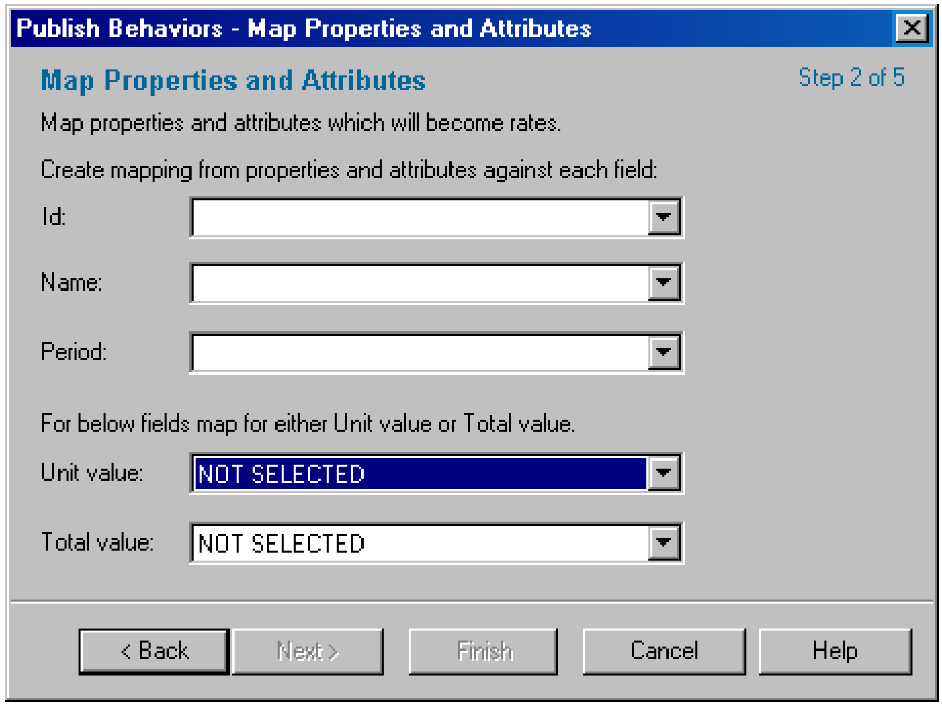
Option

Create table
: Creates a behavior table. If a table with the same name exists, then the operation quits with an error message, and the existing table is undisturbed.

Replace table
: Replaces an existing table with the same name.

Append to table
: Appends records to an existing table.

2. Map properties and attributes of the accounts being published to fields in the resulting behavior table being created.



Id
: is the identifying reference for the behavior

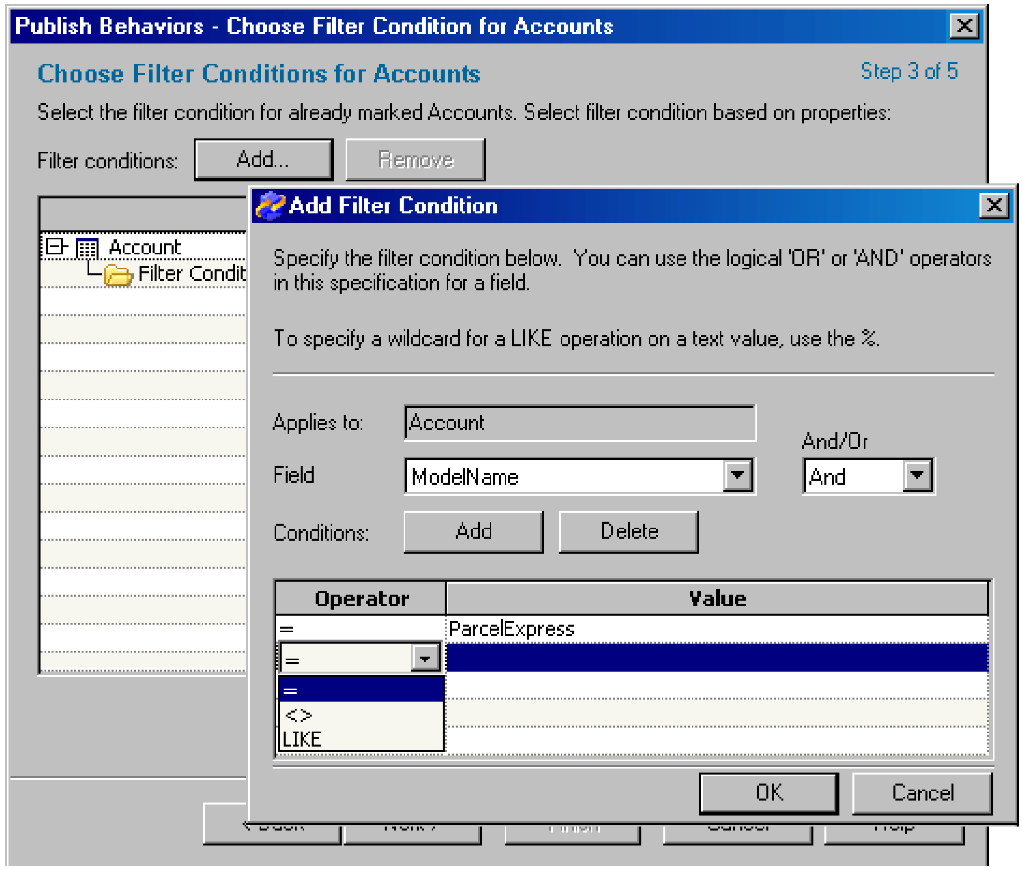
Name
: is the name of the behavior

Period
: defines the period for the costs

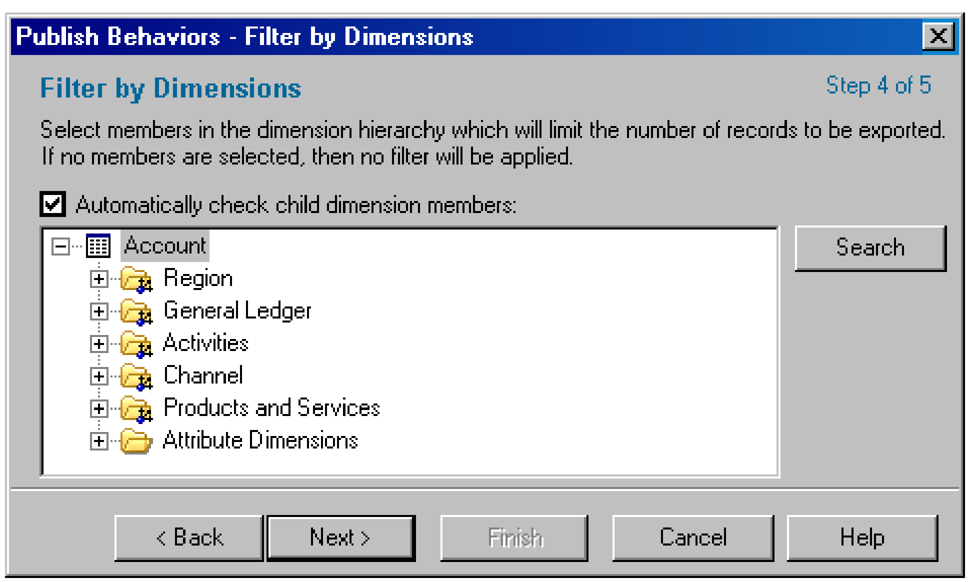
Unit value
: is the unit cost for each transaction with this source. If you select a Unit value, then you may not select a Total value.

Total value
: is the total source amount that will be spread. If you select a Total value, then you may not select a Unit value.

3. You can optionally set a condition that an account must satisfy to be published as a behavior. This allows you to select a subset of all the accounts that have been marked as behaviors.



4. You can further limit the number of accounts to be published by selecting dimensions. If you do not select any dimensions, then accounts (which are marked as behaviors) from all dimensions are published.



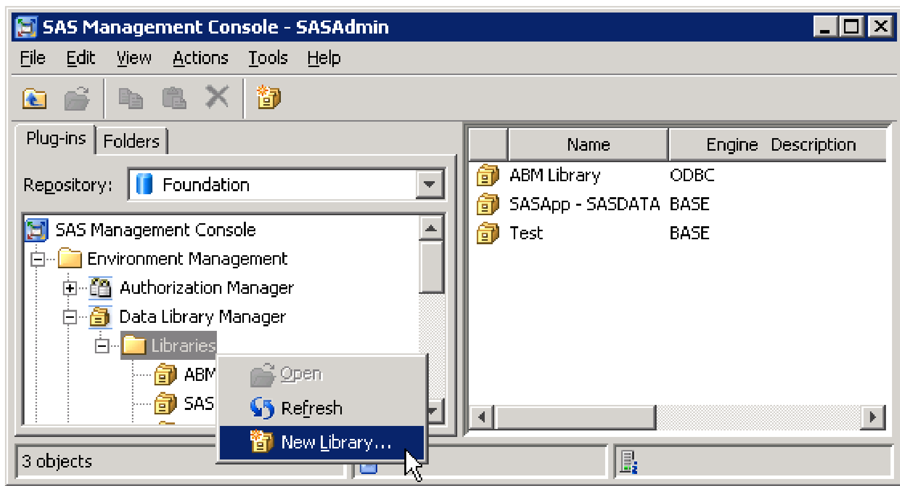
5. Review your selections, and then click **Finish**.

   The published accounts are written to a behavior table in the Profitability Management input library.

Create a SAS Profitability Management Input Library

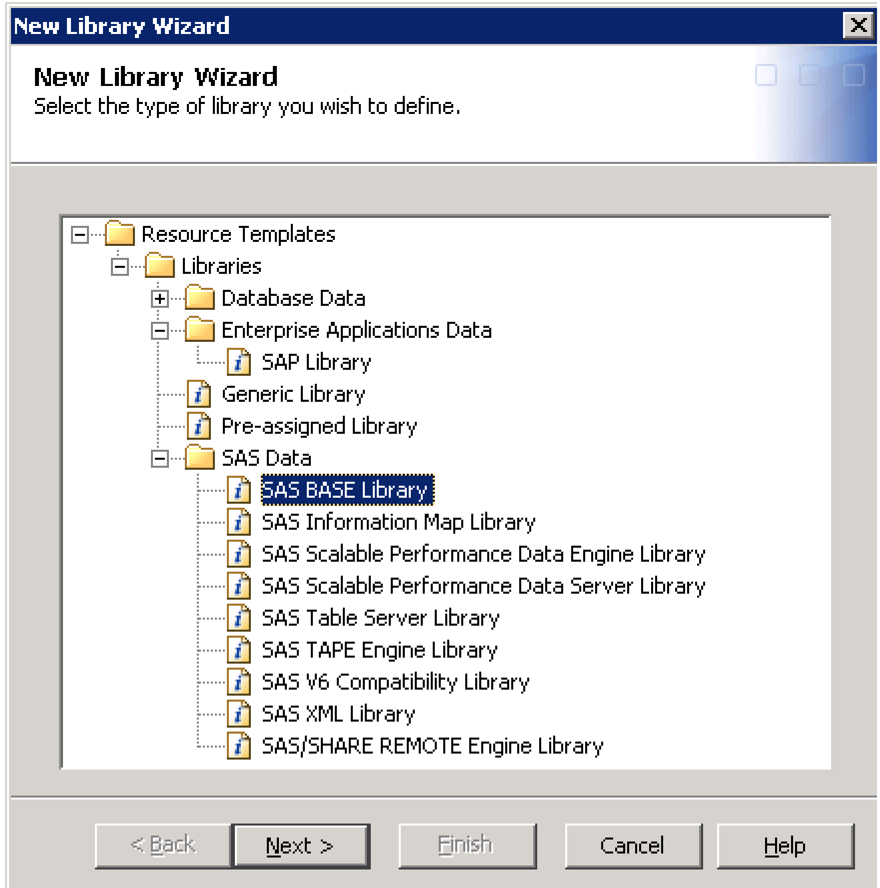
If a SAS Profitability Management input library does not already exist, follow these instructions to create one.

1. Open SAS Management Console.
2. Select the **Plug-ins** tab.
3. Right-click **Libraries** (under Data Library Manager) and select **New Library**.



4. Select the type of library, and then click **Next**.

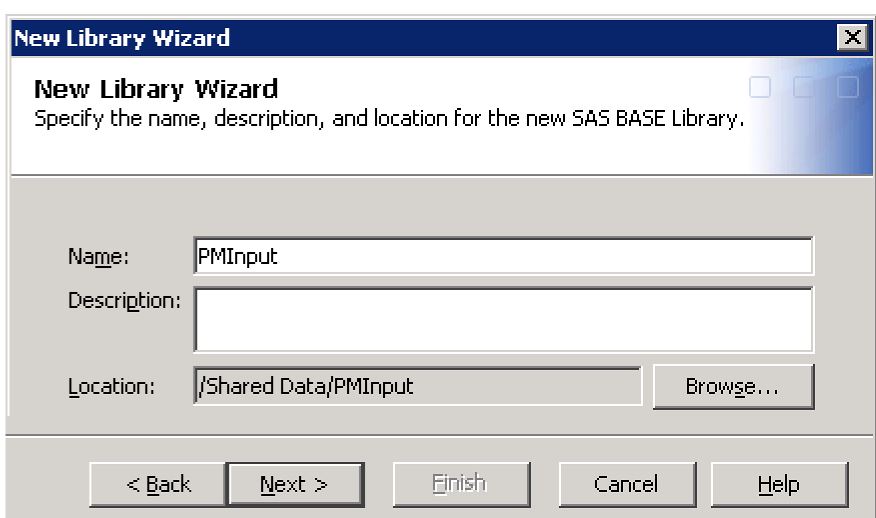
   For input libraries, SAS Profitability Management supports any library type for which you can create a libref. The following picture shows selecting a library type of SAS Base Library.



5. Specify the library Name and the Location of its metadata folder, and then click **Next**.

Name

This is the text that appears in the navigation and display areas of SAS Management Console, but is not the LIBREF for the library. The name can contain up to 60 characters. It must be unique within the folder specified in the Location field. It must also be unique on all servers where the library is assigned.

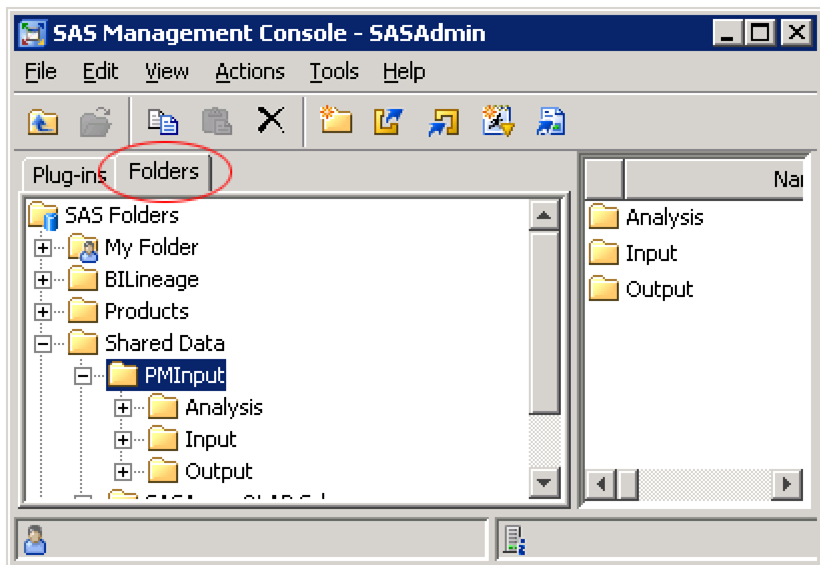


Folder

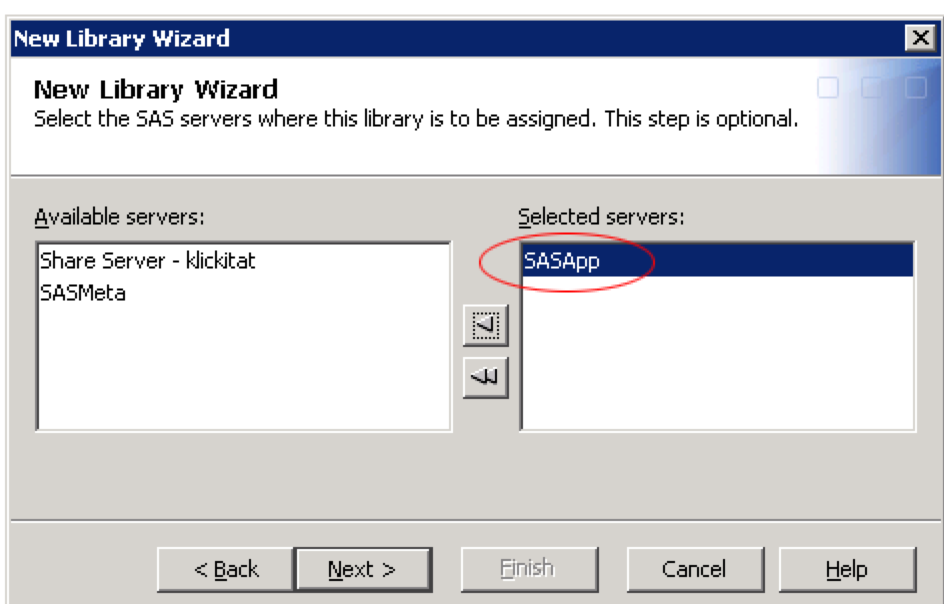
It is strongly recommended that you choose a unique folder for each library. If two libraries share the same metadata folder and both libraries contain same-named items, then the metadata folder will contain incorrect information about one of those items.

Note: In order to specify the Location of the library's metadata folder, you should have already created the folder. To create the library's metadata folder, do the following:

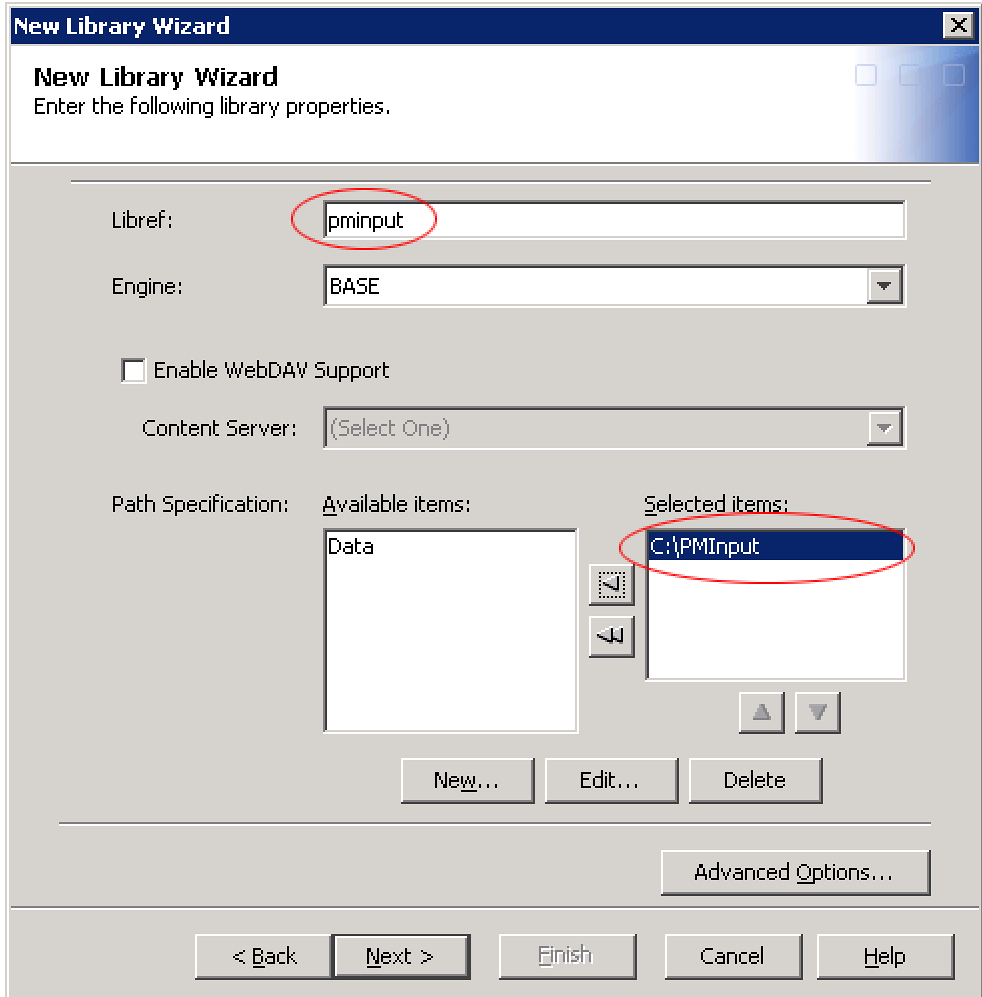
1. Click the Folders tab.
2. Right-click and select New Folder.
3. Name the folder, and then click Finish.



