SAS® Financial Management 5.3

Data Administrator's Guide

SAS® Documentation

Contents

Part 1

Data Administration

Chapter 1

Introduction to Data Administration

The Mission of Data Administration

Your main mission as a SAS Financial Management data administrator is to supply data to the SAS solution software. The relevant data spans a variety of content categories. These categories differ in the roles that they play and the times when they are needed:

- Data that belongs to certain content categories must be supplied initially in order to get the software working.
- Data that belongs to other content categories must be supplied periodically so that the software can produce timely output.
- Data that belongs to still other content categories might not be needed at all, depending on which solutions you are running and how you are using them.

Each content category involves unique considerations. However, there are some general themes:

- The ultimate destination of all the data that you supply is MySQL tables to which the solution software has access—the Data Mart.
- The main way of moving data from one table to another is by running SAS Data Integration Studio jobs.
- For many content categories, the data travels from its source to the MySQL tables through a set of intermediate SAS tables—the staging tables. Where this is the case, the journey of the data has two main phases:
  1. Run a custom, site-specific job that extracts the data from its source and loads it into the staging table that is designed to hold it.

2. Run a SAS Financial Management job or an equivalent wizard in the solution software to move the data from its staging area table to its ultimate destination in a MySQL table.

New Data Administration Features in SAS Financial Management

The following data administration features are new as of the 5.3 version of SAS Financial Management and the solutions.

- The detailed data store (DDS) was removed. Now data is moved directly from the staging tables to the Data Mart.
- A new chapter named "Loading Supplemental Schedule Detail and Fact Tables" was added. A supplemental schedule is an additional table that can be added to enable users to reference detailed information outside the financial model for use in reports, forms, and so on.
- A section named "Exporting Cell Protection Rules" was added to the chapter named "Loading Cell Protection Rules for a Model."
- Job cind_dds_102400_load_currency_table was renamed solnsvc_0210_load_stagefm_currency_table.
- Job cind_dds_create_new_dimension_type was renamed solnsvc_0100_create_a_new_dimension_type.
- Job cind_dds_100400_load_dimension_type_table was renamed solnsvc_0200_load_stagefm_dimension_type_table.
- The ENUM_VALUES, MIN_VALUE, MAX_VALUE, and PROPERTY_OPTIONS columns were added to table app_property to allow validation for member property values.
- The chapters named "Loading Measures" and "Loading Metrics" were removed.
- All the SAS Financial Management jobs were moved to the **Products** ⇨ **SAS Financial Management** folder. The jobs that are provided with the release are identified by the release number, such as **5.3 Jobs**. When the product is migrated to the next release, a new folder is included with the release name and its jobs.

Survey of Tasks

Preparatory Tasks for All Sites

The set of relevant data administration tasks depends on your site-specific circumstances. At any site, perform the following tasks in the specified order before you begin to load data:

1. Complete the installation of SAS Financial Management.
2. Prepare the SAS Data Integration Studio environment, following the instructions in Chapter 2, "Setting Up the SAS Data Integration Studio Environment," on page 9.

Tasks to Consider for Any Site

The following is a summary of the most important cross-solution tasks that are not specific to a single solution:

- To support any solution, you must load user and user group data into the Data Mart. You must also take steps to ensure that the user and group data in the Data Mart always matches the user and group data in the metadata repository. For details, see Chapter 5, “Loading Users and User Groups,” on page 21.
- If the set of predefined dimension types is not adequate for your purposes, then you must define additional dimension types, as explained in Chapter 10, “Adding a Dimension Type,” on page 55.
- You must define dimensions, members, and hierarchies for the dimension types that you are using. To do this with SAS Data Integration Studio, see Chapter 6, “Creating a Dimension,” on page 23 and Chapter 7, “Loading Members and Hierarchies into a Dimension,” on page 29.
- At your discretion, you can widen the availability of any SAS Data Integration Studio job by converting it into a stored process. See Chapter 11, “Creating a Stored Process from a SAS Data Integration Studio Job,” on page 63.