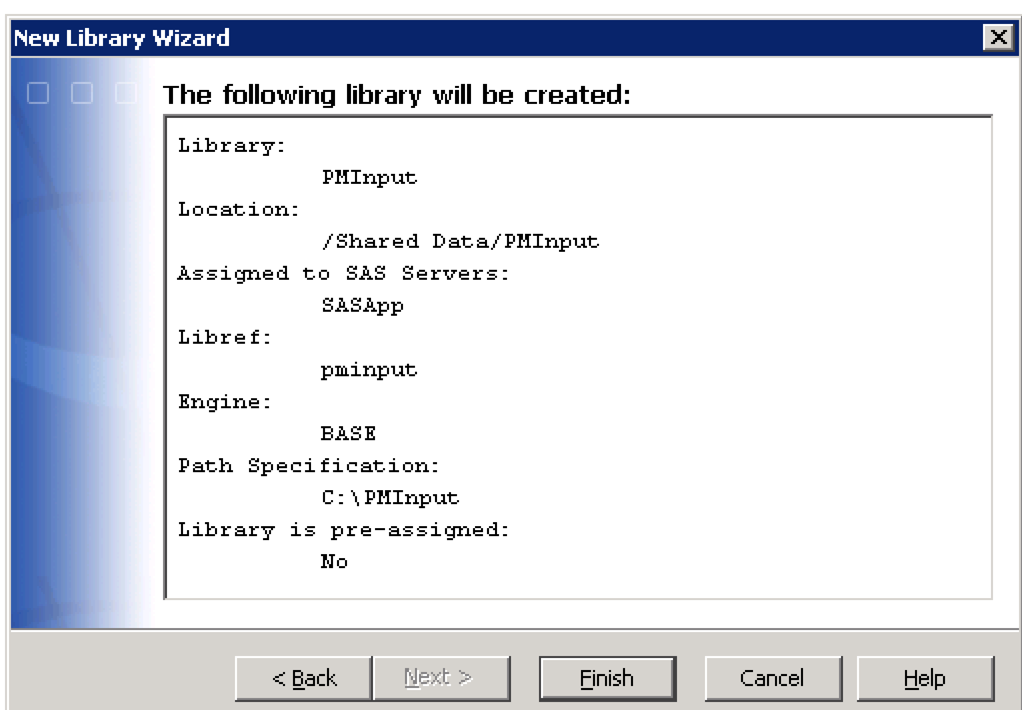
6. Select **SASApp** as the SAS server, and then click **Next**.



7. Specify a Libref for the library and the Path of the machine directory where the library's data is stored, and then click **Next**.

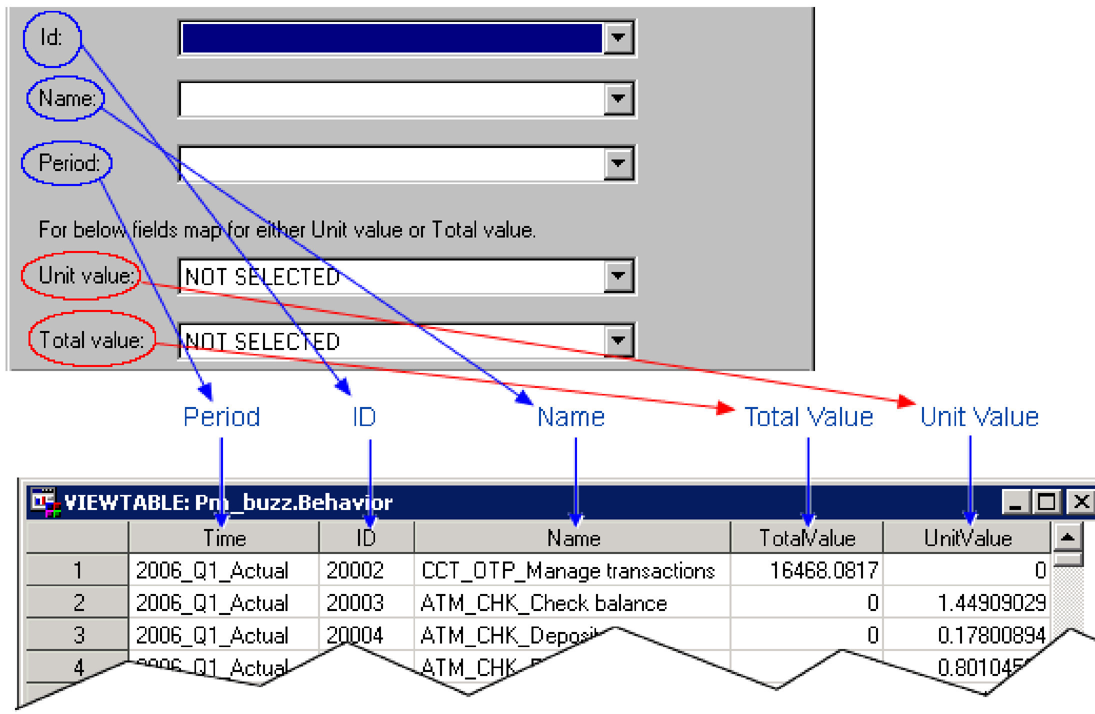


8. Review your choices, and then click **Finish**.



Map to the Behavior Table

The following picture shows a sample mapping to a behavior table. Both the names of the fields in the pictured table and their order in the table are arbitrary. You identify the columns to SAS Activity-Based Management by the process of mapping.



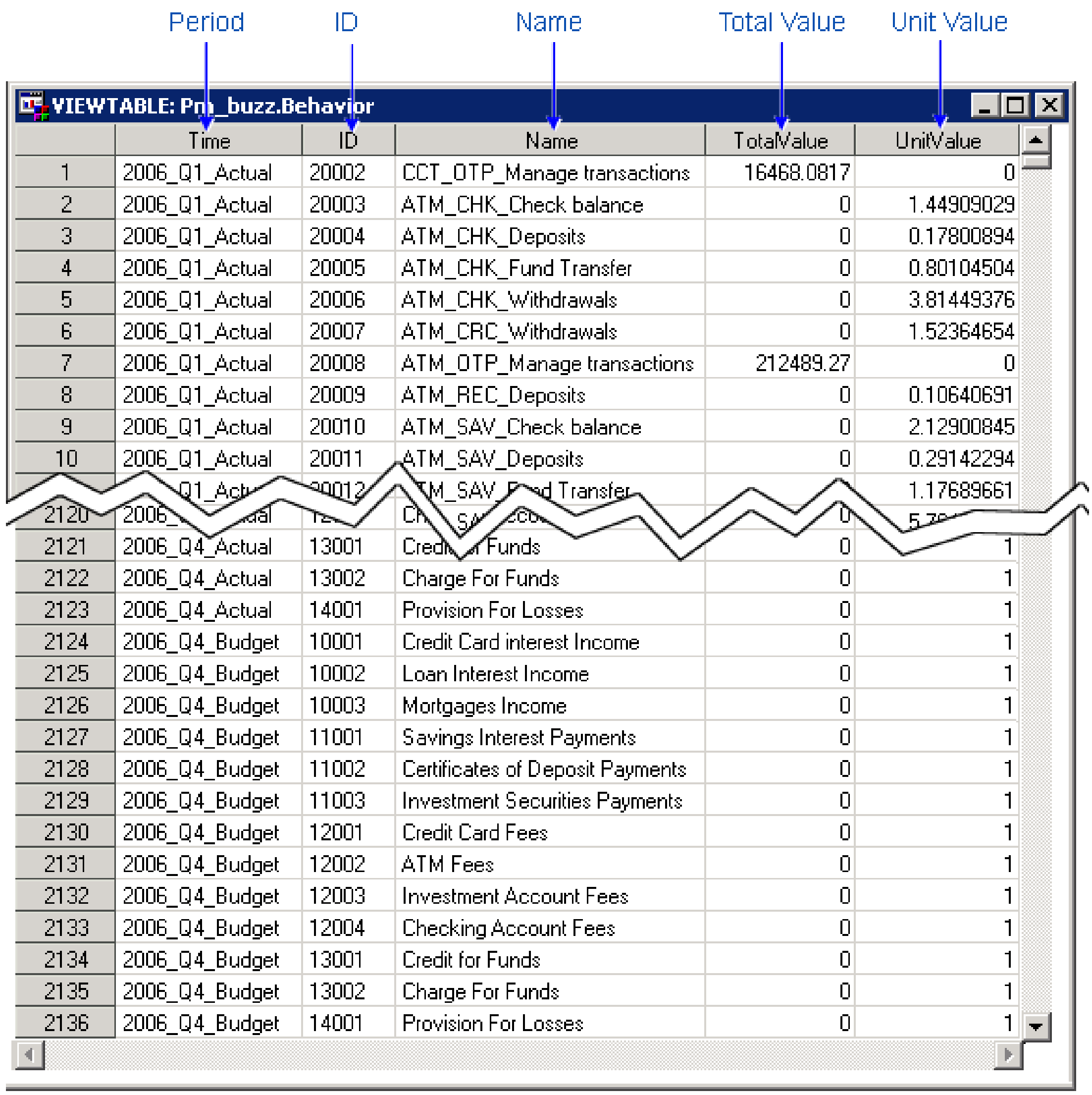
A SAS Profitability Management behavior table contains source items with their transaction costs. A behavior table has the following required columns:

Position	Name	Maximum Length	Description
1	Time	Char 32	Defines the period for the costs
2	ID	Char 32	The identifying reference for the behavior
3	Name	Char 32	The name of the behavior
4	Total Value	Numeric 8	The total source amount that will be spread
5	Unit Value	Numeric 8	The unit cost for each transaction with this source

- Each column must have the specified length.
- The name of each column is arbitrary.
- The order of the columns is arbitrary.

Note: Only one of the Unit-Value and Total-Value fields may contain a non-zero value for any row of the table.

The following picture shows a sample behavior table.



	Time	ID	Name	TotalValue	UnitValue
1	2006_Q1_Actual	20002	CCT_OTP_Manage transactions	16468.0817	0
2	2006_Q1_Actual	20003	ATM_CHK_Check balance	0	1.44909029
3	2006_Q1_Actual	20004	ATM_CHK_Deposits	0	0.17800894
4	2006_Q1_Actual	20005	ATM_CHK_Fund Transfer	0	0.80104504
5	2006_Q1_Actual	20006	ATM_CHK_Withdrawals	0	3.81449376
6	2006_Q1_Actual	20007	ATM_CRC_Withdrawals	0	1.52364654
7	2006_Q1_Actual	20008	ATM_OTP_Manage transactions	212489.27	0
8	2006_Q1_Actual	20009	ATM_REC_Deposits	0	0.10640691
9	2006_Q1_Actual	20010	ATM_SAV_Check balance	0	2.12900845
10	2006_Q1_Actual	20011	ATM_SAV_Deposits	0	0.29142294
[illegible]	[illegible]Q1_Act[illegible]	20012	[illegible]M_SAV_F[illegible]d Transfer	[illegible]	1.17689661
2120	2006_[illegible]al	[illegible]	Ch[illegible] SA[illegible]ccou[illegible]	0	[illegible]
2121	2006_Q4_Actual	13001	Cred[illegible]r Funds	0	1
2122	2006_Q4_Actual	13002	Charge For Funds	0	1
2123	2006_Q4_Actual	14001	Provision For Losses	0	1
2124	2006_Q4_Budget	10001	Credit Card interest Income	0	1
2125	2006_Q4_Budget	10002	Loan Interest Income	0	1
2126	2006_Q4_Budget	10003	Mortgages Income	0	1
2127	2006_Q4_Budget	11001	Savings Interest Payments	0	1
2128	2006_Q4_Budget	11002	Certificates of Deposit Payments	0	1
2129	2006_Q4_Budget	11003	Investment Securities Payments	0	1
2130	2006_Q4_Budget	12001	Credit Card Fees	0	1
2131	2006_Q4_Budget	12002	ATM Fees	0	1
2132	2006_Q4_Budget	12003	Investment Account Fees	0	1
2133	2006_Q4_Budget	12004	Checking Account Fees	0	1
2134	2006_Q4_Budget	13001	Credit for Funds	0	1
2135	2006_Q4_Budget	13002	Charge For Funds	0	1
2136	2006_Q4_Budget	14001	Provision For Losses	0	1

Chapter 27
SAS Strategy Management

Steps for Integrating with SAS Strategy Management

What Is SAS Strategy Management

SAS Activity-Based Management provides reliable, fact-based data that can be used to evaluate an organization's past performance and affect future decisions. Using SAS Activity-Based Management, you can mark accounts and rollup accounts from the Resource, Activity, Cost Object, and External Units modules as items for performance review. Any property or numeric attribute of a SAS Activity-Based Management account can be marked and published as a performance measure to be used in other reporting tools, such as SAS Strategy Management.

A *performance measure*, also known as a *key performance indicator* (*KPI*), can be any number that you consider significant in the context of your business and, therefore, want to track. A *KPI element*, or *metric*, is a specific, predefined measure of the success of an organization. For example, KPI elements that measure your organization's financial performance might be average annual profit per customer and average cost of sale or service. You can include this SAS Activity-Based Management information in other reporting applications, such as by integrating them with SAS Strategy Management.

TIP Anything that you define as a performance measure in SAS Activity-Based Management and display as a KPI element in SAS Strategy Management can also be displayed in the SAS Information Delivery Portal (dashboard, aggregation view, and table view portlets).

Integrating SAS Activity-Based Management with SAS Strategy Management provides quick insight to issues and a path toward their resolution. Both solutions are complimentary to almost any process or activity:

- Many measures that are used in SAS Strategy Management are financial in perspective and include elements of profitability and/or cost.
- Many measures that are used in SAS Strategy Management are metrics that are used as cost drivers in SAS Activity-Based Management.
- SAS Strategy Management identifies leading and lagging indicators of performance, and SAS Activity-Based Management provides insight into the cause of the values of these indicators.
- SAS Activity-Based Management provides the actionable detail behind the KPIs.

Calculating the costs of a SAS Activity-Based Management model provides a picture of an organization's behavior. The model values (numeric properties and attributes) that you specify as performance measures are calculated by SAS Activity-Based Management, and they can then be provided or published to a SAS Strategy Management scorecard. A collection of performance measures, or KPI elements, comprises a SAS Strategy Management project. (Scorecards can be arranged in a hierarchical tree by using the SAS Solutions Dimension Editor.)

Integration Procedure

Integrating with SAS Strategy Management is a four-step process:

1. "Select Performance Measures" on page 444
2. "Choose the Table Format for Publishing" on page 447
3. "Publish the Measures" on page 451
4. "Import Performance Measures into SAS Strategy Management" on page 451

Select Performance Measures

Add Performance Measures

The following items can be used as performance measures:

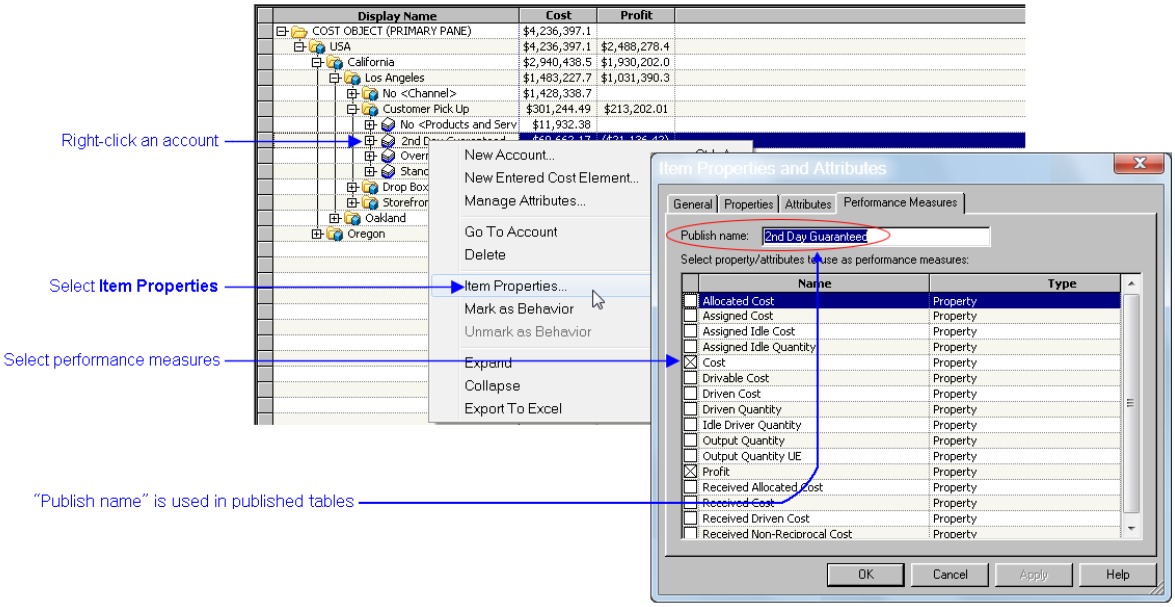
- numeric properties (costs and quantities)
- numeric attributes

To add a performance measure to an account:

1. In one of the modules, right-click an account.
2. Select **Item Properties**.
3. Select the performance measures to be applied to the account and click **OK**.

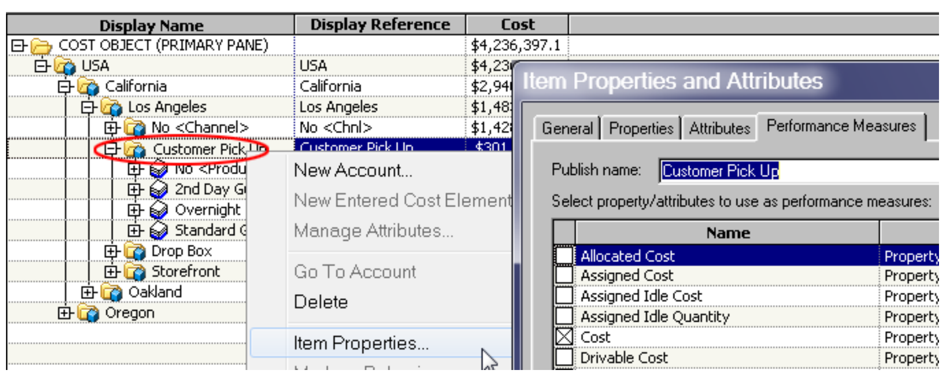
   Note: Performance measures are periodic. When you associate a performance measure with an account, it is associated only with the period/scenario association in which you attach it.

   Note: The Publish name of an account is the name that is used for the account in tables that are written to the database when you publish performance measures.

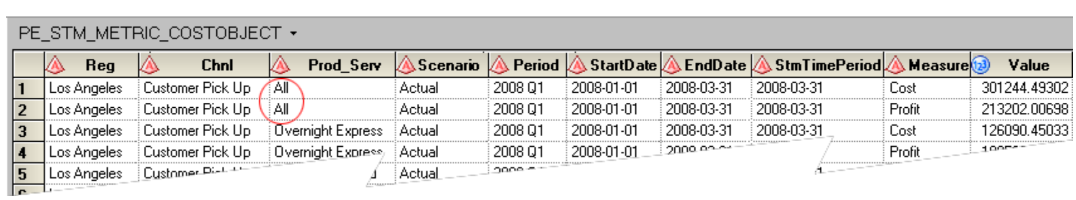


Selecting Roll–Up Accounts

To create a record in the metric table for ALL, select a roll-up account and mark it as a performance measure, as shown for example in the following picture:

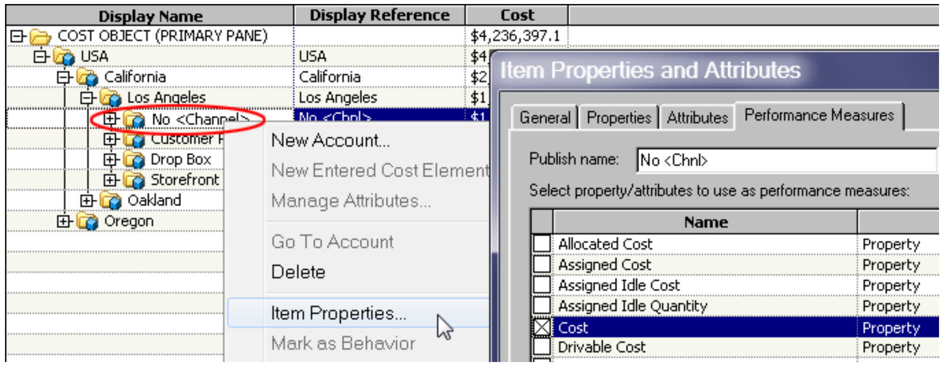


The following picture shows a portion of the resulting metric table:


PE_STM_METRIC_COSTOBJECT

	Reg	Chnl	Prod_Serv	Scenario	Period	StartDate	EndDate	StmTimePeriod	Measure	Value
1	Los Angeles	Customer Pick Up	All	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	301244.49302
2	Los Angeles	Customer Pick Up	All	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	213202.00698
3	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	126090.45033
4	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	[illegible]	[illegible]	Profit	[illegible]
5	Los Angeles	[illegible]	[illegible]	Actual	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Similarly, to create a record in the metric table for NONE, select the <No> roll-up account and mark it as a performance measure, as shown for example in the following picture:

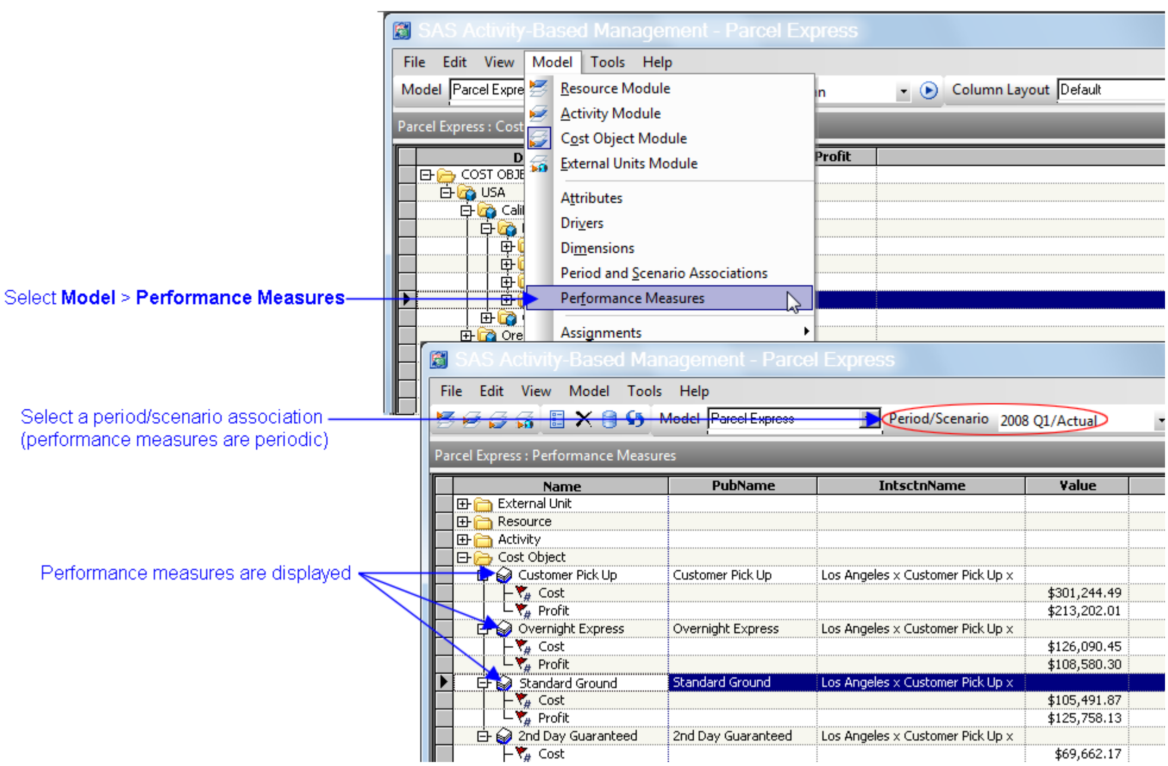


Display Performance Measures

To display the performance measures that have been added to accounts:

1. Select **Model** ⇨ **Performance Measures**.
2. Select a period/scenario association. Performance measures are periodic.

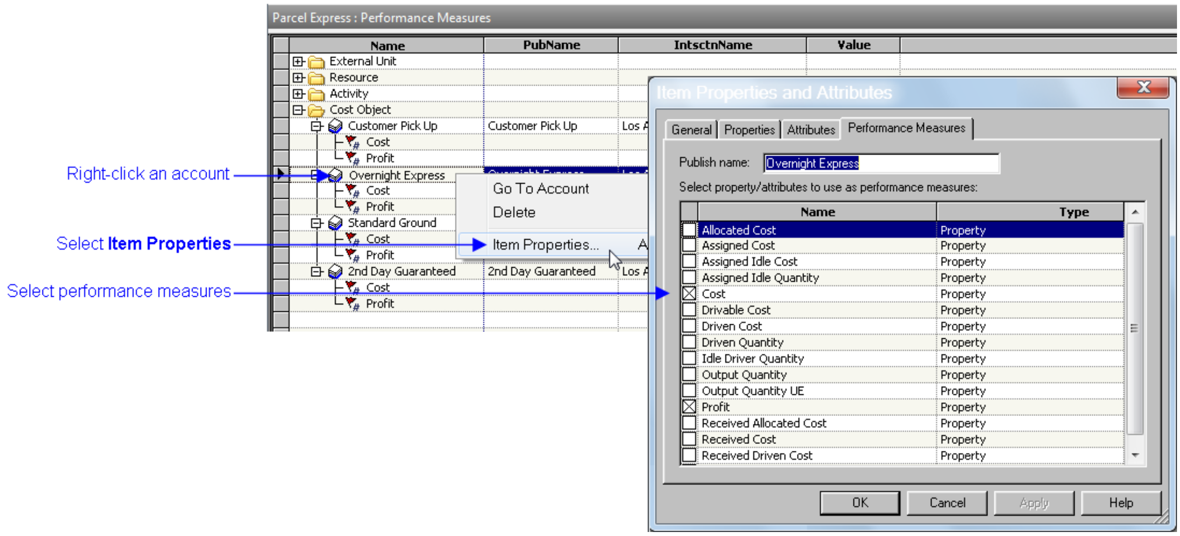
   The Performance Measures view opens to display all the performance measures that have been added to accounts.



Modify Performance Measures

You can add or remove performance measures from an account that appears in the Performance Measures view by right-clicking the account and selecting **Item Properties**.

However, if an account does not yet have a performance measure and, consequently, does not appear in the Performance Measures view, then the only way to add a performance measure is from a module view, as described previously.



Choose the Table Format for Publishing

Overview

Before publishing performance measures, choose options for the output tables. There are two options for publishing model data to SAS Strategy Management:

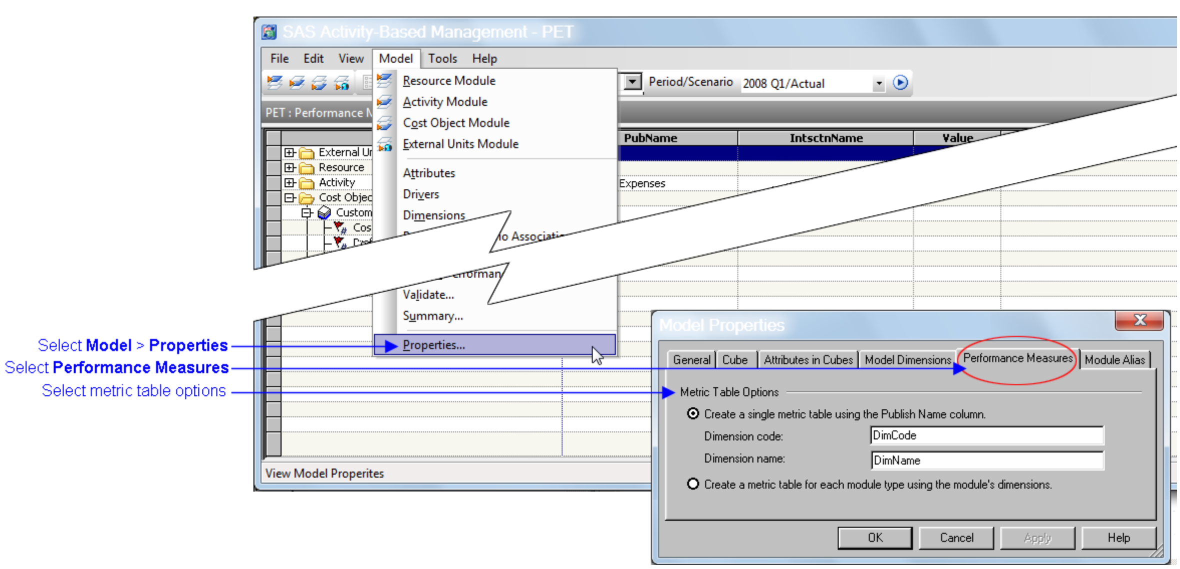
- “Create a single metric table using the Publish Name column” on page 448
- “Create a metric table for each module type using the module's dimensions ” on page 449

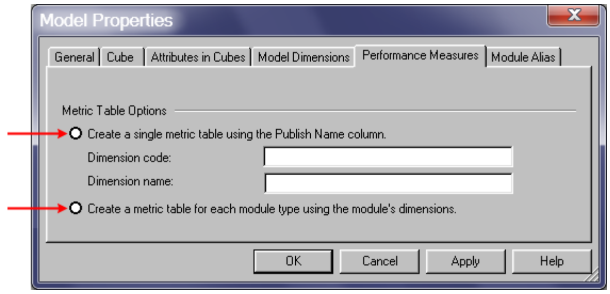
1. Select **Model** ⇨ **Properties**.

   The Model Properties window opens.

2. Select the **Performance Measures** tab.

3. Select options for the output tables.



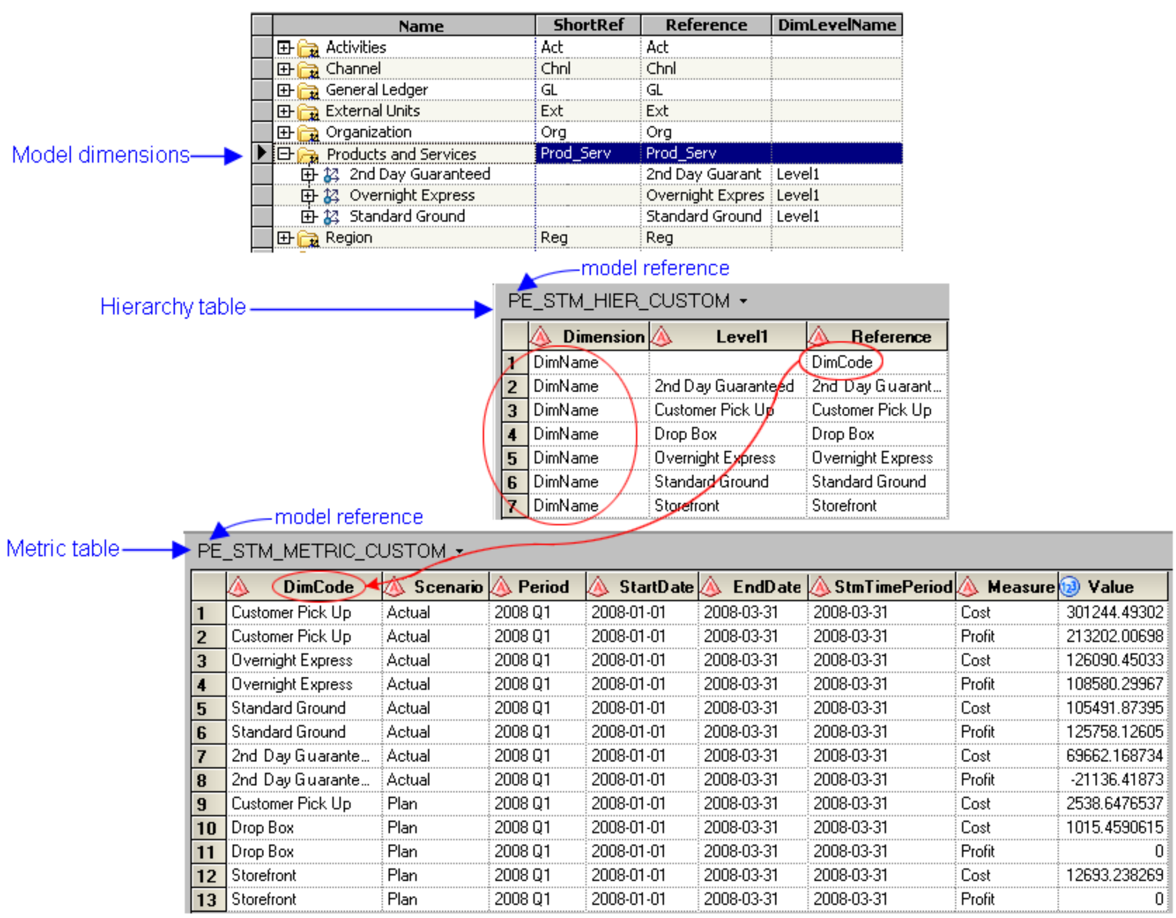


Create a single metric table using the Publish Name column

Selecting this option creates one hierarchy table and one metric table per SAS Activity-Based Management model. The metric table contains a single dimension column with dimension members, each of whose name is the "Publish name" of the account (see "Select Performance Measures" on page 444). Use this option to summarize SAS Activity-Based Management metrics into a single dimension in a SAS Strategy Management scorecard.

Entering the Dimension code and the Dimension name that you want to publish to SAS Strategy Management uniquely identifies the dimension for this SAS Activity-Based Management model.

The following picture shows a sample hierarchy table and metric table that result from choosing this option:



Name	ShortRef	Reference	DimLevelName
Activities	Act	Act	
Channel	Chnl	Chnl	
General Ledger	GL	GL	
External Units	Ext	Ext	
Organization	Org	Org	
Products and Services	Prod_Serv	Prod_Serv	
2nd Day Guaranteed		2nd Day Guarant	Level1
Overnight Express		Overnight Expres	Level1
Standard Ground		Standard Ground	Level1
Region	Reg	Reg	

	Dimension	Level1	Reference
1	DimName		DimCode
2	DimName	2nd Day Guaranteed	2nd Day Guarant...
3	DimName	Customer Pick Up	Customer Pick Up
4	DimName	Drop Box	Drop Box
5	DimName	Overnight Express	Overnight Express
6	DimName	Standard Ground	Standard Ground
7	DimName	Storefront	Storefront

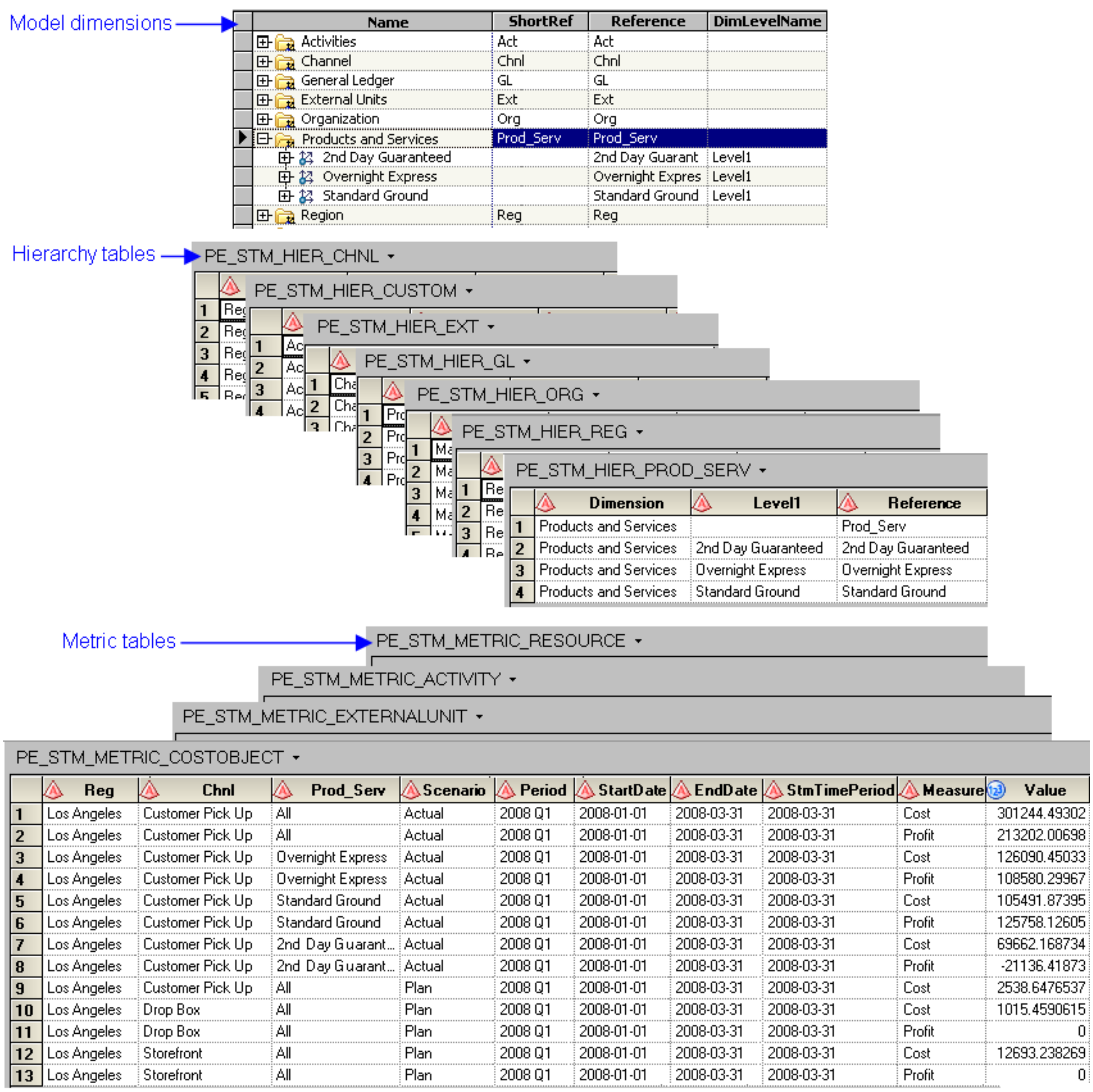
	DimCode	Scenario	Period	StartDate	EndDate	StmTimePeriod	Measure	Value
1	Customer Pick Up	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	301244.49302
2	Customer Pick Up	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	213202.00698
3	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	126090.45033
4	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	108580.29967
5	Standard Ground	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	105491.87395
6	Standard Ground	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	125758.12605
7	2nd Day Guarante...	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	69662.168734
8	2nd Day Guarante...	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	-21136.41873
9	Customer Pick Up	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	2538.6476537
10	Drop Box	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	1015.4590615
11	Drop Box	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	0
12	Storefront	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	12693.238269
13	Storefront	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	0

Note: Dimension names cannot contain blanks when used with SAS Strategy Management because of an ODBC restriction.