Tasks to Support SAS Financial Management

To support the use of SAS Financial Management, consider all of the following actions:

- To adequately describe your financial accounting data, you might need to define additional dimension types, as explained in Chapter 10, “Adding a Dimension Type,” on page 55.
- For each dimension type that is used to describe financial accounting data, you must make sure that it is properly stocked with dimensions, members, and hierarchies. For details, see Chapter 6, “Creating a Dimension,” on page 23 and Chapter 7, “Loading Members and Hierarchies into a Dimension,” on page 29.

  The following dimension types must be used to describe financial accounting data:

  - ACCOUNT
  - ANALYSIS
  - CURRENCY
  - INTORG
  - TIME

  All other dimension types are also available for use in describing financial accounting data.
- You must take steps to load fresh financial accounting data on a periodic basis. See Chapter 15, “Loading Base Data into a Financial Cycle,” on page 87.
- It is likely that you will want to load fresh currency exchange rates on a periodic basis. See Chapter 12, “Loading Exchange Rates into a SAS Financial Management Exchange Rate Set,” on page 65.
- You might want to load a system filter that users of the SAS Financial Management Add-in for Microsoft Excel can apply to read-only tables and data-entry tables. See

Chapter 18, “Loading a System Filter for the SAS Financial Management Add-In for Microsoft Excel,” on page 111.

- You might want to load **Users** tab user-member associations, which control Write access to planning forms). See “Users Tab Data” on page 41.
- You might want to load **Security** tab user-member and group-member associations. These associations control Read access to reports in the SAS Financial Management Add-in for Microsoft Excel. See “Security Tab Data” on page 42.
- If you are managing two related SAS Financial Management systems (a development system and a production system, for example), then you might want to promote dimension members and hierarchies from one system to another. See Chapter 9, “Exporting and Promoting Members and Hierarchies,” on page 49.

Server Configuration

There are three types of servers that you might work with:

- The metadata server is the server machine on which the SAS Metadata Server software is running. SAS must be available on this same machine.
- The data-tier server is the server machine on which SAS runs data-handling programs (including the logical servers for Workspace and Stored Process servers). Transformations, error tables, and jobs are installed on the data-tier server.

  The same machine is often used as both the data-tier server and the metadata server.
- The middle–tier server is the server machine on which the managed servers and SAS Remote Services run. Certain activities require you to start or stop the managed servers and SAS Remote Services, as explained in *SAS Financial Management: System Administration Guide*.

Documentation Conventions

This book uses the following documentation conventions to identify paths in the solutions configuration:

Table 1.1 *Conventions for Pathnames*

Path	Refers to	Example
`!SASROOT`	Path to the SAS root directory	Windows: `C:\Program Files\SASHome\SASFoundation\9.3` UNIX: `/usr/local/SASHome/SASFoundation/9.3`
SAS-config-dir	Path to the SAS configuration directory	Windows: `C:\SAS\Config` UNIX: `/usr/local/SAS/Config`

Path	Refers to	Example
MySQL-install-dir	Path to the MySQL installation directory	Windows: `C:\mysql` UNIX: `/usr/local/mysql`

Note:

- The name of the configuration directory and the level number might be different at your site.
- If your configuration is the result of a migration from the previous release of SAS Financial Management, then the `SASApp` directory might be called `SASMain` instead (for example, `C:\SAS\Config\Lev1\SASMain` rather than `C:\SAS\Config\Lev1\SASApp`). Please make the appropriate substitutions as you read this book.
- File system pathnames are typically shown with Windows separators ("\"); for UNIX, substitute a forward slash.

The data-tier server can be either a Windows server or a UNIX server. For a Windows server, this book assumes that the installation drive is the C drive.