Create a metric table for each module type using the module's dimensions

Selecting this option creates a different metric table for each module in the SAS Activity-Based Management model. The metric table contains a dimension column corresponding to each dimension in the module, with dimension members that are created from the SAS Activity-Based Management dimension members for that module. Use this option if you want to maintain the dimension detail of the SAS Activity-Based Management metrics in the SAS Strategy Management scorecard.

The following picture shows sample hierarchy tables and metric tables that result from choosing this option:



Name	ShortRef	Reference	DimLevelName
Activities	Act	Act	
Channel	Chnl	Chnl	
General Ledger	GL	GL	
External Units	Ext	Ext	
Organization	Org	Org	
Products and Services	Prod_Serv	Prod_Serv	
2nd Day Guaranteed		2nd Day Guarant	Level1
Overnight Express		Overnight Expres	Level1
Standard Ground		Standard Ground	Level1
Region	Reg	Reg	

PE_STM_HIER_PROD_SERV

	Dimension	Level1	Reference
1	Products and Services		Prod_Serv
2	Products and Services	2nd Day Guaranteed	2nd Day Guaranteed
3	Products and Services	Overnight Express	Overnight Express
4	Products and Services	Standard Ground	Standard Ground

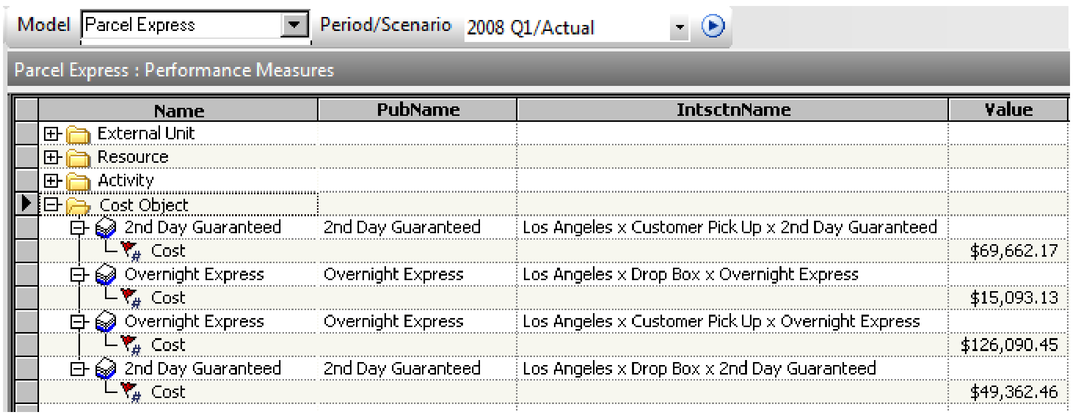
PE_STM_METRIC_COSTOBJECT

	Reg	Chnl	Prod_Serv	Scenario	Period	StartDate	EndDate	StmTimePeriod	Measure	Value
1	Los Angeles	Customer Pick Up	All	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	301244.49302
2	Los Angeles	Customer Pick Up	All	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	213202.00698
3	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	126090.45033
4	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	108580.29967
5	Los Angeles	Customer Pick Up	Standard Ground	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	105491.87395
6	Los Angeles	Customer Pick Up	Standard Ground	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	125758.12605
7	Los Angeles	Customer Pick Up	2nd Day Guarant...	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	69662.168734
8	Los Angeles	Customer Pick Up	2nd Day Guarant...	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	-21136.41873
9	Los Angeles	Customer Pick Up	All	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	2538.6476537
10	Los Angeles	Drop Box	All	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	1015.4590615
11	Los Angeles	Drop Box	All	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	0
12	Los Angeles	Storefront	All	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	12693.238269
13	Los Angeles	Storefront	All	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Profit	0

Collapsing the Metric Tables

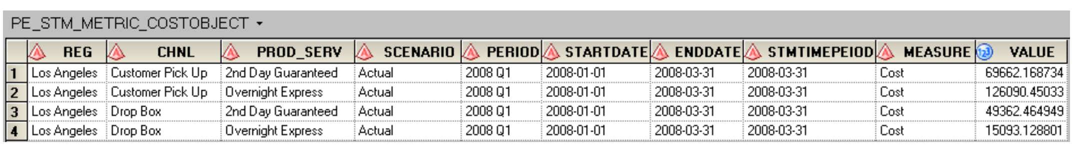
If you choose the option to Create-a-metric-table-for-each-module-type-using-the-module's-dimensions, then you must perform some post-processing on the metric tables published by SAS Activity-Based Management before they can be used by SAS Strategy Management. The reason is that SAS Strategy Management supports a single dimension in a table, and the metric tables published by SAS Activity-Based Management contain multiple dimensions. Consequently, you must perform post-processing to collapse any metric table that you want to import into SAS Strategy Management such that the metric table contains a single dimension.

To take a very simple example, suppose that you have selected four accounts in the Cost Object module as performance measures as shown in the following picture:



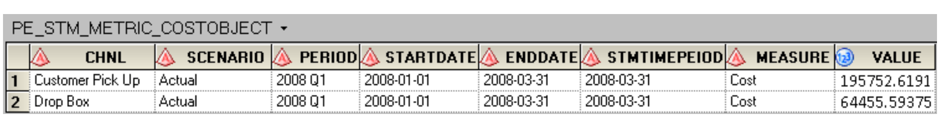
Name	PubName	IntsctnName	Value
External Unit			
Resource			
Activity			
Cost Object			
2nd Day Guaranteed	2nd Day Guaranteed	Los Angeles x Customer Pick Up x 2nd Day Guaranteed	
Cost			$69,662.17
Overnight Express	Overnight Express	Los Angeles x Drop Box x Overnight Express	
Cost			$15,093.13
Overnight Express	Overnight Express	Los Angeles x Customer Pick Up x Overnight Express	
Cost			$126,090.45
2nd Day Guaranteed	2nd Day Guaranteed	Los Angeles x Drop Box x 2nd Day Guaranteed	
Cost			$49,362.46

Publishing performance measures with the option to create-a-metric-table-for-each-module-type-using-the-module's-dimensions creates the metric table shown in the following picture:


PE_STM_METRIC_COSTOBJECT

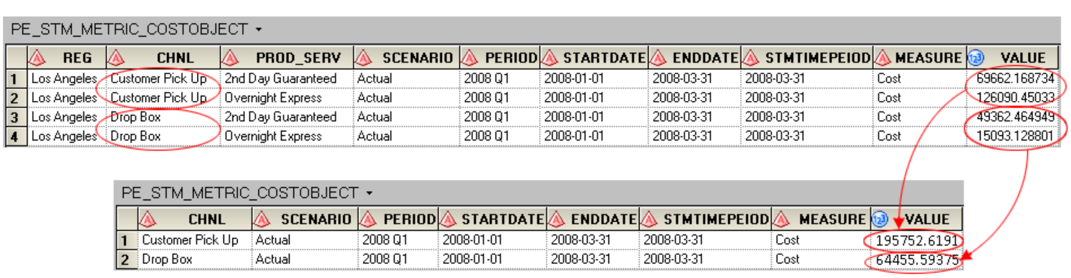
	REG	CHNL	PROD_SERV	SCENARIO	PERIOD	STARTDATE	ENDDATE	STMTIMEPEIOD	MEASURE	VALUE
1	Los Angeles	Customer Pick Up	2nd Day Guaranteed	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	69662.168734
2	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	126090.45033
3	Los Angeles	Drop Box	2nd Day Guaranteed	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	49362.464949
4	Los Angeles	Drop Box	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	15093.128801

After doing post-processing to collapse the table on the CHNL (Channel) dimension, the following table (containing a single dimension) is produced that can be used with SAS Strategy Management:


PE_STM_METRIC_COSTOBJECT

	CHNL	SCENARIO	PERIOD	STARTDATE	ENDDATE	STMTIMEPEIOD	MEASURE	VALUE
1	Customer Pick Up	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	195752.6191
2	Drop Box	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	64455.59375

You can notice in this example that the VALUE column contains the sum of the VALUE column of the rows that are collapsed, as shown in the following picture:


PE_STM_METRIC_COSTOBJECT

	REG	CHNL	PROD_SERV	SCENARIO	PERIOD	STARTDATE	ENDDATE	STMTIMEPEIOD	MEASURE	VALUE
1	Los Angeles	Customer Pick Up	2nd Day Guaranteed	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	69662.168734
2	Los Angeles	Customer Pick Up	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	126090.45033
3	Los Angeles	Drop Box	2nd Day Guaranteed	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	49362.464949
4	Los Angeles	Drop Box	Overnight Express	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	15093.128801

PE_STM_METRIC_COSTOBJECT

	CHNL	SCENARIO	PERIOD	STARTDATE	ENDDATE	STMTIMEPEIOD	MEASURE	VALUE
1	Customer Pick Up	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	195752.6191
2	Drop Box	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	64455.59375

Of course, in your post-processing you do not have to use the SUM function. You can choose to collapse the rows in any way that you like.

Publish the Measures

Procedure

To publish the performance measures:

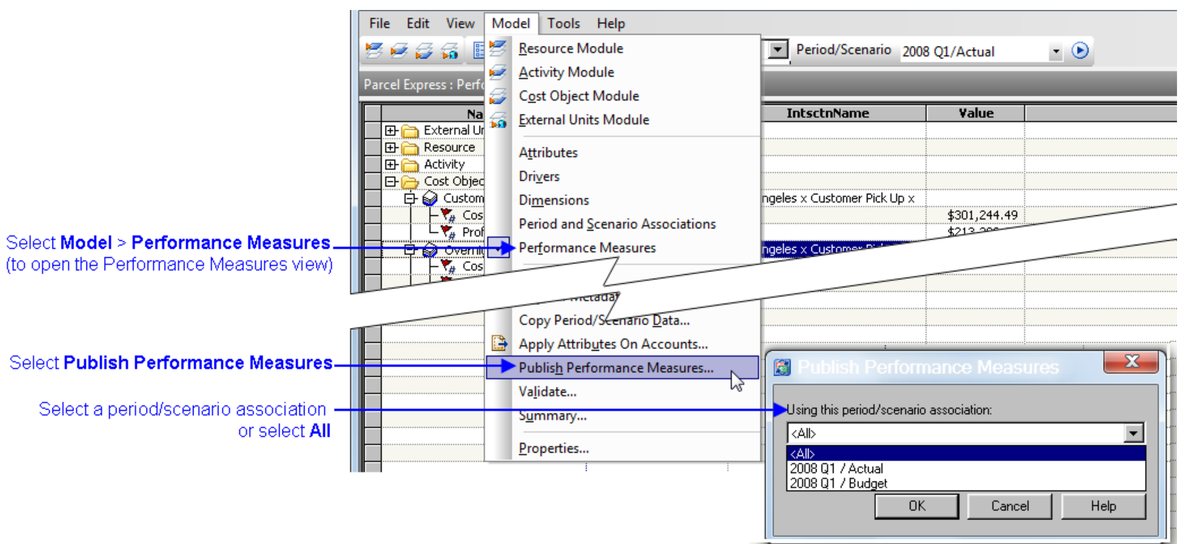
1. Select **Model** ⇨ **Performance Measures**.

   The Performance Measures view opens.

2. Select **Publish Performance Measures**.

   The Publish Performance Measures window opens for selecting a period/scenario to publish.

3. Select a period/scenario association to publish, or select **All** to publish data for all period/scenario associations.



Import the Measures

For information on importing your performance measures into SAS Strategy Management, see “Import Performance Measures into SAS Strategy Management” on page 451.

Import Performance Measures into SAS Strategy Management

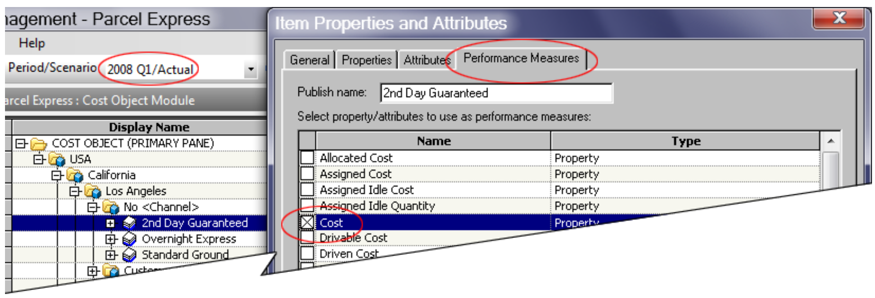
Summary

After publishing the measures, log on to SAS Strategy Management to import the measures

This section describes a simple example of importing a single performance measure into SAS Strategy Management to create a minimal scorecard. This example assumes the following performance measures have been set in SAS Activity-Based Management:

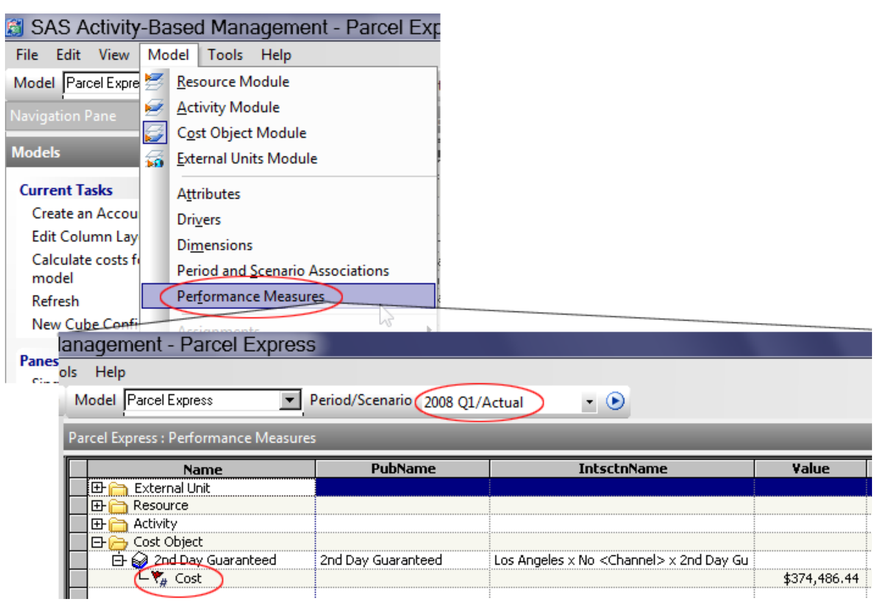
Module	Account	Period/Scenario	Performance Measure
Cost Object	2nd Day Guaranteed	2008/Actual	Cost
Costs Object	2nd Day Guaranteed	2008/Plan	Cost

The following picture shows a performance measure of Cost being set on the 2nd Day Guaranteed account in 2008/Actual.



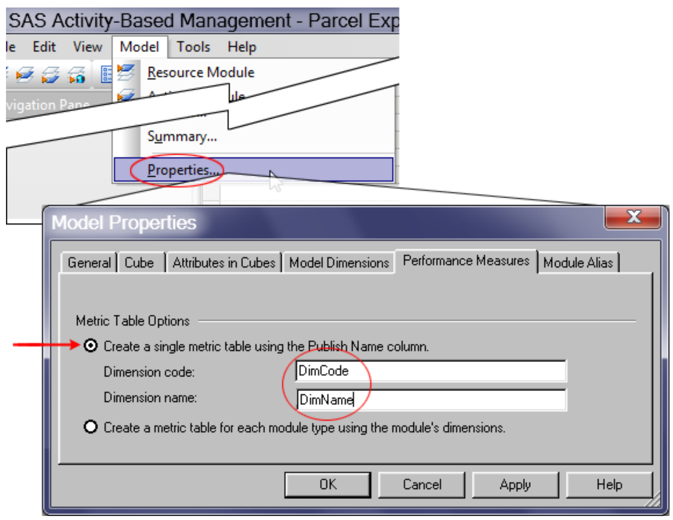
And a performance measure of Cost is also set on the 2nd Day Guaranteed account in 2008/Plan.

The following picture shows the Performance Measure page for 2008/Actual. (The Performance Measure page for 2008/Plan is not shown, but is similar.)



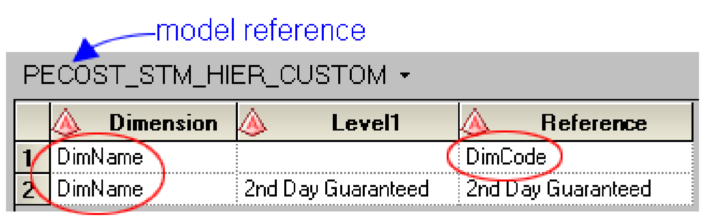
The following picture shows how to select the option to **Create a single metric table** for importing into SAS Strategy Management when performance measures are published from SAS Activity-Based Management. The picture also shows that columns in the table are to be named as follows:

Column Description	Column Name
Dimension code	DimCode
Dimension name	DimName



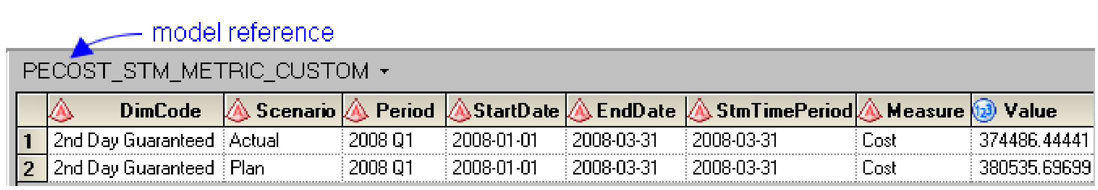
The following picture shows the two tables that are created in the SAS Activity-Based Management database when performance measures are published, given the assumptions described earlier. The tables are automatically registered in metadata with the SAS Metadata Server so that they are accessible by SAS Strategy Management.

Hierarchy table:



	Dimension	Level1	Reference
1	DimName		DimCode
2	DimName	2nd Day Guaranteed	2nd Day Guaranteed

Metric (content) table:



	DimCode	Scenario	Period	StartDate	EndDate	StmTimePeriod	Measure	Value
1	2nd Day Guaranteed	Actual	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	374486.44441
2	2nd Day Guaranteed	Plan	2008 Q1	2008-01-01	2008-03-31	2008-03-31	Cost	380535.69699

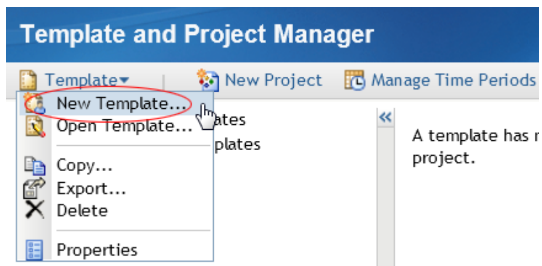
Importing the performance measures entails the following steps:

1. "Create a New Template" on page 454
2. "Create a New Project" on page 456
3. "Import the Hierarchy Table" on page 457
4. "Import the Metric Table" on page 460
5. "View the Scorecard" on page 465

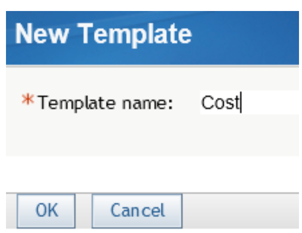
Create a New Template

1. Log on to SAS Strategy Management.

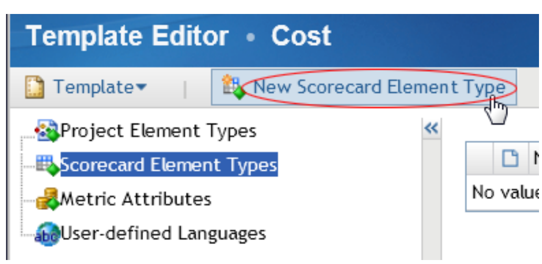
2. Select **Template** ⇨ **New Template**



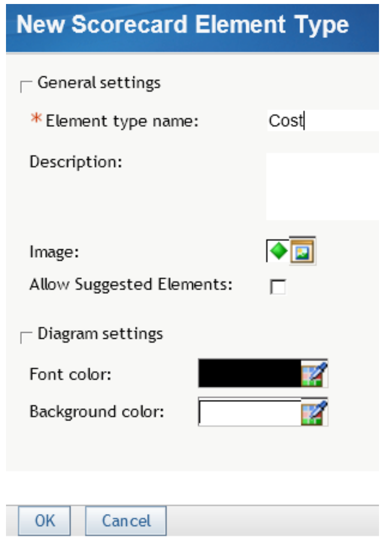
3. Name the template and click **OK**.



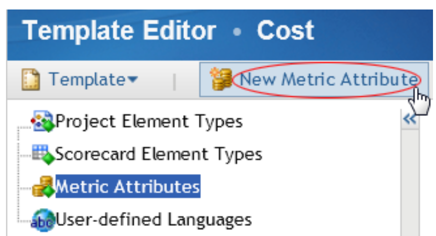
4. Select **Scorecard Element Types** and click **New Scorecard Element Type**.



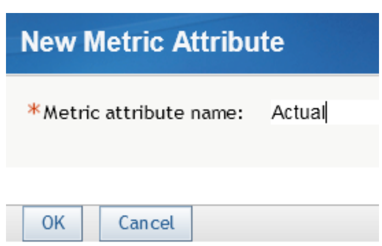
5. Name the scorecard element type and click **OK**.



6. Select **Metric Attributes** and click **New Metric Attribute**.

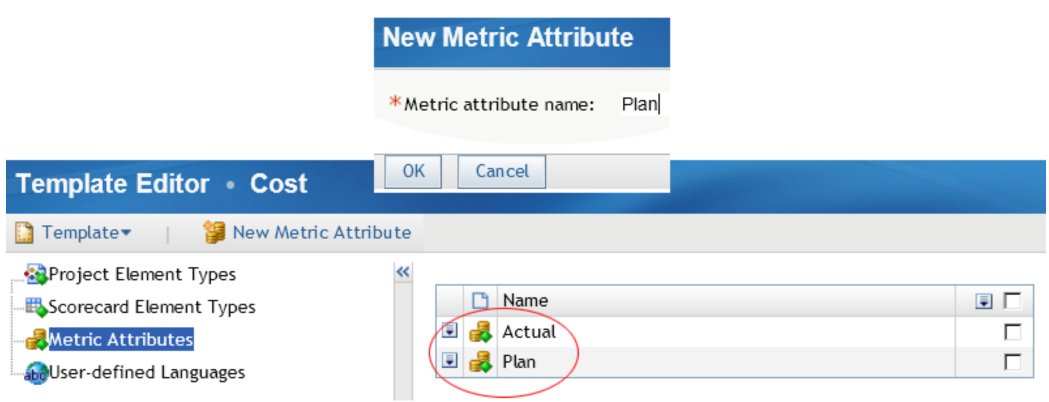


7. Name the metric attribute *Actual* and click **OK**.



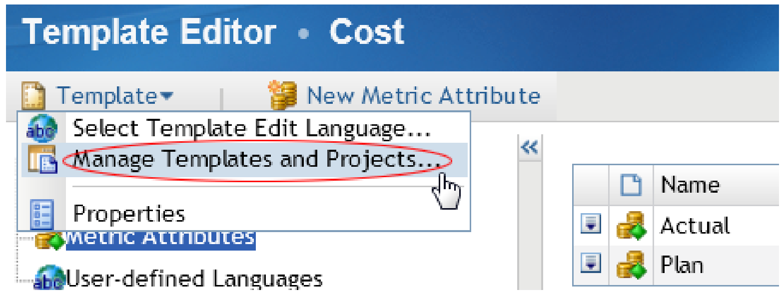
8. Repeat the same procedure to create another metric attribute. That is, Select **Metric Attributes** and click **New Metric Attribute**.

   Name the new metric attribute *Plan* and click **OK**.

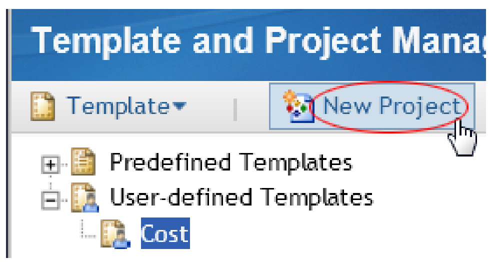


Create a New Project

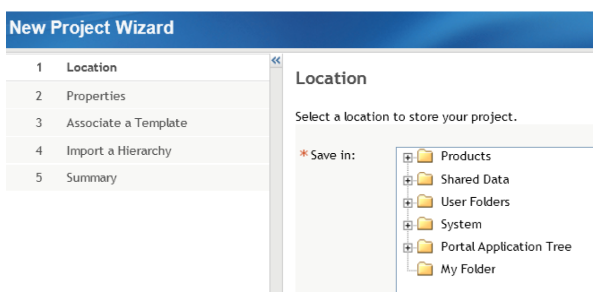
1. Select **Template** ⇨ **Manage Templates and Projects**.



2. Click **New Project**.

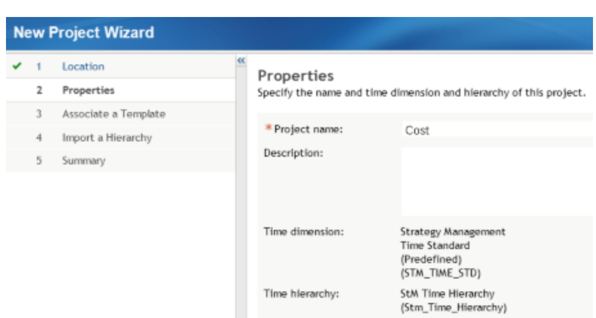


3. Select a folder for saving the project, and then click **Next**.

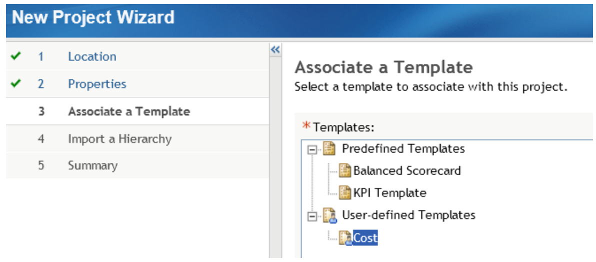


4. Name the project and select a time dimension, and then click **Next**.

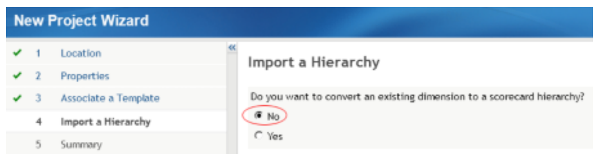
   Note: Your time dimension should be equivalent to the time dimension of your SAS Activity-Based Management period dimension, which you can import into SAS Strategy Management.



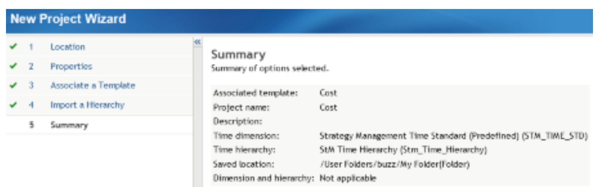
5. For the **Associate a Template** step, simply click **Next**. The project is associated with the template that you just created.



6. For the **Import a Hierarchy** step, select **No**, and then click **Next**. We are going to import the SAS Activity-Based Management hierarchy table in another way.



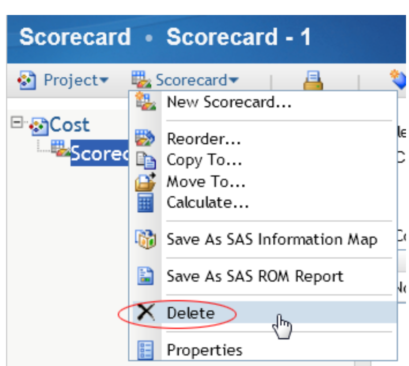
7. The project summary is displayed. Click **Finish**.



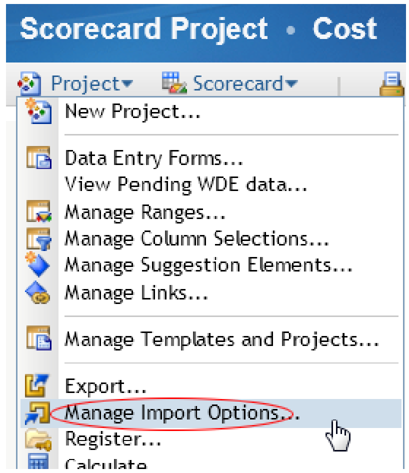
Import the Hierarchy Table

1. Select the scorecard that is automatically created for the new project, and then select **Scorecard** ⇨ **Delete**.

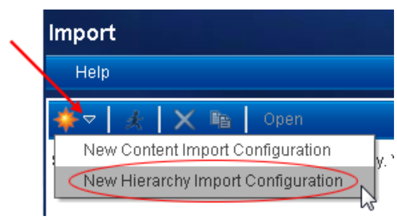
   Note: When you create a new project, a skeletal scorecard is automatically created. You can delete this scorecard because we are going to create one by importing the tables from SAS Activity-Based Management.



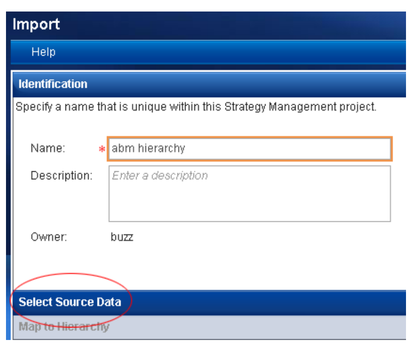
2. Select **Project** ⇨ **Manage Import Options**.



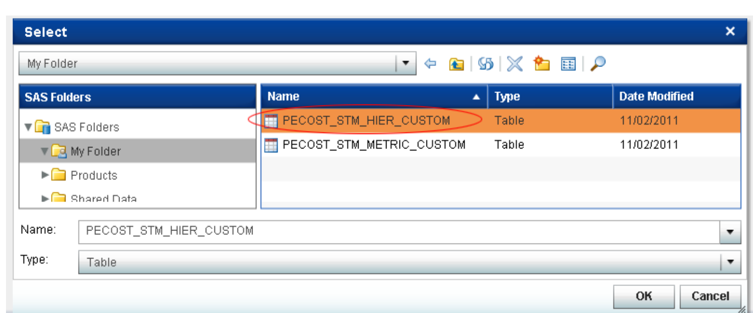
3. Select **New Hierarchy Import Configuration** from the **New import configuration** drop-down menu.



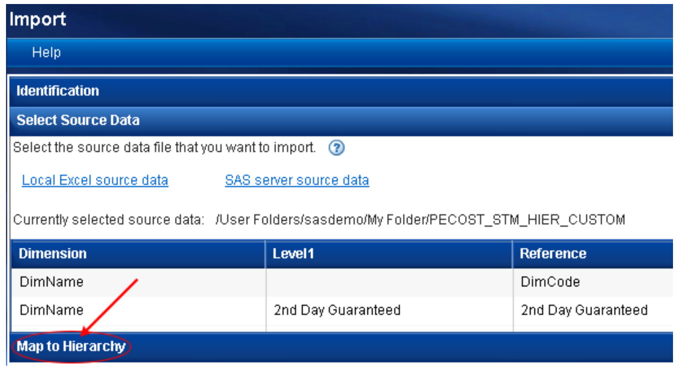
4. Name the hierarchy import configuration, and then click **Select Source Data**.



5. Browse to the table <*model_reference*>**_STM_HIER_CUSTOM**; select it and then click **OK**.



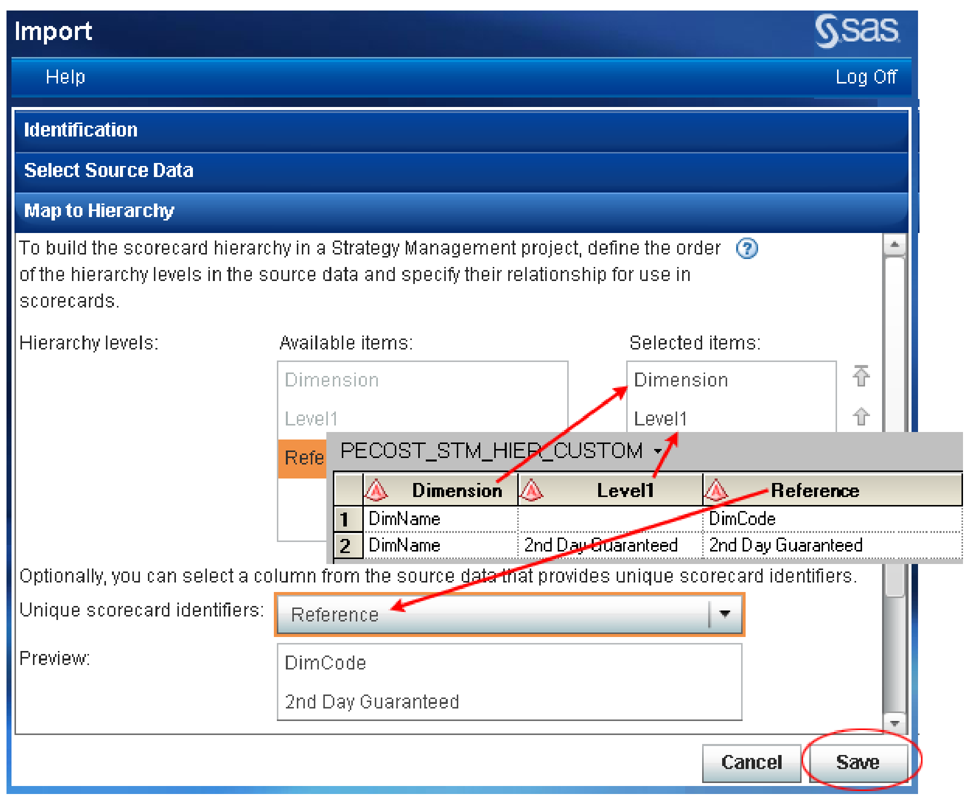
6. After the metadata has been read, click **Map to Hierarchy**.



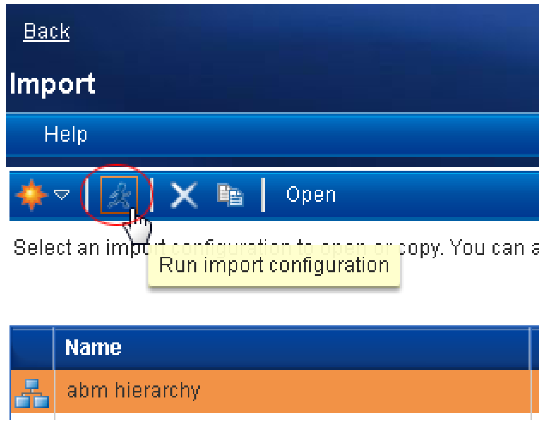
7. For **Hierarchy Levels**, select:
   - **Dimension**
   - **Level1**

   For **Unique Scorecard Identifiers**, select:
   - **Reference**

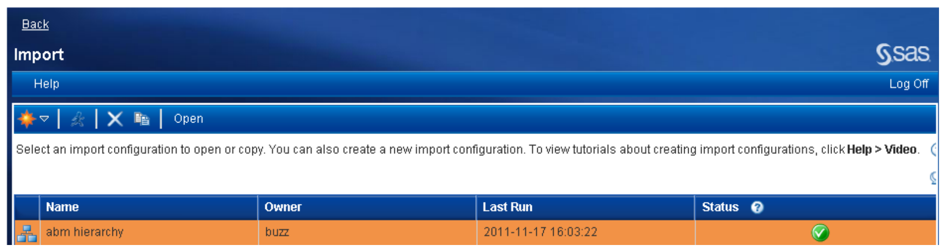
   Then click **Save**.



8. Select the hierarchy and click **Run**.

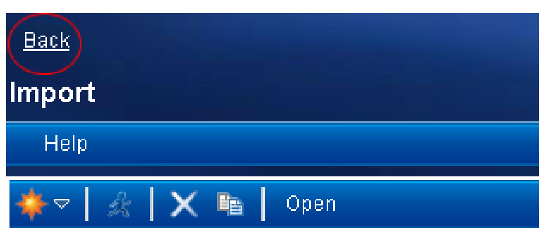


The hierarchy table is imported.

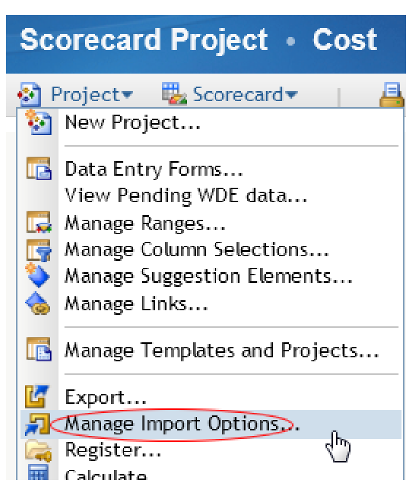


Import the Metric Table

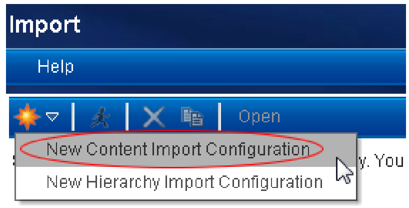
1. Click **Back**.



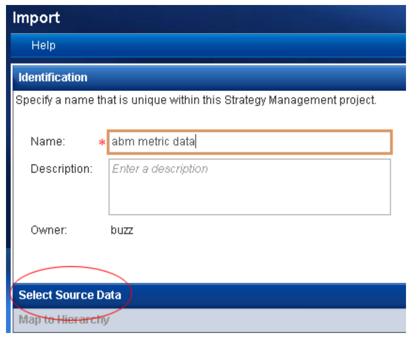
2. Select **Project** ⇨ **Manage Import Options**.



3. Select **New Content Import Configuration** from the **New import configuration** drop-down menu.

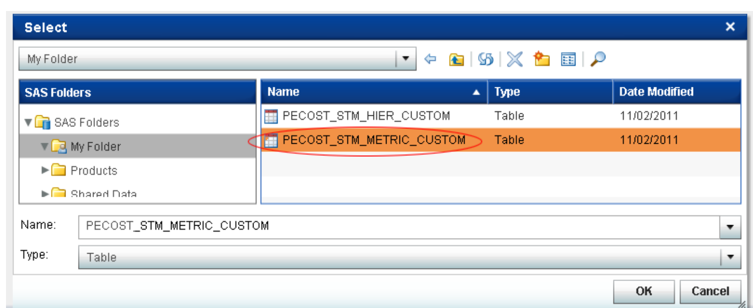


4. Name the content import configuration, and then click **Select Source Data**.

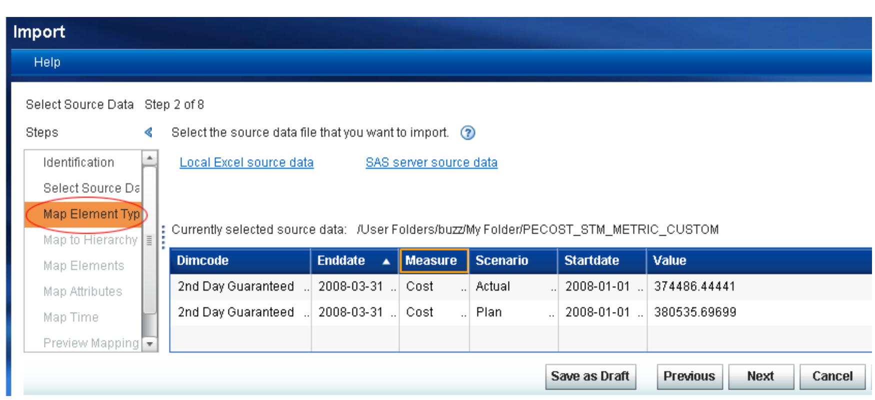


5. Select **Sas server source data**.

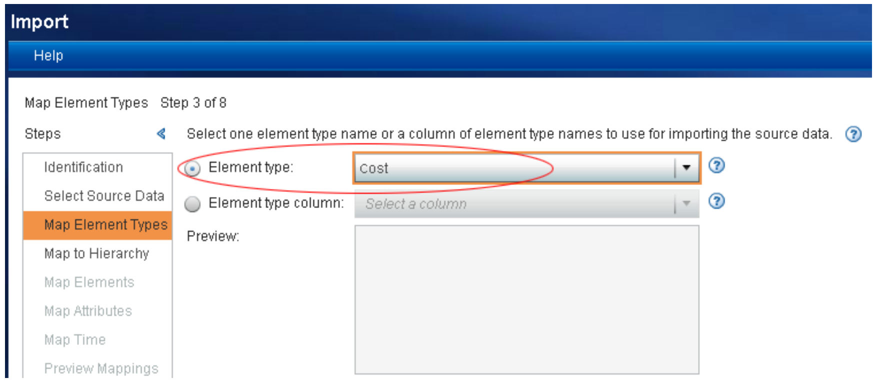
6. Browse to the table <*model_reference*>**_STM_METRIC_CUSTOM**; select it and then click **OK**.



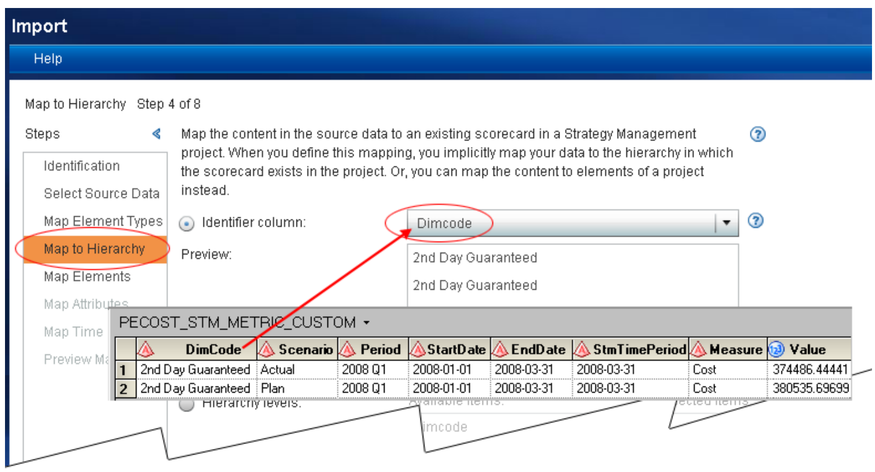
7. After the metadata has been read, click **Map Element Type**.



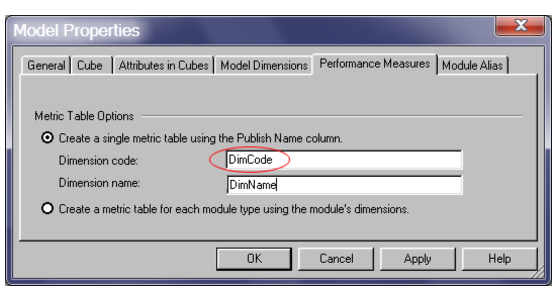
8. For **Element Type**, select the element type that you specified in creating the new template. See “Create a New Template” on page 454.



9. Click **Map to Hierarchy**. For the **Identifier Column**, select **DimCode**.



Remember that **DimCode** is the name that we are assuming was entered for the **Dimension code** in the model properties of our model. See “Choose the Table Format for Publishing” on page 447.



10. Click **Map elements**, and then:

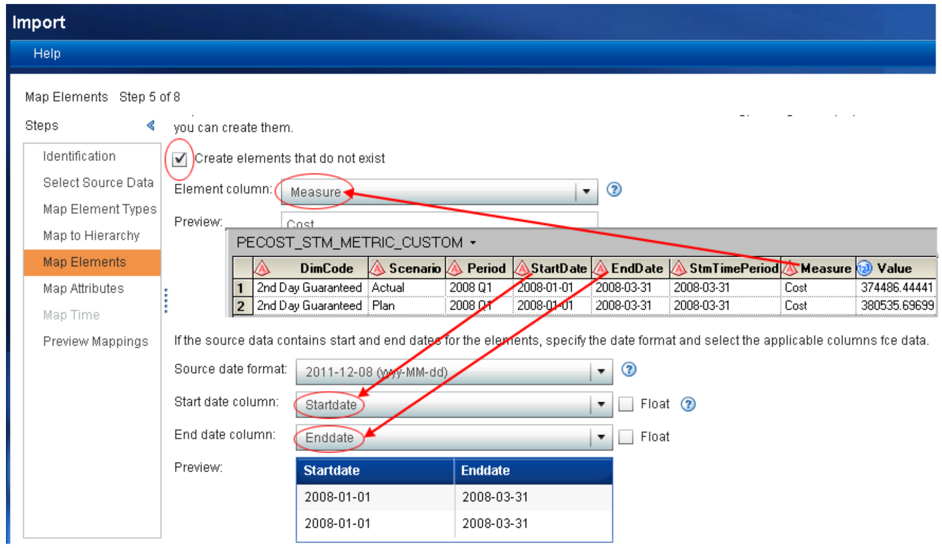
    a. Select **Create elements that do not exist**.

    b. For **Element column**, select **Measure**.

    c. For **Source date format**, select **2011–11–17 (yyyy-MM-dd)**

    d. For **Start date column**, unselect **Float** and then select **Startdate**.

    e. For **End date column**, unselect **Float** and then select **Enddate**.

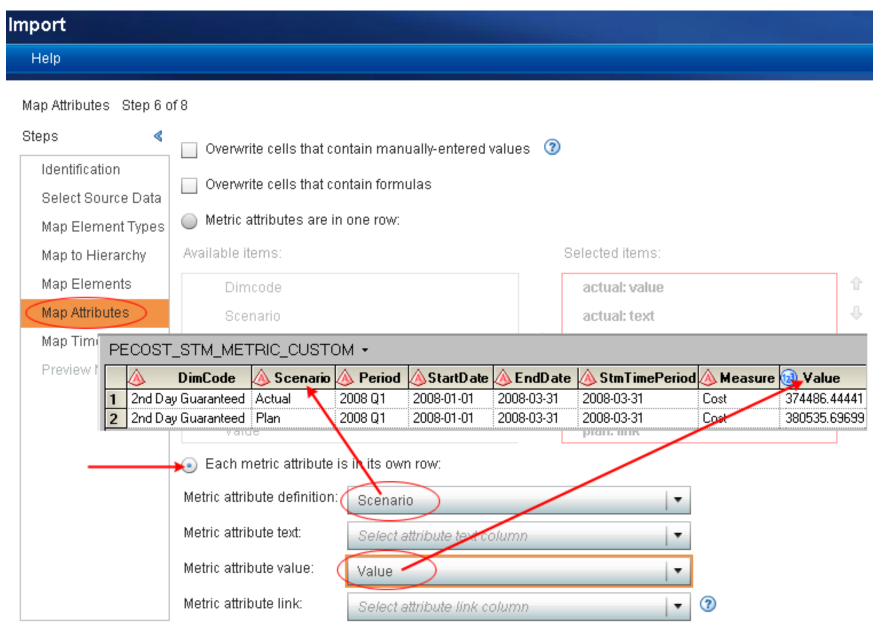


11. Click **Map Attributes**, and then:

   a. Select **Each metric attribute is in its own row**.

   b. For **Metric attribute definition**, select **Scenario**.

   c. For **Metric attribute value**, select **Value**.

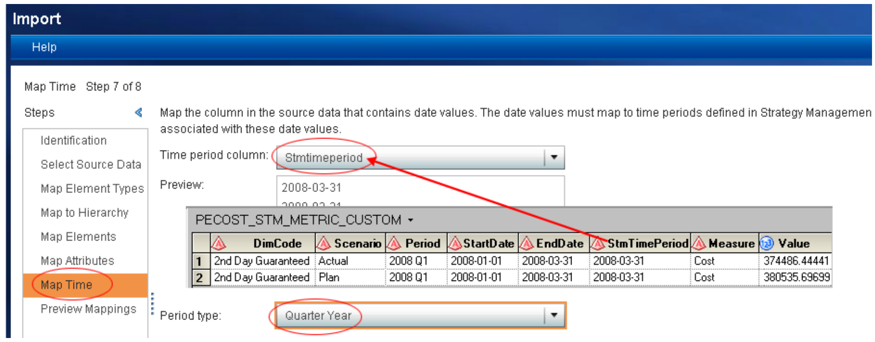


12. Click **Map Time**, and then:

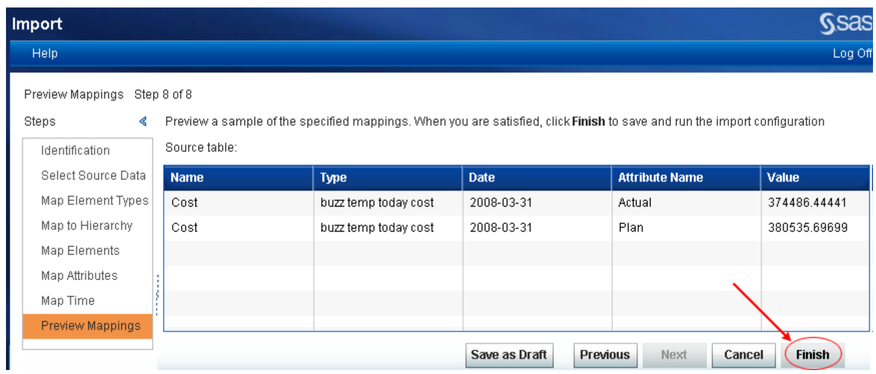
   a. For **Time period column**, select **StmTimePeriod**.

   Note: StmTimePeriod is a column that is created specifically for use with SAS Strategy Mangement, which requires a Time period column distinct from the Start date and End date columns. The StmTime period column contains a copy of the data in the End date column.

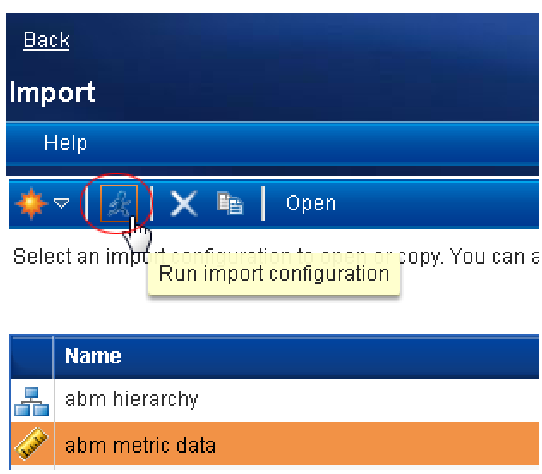
   b. For **Period type**, select **Quarter Year**.



13. Select **Preview Mappings**, and then click **Finish**.



14. Select the content table and click **Run**.

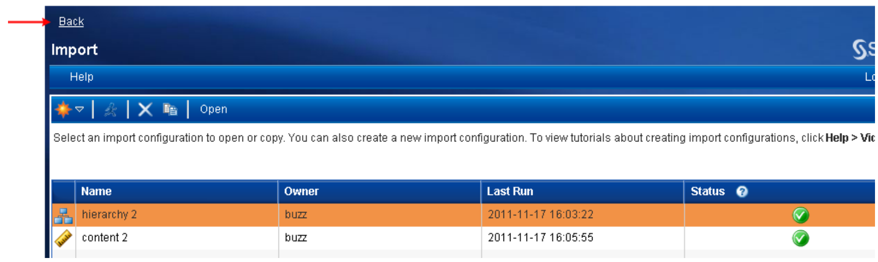


The content table is imported.

View the Scorecard

To view the scorecard:

1. Click **Back**.



2. Do the following:

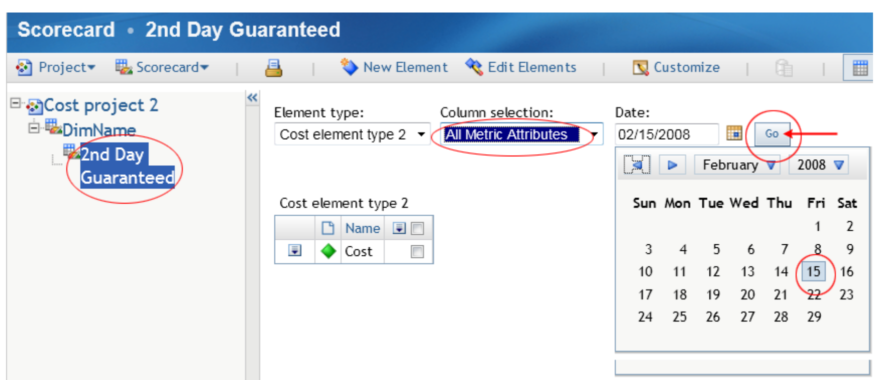
   a. Expand the project and select **2nd Day Guaranteed**.

   b. For **Column selection**, select **All Metric Attributes** to display Actual and Plan together (Select **Actual** or **Plan** to show only one.)

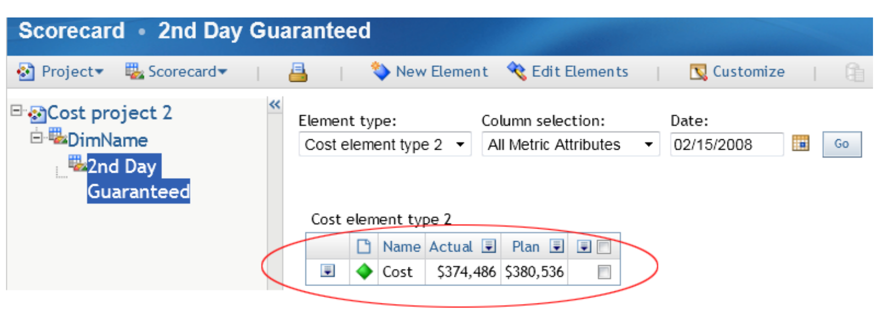
   c. For **Date**, select any date between the Startdate and the Enddate.

   Note: In selecting the date, you must select a day as well as a month and year.

   d. Click **Go**.



The scorecard is displayed.



Chapter 28
SAS Enterprise Guide

Using the Add-In for SAS Enterprise Guide

Overview

You can view SAS Activity-Based Management data using SAS Enterprise Guide. To view the data you must first add the SAS Activity-Based Management Add-In for Enterprise Guide.

Note: Because the SAS Activity-Based Management client runs as 32–bit, you can use the Activity-Based Management Add-In for Enterprise Guide only on the 32–bit version of SAS Enterprise Guide.

Adding the Add-In

1. Open SAS Enterprise Guide.

   Note: Connect to the machine that contains the instance of SAS that you will use to retrieve the SAS Activity-Based Management data from the ABM database. To change the machine connection, do the following:

   a. Select **Tools** ⇨ **Options** from the SAS Enterprise Guide menu bar.

   b. Click **Administration**.

   c. Click the **Modify** button (under Connections).

   d. Click **Add** to add a connection to a server machine.

2. Select **Tools** ⇨ **Add-in** ⇨ **Add-In Manager** .

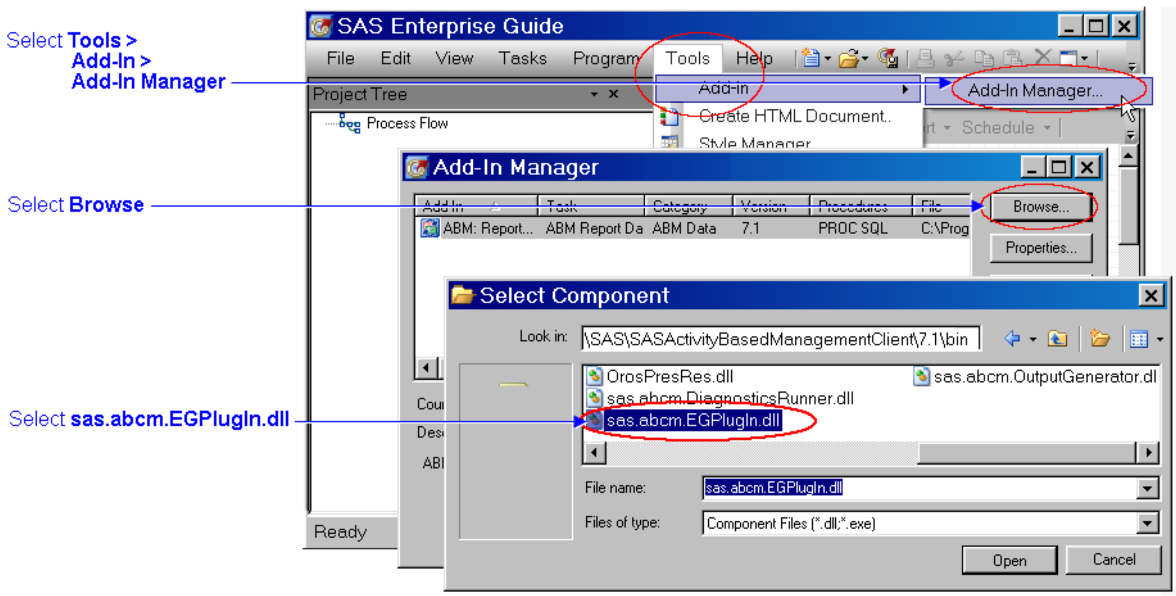
   The Add-In Manager opens.

3. Click **Browse** and browse to the SAS Activity-Based Management Add-In for Enterprise Guide (sas.abcm.EGPlugIn.dll). The .dll file for the add-in is installed along with the SAS Activity-Based Management Client.

   Add-In
   : **sas.abcm.EGPlugIn.dll**

   Path
   : (installation directory)**SASHome\SASActivityBasedManagementClient\7.2\bin**

4. Click **Open**.



Running the Add-In

1. Open SAS Enterprise Guide

   Note: Connect to the machine that contains the instance of SAS that you will use to retrieve the SAS Activity-Based Management data from the ABM database.

2. Select **Tools** ⇨ **Add-in** ⇨ **ABM Report Data Selection**.

   The SAS Activity-Based Management Log On window appears.

3. Log on to the SAS Activity-Based Management Metadata Server from which you want to retrieve data to use inside SAS Enterprise Guide.

   The Report Data window appears.

4. Specify the following information concerning the report data that you want to view in SAS Enterprise Guide:

   - Type of report data (Destination Furthest, Resource Contributions, ..., Unit Cost)
   - Model or models
   - Periods and scenarios
   - Modules within the model(s)
   - Report filters
   - Report options

5. Select the **IOM (Integrated Object Model) Destination**.

   IOM Server
   : Specify the IOM Server that is to be used to retrieve the data.

     Note: The IOM Server resides on the server machine to which you connected in starting SAS Enterprise Guide.

   Library and Data Set
   : Specify the SAS Activity-Based Management data set where the SAS Activity-Based Management data will be stored.

     Note: The libraries listed in the drop-down list are those that are assigned to the IOM Server that you select.

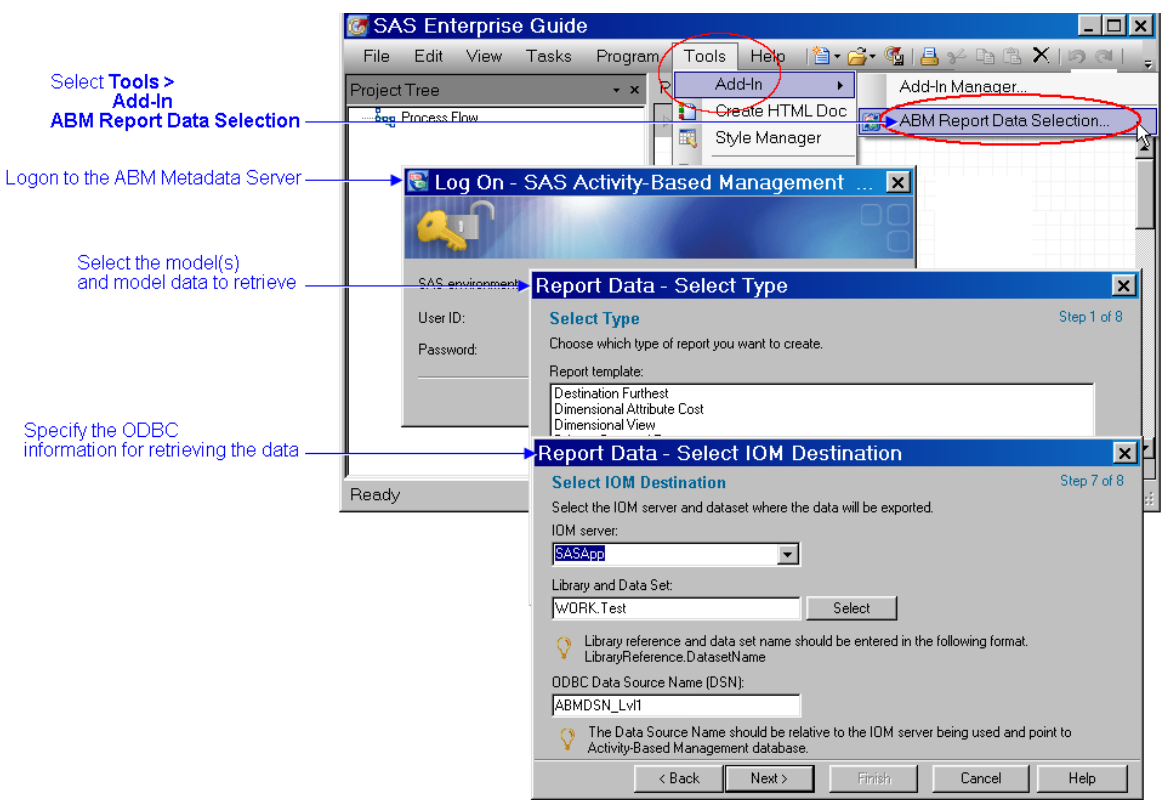
   ODBC Data Source Name (DSN)
   : Use ODBC to make a connection between the machine to which SAS Enterprise Guide is connected and the SAS Activity-Based Management database. The data source name (DSN) must be on the same machine as the IOM Server.

     To display a list of data source names on the machine on which the IOM Server is running:

     a. Select **Start** ⇨ **Settings** ⇨ **Control Panel**.

     b. Select **Administrative Tools**.

     c. Select **Data Sources (ODBC)**.



6. Click **OK**.

   The data is written to the data set which you can now view and analyze with SAS Enterprise Guide.

Using SAS Enterprise Guide

For information on SAS Enterprise Guide, see the product documentation page at support.sas.com/documentation/onlinedoc/guide.

Note: To avoid having metadata be out of synch with the data in your Metadata library, you might have to set the **METAOUT** option for the Metadata library.

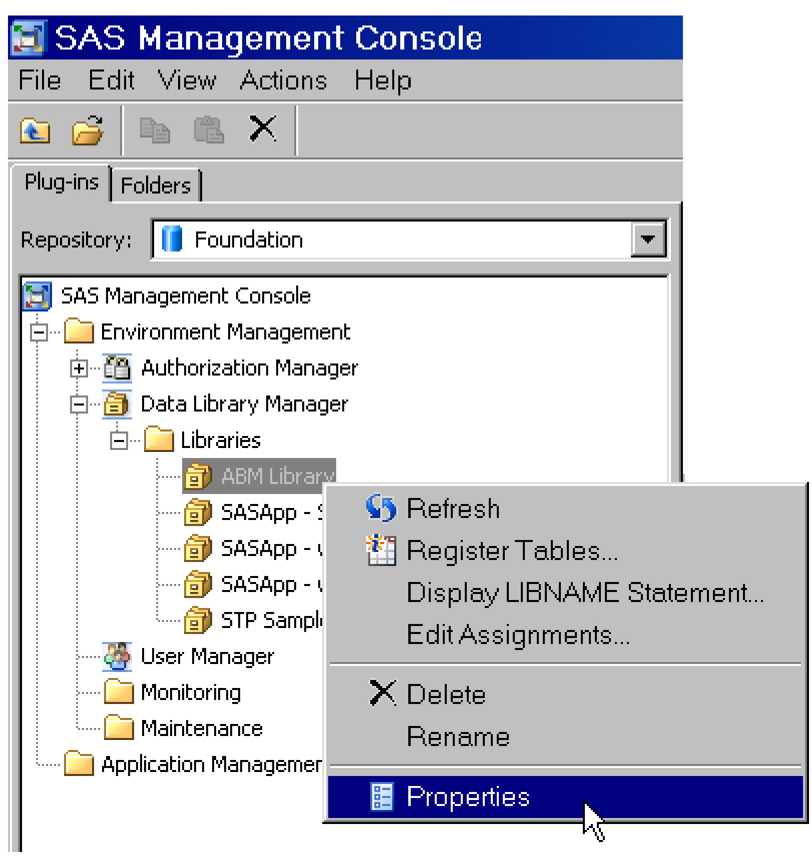
Set the METAOUT Option for the Metadata Library

When generating a SAS Activity-Based Management report using SAS Enterprise Guide, you might encounter the following setup error:

```
ERROR: You cannot create or delete datasets, views or indexes in this mode.
Try the option METAOUT=DATA. Use Proc Metalib to create metadata for datasets.
```

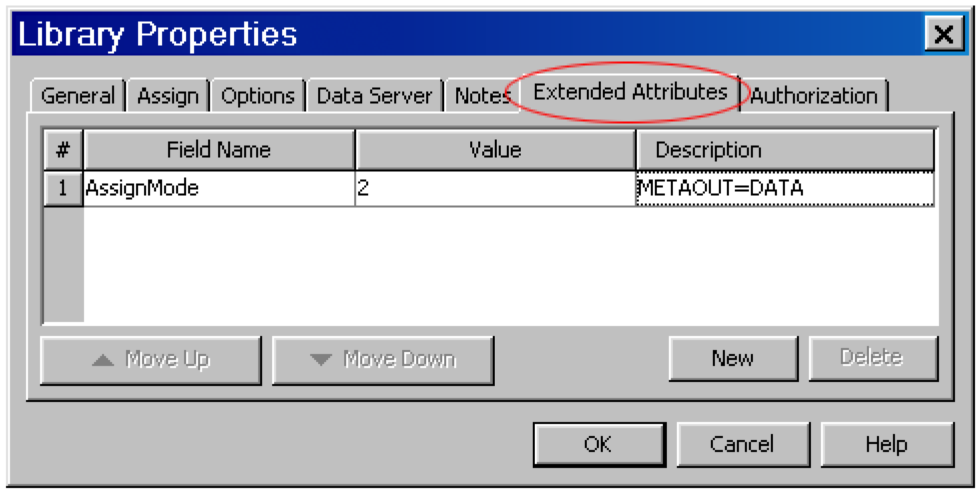
To circumvent this problem, try settiing the **METAOUT** option of your metadata library to be DATA. To do so, do the following:

1. Open SAS Management Console connected to your SAS Metadata Server.
2. Click the **Plug-ins** tab.
3. Expand **Data Library Manager**.
4. Expand **SAS Libraries**.
5. Right-click the library in question and select **Properties**.



6. Click the **Extended Attributes** tab
7. Click **New** and enter the following information:

Field Name	Value	Description
AssignMode	2	METAOUT=DATA



8. Click **OK** to close the Properties window.
9. Right-click **Libraries** and select **Refresh**.
10. Restart SAS Enterprise Guide and confirm that the library is now available

Part 9

The ABC Procedure

Chapter 29

PROC ABC

Overview

Before SAS Activity-Based Management 6.4, the implementation of calculate and fact-table generation was specific to the 32-bit Microsoft Windows operating system. This resulted in limitations:

- The size of a model was constrained by memory. The data for a whole period/scenario was loaded into memory to guarantee fast performance. Using 32-bit Windows limited the size of a model to less than 3 gigabytes.
- Customers were not allowed to leverage other high-performance platforms when performance was an issue, particularly when generating fact tables.

These limitations restricted what customers could do and prevented SAS Activity-Based Management from working with customer models without additional work. In some cases, the limitations required users to re-engineer the structure of their model (a non-trivial task). The larger the models, the more problematic the situation became.

Starting with SAS Activity-Based Management 6.4, you could run the ABC procedure interactively using the SAS interface or the SAS Activity-Based Management client interface. SAS Activity-Based Management uses the ABC procedure internally to calculate in the client. In addition, the ability to query contributions allows users to dynamically ask questions about a calculated model.

In SAS Activity-Based Management 7.2 you can use the ABC Procedure to validate a model and to generate a fact table.

The following databases are supported:

- Microsoft SQL Server 2005
- Oracle 11g
- MySQL 5

Note: If you want to run the ABC Procedure in a SAS session outside of SAS Activity-Based Management, then you must set the CLASSPATH variable for the following JAR files:

- sas.solutions.abm.contributions.jar
- commons-logging.jar
- log4j.jar
- sas.antlr.jar
- sas.solutions.abm.contributions.nls.jar (for localized versions)

Syntax

```
PROC ABC
   CONNECTION='connection-string'
   DIAGNOSTICS=CALC | QUERY
   DSN='odbc-dataset-name'
   PASSWORD='password'
   USERID='userid';

   ATTRIBUTE 'attr-reference'
       ID=attribute-id;

   CALCULATE
       DRIVERSEQLIMIT=sequence-limit-number
       DRIVERRULES=ENABLE | DISABLE or ON | OFF
       MSGLIMIT=message-limit-number
       OPID='operation-string';

   DIMENSION 'stage-reference'
       ID=dimension-id
       LEVEL=level-id
       REFERENCE=dimension-reference | REF=dimension-reference;

   FACTGEN MSC | SSC | RC | RCI
       ATTRIBUTES=('attr-reference-1' ... 'attr-reference-n')
       STAGES=('stage-reference-1' ... 'stage-reference-n')
       FILTERS=('ID-1' ... 'ID-n')
       CUBEACTION=GENERATEROWS | COUNTROWS
       CUBECONFIGID=cube-configuration-id
       FILTERTYPE=ACCOUNT | MODULE
       NAME='fact-table-name'
       OPID=operation-id
       STAGEDEF=MODULES | STAGES
       SUPPRESSZEROCOSTS | SUPZEROCOSTS=NO | YES;

   LIST

   LOAD
       MODEL='model-name'
       MODELID=model-id
       MODELNUM='model-number1, model-number2,..., model-numbern'
       PERIOD=ALL | '<path/> period-name'
       PERIODID=period-id
       SCENARIO=ALL | '<path/>scenario-name'
       SCENARIOID=scenario-id;
```

```
MODELVALIDATE
    EMPTYATTRIBUTE | EMPTYATTR
    MSGLIMIT=maximum-messages-in-log
    NEGATIVEDRIVERQUANTITYASSIGNMENT | NEGDRVQTYASGN
    OVERDRIVENSOURCEACCOUNT | OVRDRVSRCACT
    UNASSIGNEDACCOUNT | UNSGNACT
    ZEROCOSTACCOUNT | ZCOSTACT;

QUERY
    'query-string' </ OUT='output-dataset-name'>;

STAGE 'stage-reference'
   ID=stage-id
   CASHFLOW=IN | OUT
   TYPE=MODULE | STAGE;
```

PROC ABC Statement

Overview

The PROC ABC statement invokes the procedure.

```
PROC ABC
   CONNECTION='connection-string'
   DIAGNOSTICS=CALC | QUERY
   DSN='odbc-dataset-name'
   PASSWORD='password'
   USERID='userid';
```

Required Arguments

CONNECTION='connection-string'
: Specifies optional values for connecting to the database.

DIAGNOSTICS=CALC | QUERY
: Specifies diagnostics for calculate code and query code. If you select this option, the system checks to make sure that the procedure has everything it needs to work.

DSN='odbc-dataset-name'
: Specifies the data set name for an ODBC connection. If you do not specify a name, then ABMDSN is assumed.

PASSWORD='password'
: Specifies the password that is associated with the database specified with the DSN= option.

USERID='userid'
: Specifies the user ID that is associated with the database specified with the DSN= option.

ATTRIBUTE Statement

Overview

Each ATTRIBUTE statement specifies a numeric attribute that is to be included in fact table generation.

Note: ATTRIBUTE statements are required only when generating a fact table for a Resource Contributions cube (RC) or a Multi-Stage Contributions cube (MSC).

```
ATTRIBUTE 'attr-reference'
    ID=attribute-id;
```

Required Arguments

'*attr-reference*'
: A quoted string that specifies the reference of the numeric attribute. Go to the Attributes page to see an attribute's reference.

ID=*attribute-id*
: Specifies the ID in the database of the numeric attribute.

CALCULATE Statement

Overview

The CALCULATE statement invokes the calculate code to operate on the model that is specified on the LOAD statement. Parameters that are needed for the calculate code to operate are specified. Any model options that are specific to calculate are also specified.

```
CALCULATE
    DRIVERSEQLIMIT=sequence-limit-number
    DRIVERRULES=ENABLE | DISABLE or ON | OFF
    MSGLIMIT=message-limit-number
    OPID='operation-string';
```

Required Arguments

DRIVERSEQLIMIT=*sequence-limit-number*
: Specifies the maximum number of times that recursive driver calculations are performed.

DRIVERRULES=ENABLE | DISABLE or ON | OFF
: Specifies whether calculation is to create rule-based assignments or not. The default is ENABLE i.e., calculation creates assignments for those drivers whose assignments are derived based upon a rule formula. You can disable rule-based assignment to decrease calculation time.

MSGLIMIT=*message-limit-number*
Specifies the maximum number of messages for each type of warning or error that is reported to the client. This option allows you to limit the number of messages that are reported for models that report numerous messages.

OPID='*operation-string*'
Specifies a string that is used by the SAS Activity-Based Management client to uniquely identify the connection.

DIMENSION Statement

Overview

Each DIMENSION statement specifies a dimension that is to be included in fact table generation.

Note: DIMENSION statements are required only when generating a fact table for a Multi-Stage Contributions cube (MSC).

```
DIMENSION 'stage-reference'
    ID=dimension-id
    LEVEL=level-id
    REFERENCE=dimension-reference | REF=dimension-reference;
```

Required Arguments

'*stage-reference*'
A quoted string that specifies the module or stage in which the dimension occurs:

Module Reference
In a model in which there are no stages per se, each module is considered as defining a single stage (one stage per module). Following are the stage-references to specify for each module: 'ExternalUnit' 'Resource' 'Activity' 'CostObject'

Stage Reference
In a model that uses stages, the stage-reference is the reference of the attribute that defines the stage for example, 'Parcel Handling'. See Add Stage Attributes to Accounts.

ID=*dimension-id*
Specifies the database identifier of the dimension that is to be included in the fact table.

REFERENCE=*dimension-reference* | REF= *dimension-reference*
Specifies the dimension reference. To see a dimension reference, go to the Dimensions page of a model.

Options

LEVEL=*level-id*
Specifies the database identifier of the level up to which (and including) the dimension is to be included in the fact table. If LEVEL= is not specified, then all levels are included.

FACTGEN Statement

Overview

The FACTGEN statement specifies the type of fact table to be generated.

```
FACTGEN MSC | SSC | RC | RCI
    ATTRIBUTES=('attr-reference-1' ... 'attr-reference-n')
    STAGES=('stage-reference-1' ... 'stage-reference-n')
    FILTERS=('ID-1' ... 'ID-n')
    CUBEACTION=GENERATEROWS | COUNTROWS
    CUBECONFIGID=cube-configuration-id
    FILTERTYPE=ACCOUNT | MODULE
    NAME='fact-table-name'
    OPID=operation-id
    STAGEDEF=MODULES | STAGES
    SUPPRESSZEROCOSTS | SUPZC=NO | YES;
```

Required Arguments

MSC | SSC | RC | RCI
Specifies the type of fact table to be generated:

MSC	Multi-Stage Contributions
SSC	Single-stage Contributions
RC	Resource Contributions
RCI	Report Table generation

ATTRIBUTES=('*attr-reference-1*' ... '*attr-reference-n*')
Specifies the numeric attributes to be included in the fact table.

Note: Specifying ATTRIBUTES= is required only when generating a fact table for a Multi-Stage Contributions cube (MSC) or a Resource Contributions cube (RC).. Each attribute in the list points back to an ATTRIBUTE statement that specifies a numeric attribute to be included in the fact table. The ATTRIBUTE statement contains details about an individual attribute, and the ATTRIBUTES= option in the FACTGEN statement lists the attributes that are to be included in the fact table. If an attribute is not included in the ATTRIBUTES= list, then it is not included in the fact table even it has an ATTRIBUTE statement.

```
ATTRIBUTE 'Average Time To Expedite' ID=1070;
ATTRIBUTE 'Number of Inspections' ID=1071;
FACTGEN RC
    ATTRIBUTES=('Average Time To Expedite' 'Number of Inspections')
```

STAGES=('*stage-ref1*' ... '*stage-refn*')
: Specifies the modules or stages to be included in the fact table.

Note: Specifying STAGES= is required only when generating a fact table for a Multi-Stage Contributions cube (MSC).Each stage-ref in the list points back to a statement that specifies a module or stage to be included in the fact table. The STAGE statement contains details about an individual module or stage, and the STAGES= option in the FACTGEN statement lists the stages that are to be included in the fact table. If a module or stage is not included in the STAGES= list, then it is not included in the fact table even it has a STAGE statement.

```
STAGE'Stage 1 Resources'ID=1047 COSTFLOW=IN;
STAGE'Stage 2 Customer Service'ID=1048 COSTFLOW=OUT;
FACTGEN RC
   STAGES=('Stage 1 Resources' 'Stage 2 Customer Service')
```

CUBECONFIGID=*cube-configuration-id*
: Specifies the database identifier of the cube configuration to be used in processing the fact table.

FILTERS=('*ID-1*' ... '*ID-n*')
: This option is required only for Report table generation (RCI). It specifies the IDs of either the accounts or modules to be included in the generated Report table. Use the FILTERTYPE= option to specify whether the IDs are of accounts or of modules.

NAME=*fact-table-name*
: Specifies the fact table name.

Options

CUBEACTION=GENERATEROWS | COUNTROWS
: Specifies whether to generate rows or count rows. The default is to generate rows.

FILTERTYPE=ACCOUNT | MODULE
: This option is required if you include the FILTERS= option for Report table generation. FILTERTYPE= specifies whether the FILTERS= option includes the IDs of accounts or or modules.

OPID=*operation-id*
: Do not include this option it is generated by the system..

STAGEDEF=MODULES | STAGES
: Specifies whether the model uses modules or stages. The default is MODULES..

SUPPRESSZEROCOSTS | SUPZC=NO| YES
: Specifies whether or not to suppress zero costs. The default is NO.

LIST Statement

The LIST statement lists the models that are available in the database provided in the PROC statement. The list is numbered so that the LOAD statement can accept a model specification based on this number for operations.

```
LIST
```

LOAD Statement

Overview

The LOAD statement specifies the model that is being worked on. Execution of a LOAD statement sets up the model that is used for calculate or query.

```
LOAD
    MODEL='model-name'
    MODELID=model-id
    MODELNUM='model-number1, model-number2,..., model-numbern'
    PERIOD=ALL | '<path/> period-name'
    PERIODID=period-id
    SCENARIO=ALL | '<path/>scenario-name'
    SCENARIOID=scenario-id;
```

Required Arguments

DBTYPE=MSSQL | ORA | MYSQL
Specifies model database: Microsoft SQL Server, Oracle, or My SQL.

MODEL='*model-name*'
Specifies the model to be used for calculations.

MODELID=*model-id*
Specifies the ID of the model to be used for calculations.

MODELNUM='*model-number1*, *model-number2*,..., *model-numbern*'
Specifies one or more models to be calculated. The model numbers are returned as the output of the LIST statement.

PERIOD=ALL | '*<path/>period-name*'
Specifies the periods to be calculated.

PERIODID=*period-id*
Specifies the period to be calculated. Specify ALL to calculate all periods.

SCENARIO=ALL | '*<path/>scenario-name*'
Specifies the scenario to be calculated. Specify ALL to calculate all scenarios.

SCENARIOID=*scenario-id*
Specifies the ID of the scenario to be calculated.

MODELVALIDATE Statement

Overview

The MODELVALIDATE statement triggers model validation and lists the types of warnings that are to be flagged.

```
MODELVALIDATE
    EMPTYATTRIBUTE | EMPTYATTR
```

```
MSGLIMIT=maximum-messages-in-log
NEGATIVEDRIVERQUANTITYASSIGNMENT | NEGDRVQTYASGN
OVERDRIVENSOURCEACCOUNT | OVRDRVSRCACT
UNASSIGNEDACCOUNT | UNSGNACT
ZEROCOSTACCOUNT | ZCOSTACT;
```

Options

Specify the following types of warning messages to be included in the message log:

EMPTYATTRIBUTE | | EMPTYATTR
Flags an attribute without a value.

MSGLIMIT=*maximum-messages-in-log*
Specifies the maximum number of messages to be included in the message log.

NEGATIVEDRIVERQUANTITYASSIGNMENT | NEGDRVQTYASGN
Flags a driver whose driver quantity is negative.

OVERDRIVENSOURCEACCOUNT | OVRDRVSRCACT
Flags a source account whose driver is attempting to flow more costs than are contained in the account.

Note: The model must have been calculated for this warning message to be issued.

UNASSIGNEDACCOUNT | UNSGNACT
Flags an account that receives no assignments..

ZEROCOSTACCOUNT | ZCOSTACT
Flags an account whose cost is zero.

QUERY Statement

Overview

The QUERY statement invokes the contributions query code. The query operates on the model that is specified in the LOAD statement. Multiple QUERY statements can be specified after a single LOAD statement. The Contributions tab of SAS Activity-Based Managaement allows a user to query contributions from the interface. After performing a query on the Contributions tab, you can get the PROC ABC statement and paste it into an ASCII editor to use with your own SAS program.

```
QUERY
    'query-string' </ OUT='output-dataset-name'>;
```

Required Arguments

query-string
Specifies the query to be performed.

Options

OUT='*output-dataset-name*'
Specifies where output from the query goes. If OUT= is not specified, then a data set in the WORK library is created.

STAGE Statement

Overview

Each STAGE statement specifies a module or stage that is to be included in fact table generation.

Note: STAGE statements are required only when generating a fact table for a Multi-Stage Contributions cube (MSC).

```
STAGE 'stage-reference'
   ID=stage-id
   CASHFLOW=IN | OUT
   TYPE=MODULE | STAGE;
```

Required Arguments

'*stage-reference*'
A quoted string that specifies the module or stage to be included in fact table generation:

Module Reference
In a model in which there are no stages per se, each module is considered as defining a single stage (one stage per module). Following are the stage-references to specify for each module: 'ExternalUnit' 'Resource' 'Activity' 'CostObject'

Stage Reference
In a model that uses stages, the stage-reference is the reference of the attribute that defines the stage for example, 'Parcel Handling'. See Add Stage Attributes to Accounts.

ID=*stage-id*
Specifies the database identifier of a stage that is to be included in fact table generation.

Options

CASHFLOW=IN | OUT
Specifies whether to include cost flows into or out of the selected module or stage. The default is IN.

TYPE=MODULE | STAGE
Specifies whether the STAGE statement refers to a module or stage. The default is MODULE.

Rules Governing the ABC Procedure

Overview

- Last statement wins
- ATTRIBUTE, STAGE, and DIMENSION statements must come before a FACTGEN statement
- All attributes, stages, and dimensions must be defined before they can be referred to in a FACTGEN statement
- ATTRIBUTE, STAGE and DIMENSION statements can appear in any order before a FACTGEN statement
- Multiple FACTGEN statements can appear in one PROC ABC statement

When Multiple Statements Define the Same Item the Last Statement Wins

Example:

```
Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: ATTRIBUTE 'No of Employees' ID=100;
Line 3: STAGE 'Stage1' REF='Stage1';
Line 4: DIMENSION 'Stage1' REF='GL' LEVEL=5;
Line 5: ATTRIBUTE 'No of Employees' ID=500;
Line 6: FACTGEN MSC STAGES=('Stage1') ATTRIBUTES=('No of Employees');
Line 7: RUN;
```

Only Those ATTRIBUTE, STAGE, and DIMENSION Statements That Appear Before a FACTGEN Statement are Considered as Input to That Specific FACTGEN Operation

Example:

```
Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: ATTRIBUTE 'No of Employees' ID=100;
Line 3: STAGE 'Stage1' REF='Stage1';
Line 4: DIMENSION S1 REF='GL' LEVEL=5;
Line 5: ATTRIBUTE 'No of Employees' ID=500;
Line 6: FACTGEN MSC STAGES=(S1) ATTRIBUTES=('No of Employees');
Line 7: ATTRIBUTE 'Performance' ID=200;
Line 8: RUN;
```

All Attributes, Stages and Dimensions Must Be Defined Before They Are Referred to in a FACTGEN Statement

Example:

```
Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: ATTRIBUTE 'No of Employees' ID=100;
```

```
Line 3: STAGE S1 REF='Stage1';
Line 4: DIMENSION S1 REF='GL' LEVEL=5;
Line 5: ATTRIBUTE 'No of Employees' ID=500;
Line 6: FACTGEN MSC STAGES=(S1) ATTRIBUTES=('No of Employees', 'Performance');
Line 7: ATTRIBUTE 'Performance' ID=200;
Line 8: RUN;
```

ATTRIBUTE, STAGE and DIMENSION Statements Can Appear in Any Order Before a FACTGEN Statement

Example:

The following two procedure invocations are equivalent:

```
Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: ATTRIBUTE 'No of Employees' ID=100;
Line 3: STAGE S1 REF='Stage1';
Line 4: DIMENSION S1 REF='GL' LEVEL=5;
Line 5: FACTGEN MSC STAGES=(S1) ATTRIBUTES=('No of Employees');
Line 6: RUN;

Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: DIMENSION S1 REF='GL' LEVEL=5;
Line 3: ATTRIBUTE 'No of Employees' ID=100;
Line 4: STAGE S1 REF='Stage1';
Line 5: FACTGEN MSC STAGES=(S1) ATTRIBUTES=('No of Employees');
Line 6: RUN;
```

Multiple FACTGEN Statements Can Appear in One PROC ABC Statement

Example:

```
Line 1: PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
Line 2: ATTRIBUTE 'No of Employees' ID=100;
Line 3: STAGE S1 REF='Stage1';
Line 4: DIMENSION S1 REF='GL' LEVEL=5;
Line 5: FACTGEN MSC STAGES=(S1) ATTRIBUTES=('No of Employees');
Line 7: DIMENSION S1 REF='Org' LEVEL=3;
Line 8: FACTGEN RC STAGES=(S1);
Line 9: RUN;
```

Sample Uses of the ABC Procedure

Fact Table Generation for a Multi-Stage Contributions Cube

```
PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
LOAD modelid=1017 periodid=19 scenarioid=1;
ATTRIBUTE 'Compled Expedite Requests' ID=1069;
ATTRIBUTE 'Average Time To Expedite' ID=1070;
ATTRIBUTE 'Number of Inspections' ID=1071;
```

```
STAGE 'Stage 1 Resources' ID=1047 COSTFLOW=IN;
STAGE 'Stage 2 Customer Service' ID=1048 COSTFLOW=OUT;
STAGE 'Stage 3 Parcel Handling' ID=1049 COSTFLOW=OUT;
STAGE 'Stage 4 Parcel Delivery' ID=1050 COSTFLOW=OUT;
STAGE 'Stage 5 Final' ID=1051 COSTFLOW=OUT;
DIMENSION 'Stage 1 Resources' ID=1003 REF='GL' LEVEL=1;
DIMENSION 'Stage 1 Resources' ID=1005 REF='Org' LEVEL=1;
DIMENSION 'Stage 1 Resources' ID=1007 REF='Reg' LEVEL=3;
DIMENSION 'Stage 1 Resources' ID=1009 REF='Fixed_Variable' LEVEL=1;
DIMENSION 'Stage 2 Customer Service' ID=1001 REF='Act' LEVEL=1;
DIMENSION 'Stage 2 Customer Service' ID=1005 REF='Org' LEVEL=1;
DIMENSION 'Stage 2 Customer Service' ID=1007 REF='Reg' LEVEL=3;
DIMENSION 'Stage 2 Customer Service' ID=1010 REF='Customer Value' LEVEL=1;
DIMENSION 'Stage 2 Customer Service' ID=1011 REF='Importance' LEVEL=1;
DIMENSION 'Stage 3 Parcel Handling' ID=1001 REF='Act' LEVEL=1;
DIMENSION 'Stage 3 Parcel Handling' ID=1005 REF='Org' LEVEL=1;
DIMENSION 'Stage 3 Parcel Handling' ID=1007 REF='Reg' LEVEL=3;
DIMENSION 'Stage 3 Parcel Handling' ID=1010 REF='Customer Value' LEVEL=1;
DIMENSION 'Stage 3 Parcel Handling' ID=1011 REF='Importance' LEVEL=1;
DIMENSION 'Stage 4 Parcel Delivery' ID=1001 REF='Act' LEVEL=1;
DIMENSION 'Stage 4 Parcel Delivery' ID=1005 REF='Org' LEVEL=1;
DIMENSION 'Stage 4 Parcel Delivery' ID=1007 REF='Reg' LEVEL=3;
DIMENSION 'Stage 4 Parcel Delivery' ID=1010 REF='Customer Value' LEVEL=1;
DIMENSION 'Stage 4 Parcel Delivery' ID=1011 REF='Importance' LEVEL=1;
DIMENSION 'Stage 5 Final' ID=1002 REF='Chnl' LEVEL=1;
DIMENSION 'Stage 5 Final' ID=1006 REF='Prod_Serv' LEVEL=1;
DIMENSION 'Stage 5 Final' ID=1007 REF='Reg' LEVEL=3;

FACTGEN MSC STAGES=('Stage 1 Resources' 'Stage 2 Customer Service' 'Stage 3 Parcel Handling' 'Stage 4 Parcel De
ATTRIBUTES=('Compled Expedite Requests' 'Average Time To Expedite' 'Number of Inspections')
STAGEDEF=STAGES SUPPRESSZEROCOSTS=NO NAME='M1017_C1017MSC' MSGLIMIT=50;
QUIT;
```

Fact Table Generation for a Resource Contributions Cube

```
PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
LOAD modelid=1017 periodid=19 scenarioid=1;
FACTGEN RC SUPRESSZEROCOSTS=NO NAME='M1017_ContribRes' MSGLIMIT=50';
QUIT;
```

Validate Model

```
PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
LOAD modelid=1017 periodid=19 scenarioid=1;
MODELVALIDATE OVRDRVSRCACT UASGNACT ZCOSTACT EMPTYATTR NEGDRVQTYASGN MSGLIMIT=50';
QUIT;
```

Report Table Generation

```
PROC ABC UserId='SASABMDBUSER' Password='SASABMBeatAL' CONNECTION='';
LOAD modelid=1019 periodid=3 scenarioid=1;
FACTGEN RCI FILTERS=('1' '2' '3')
```

```
FILTERTYPE=ACCOUNT NAME='M1019_ContribRes' MSGLIMIT=50;
QUIT;
```

LIST Statement

The following examples illustrate the output of a LIST statement:

```
PROC ABC DSN="myAbm"  userid="abmuser"  password="secret";
LIST;

Modelnum      ModelName(ModelID)      PeriodName(PeriodID)       ScenarioName(ScenarioID)          Calculated?

1             MyParcelExpress(1002)   2000 Q1(8)                 ACTUAL(1)                         YES

2             BellSouth(1011)         2001(2)                    BUDGET(2)                         NO

3             QAtestingbigone(1009)   2002(4)                    PLAN(3)                           NO

4             Cargill Model(1010)     2002(4)                    ACTUAL(1)                         YES
```

LIST followed by LOAD and CALCULATE

The following example illustrates the output of a LIST statement that is followed by a LOAD statement that uses the output of the LIST statement, and by a CALCULATE statement.

```
PROC ABC DSN="newAbm"  userid="abmuser"  password="secret";
LIST;

Modelnum      ModelName(ModelID)      PeriodName(PeriodID)       ScenarioName(ScenarioID)          Calculated?

1             MyParcelExpress(1002)   2000 Q1(8)                 ACTUAL(1)                         YES

2             BellSouth(1011)         2001(2)                    BUDGET(2)                         NO

3             QAtestingbigone(1009)   2002(4)                    PLAN(3)                           NO


LOAD  model="BellSouth" period="2001" scenario="BUDGET";
CALCULATE ;

Calculation on the model is run and results displayed.

LOAD  modelnum=3;
CALCULATE  DRIVERSEQLIMIT=2;

Calculation on the model is run and output is displayed.
```

LIST Followed by LOAD and QUERY

The following example illustrates the output of a LIST statement that is followed by a LOAD statement that uses the output of the LIST statement, and by a QUERY statement.

```
PROC ABC DSN="myAbm"  userid="abmuser"  password="secret";
LIST;

Modelnum      ModelName(ModelID)     PeriodName(PeriodID)       ScenarioName(ScenarioID)          Calculated?

1             MyParcelExpress(1002)  2000 Q1(8)                 ACTUAL(1)                         YES

2             BellSouth(1011)        2001(2)                    BUDGET(2)                         NO

3             QAtestingbigone(1009)  2002(4)                    PLAN(3)                           NO

4             Cargill Model(1010)    2002(4)                    ACTUAL(1)                         YES


LOAD modelid=1009 periodid=4 scenarioid=3;
QUERY  "SELECT NON EMPTY HIERARCHIZE({([Tab Contas].[Level1])})
        IN Modules.[Resource] ON ROWS,  NON EMPTY
        HIERARCHIZE({([Produtos].[Level1])}) IN Modules.[Cost Object]
        ON COLUMNS FROM MSC" / OUT=ABMOUTPUT.CARGIL1;
```

Query is run and results displayed.

LIST Fllowed by LOAD and QUERY, More Examples

```
Proc abc dsn="abmdsn" userid="sasabmdbuser" password="Orion123" Diag=Calc;
run;

Proc abc dsn="abmdsn" userid="sasabmdbuser" password="Orion123" Diag=Query;
run;

Proc abc dsn="abmdsn" userid="sasabmdbuser" password="Orion123";
List;
run;

Proc abc dsn="abmdsn" userid="sasabmdbuser" password="Orion123";
Load ModelID=1003 PeriodID=10 ScenarioId=1;
calc;
run;
```