This book uses the following abbreviations:

- SAS Financial Management Data Mart — Data Mart
- SAS Financial Management staging area — staging area

Related Documentation

- *SAS Financial Management: Data Model Reference* contains column-by-column descriptions of all the tables that are installed with SAS Financial Management.
- *SAS Financial Management: System Administration Guide* discusses configuration and administration tasks for SAS Financial Management and the related solutions. Topics include security, content administration, portal administration, application administration, and J2EE application server configuration.
- *SAS Financial Management Adapter for SAP: User's Guide* explains how to extract data from SAP for use in SAS Financial Management.
- *SAS Financial Management: Customization Guide.*

These books are available at the following site:

- **SAS Financial Management:** `http://support.sas.com/documentation/onlinedoc/fm`

This site is password-restricted. You can find the user name and password in the preinstallation checklist or by calling Technical Support.

Chapter 2

Setting Up the SAS Data Integration Studio Environment

Overview of Setup Tasks

This chapter describes setup tasks that you must perform after you install SAS Financial Management and before you start using SAS Data Integration Studio to load data. These setup tasks consist of establishing access settings for the data-tier server.

Access Settings for the Data-Tier Server

Protecting Data Directories

For information about operating-system protection for files and folders, see "Post-Configuration Steps" in the *SAS Financial Management: System Administration Guide*.

Giving Server Access to Users of SAS Data Integration Studio

Overview

Each user of SAS Data Integration Studio must have a user ID and password for the data-tier server.

This user must not be the unrestricted user. If you log on as the unrestricted user, then you cannot attach the libraries that are necessary to run SAS Data Integration Studio jobs.

The user must have the following rights and permissions:

- the `Log on as a batch job` right

  The recommended way to grant this right to a user is to place the user in the SAS Server Users group and grant the right to this group. For more information, see

"Windows Privileges" in the *SAS Intelligence Platform: Security Administration Guide*.

- Read, Write, and Update access to the ***SAS-config-dir*\Lev1\Data** directory and all its subdirectories.

Groups and Roles for Data Administrators

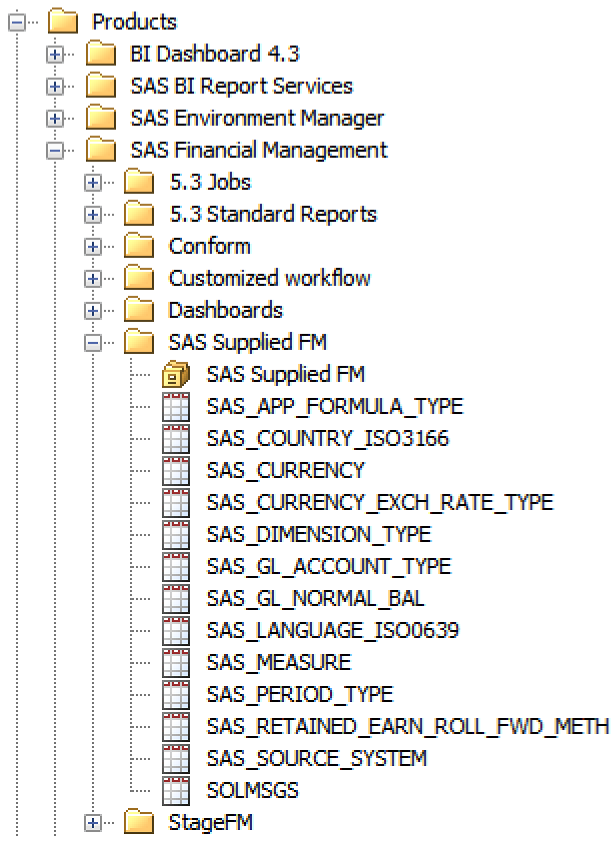
For information about group and role requirements for data administrators, see "Assigning Groups and Roles" in the *SAS Financial Management: System Administration Guide*.

Chapter 3 Using SAS Data Integration Studio to Supply Data to Solutions

Overview of the Main Data Pathway

Most data moves from its source, through the staging area to a destination Data Mart.

In general, the sources of data are transactional systems or databases that are outside the SAS environment. However, there are some source tables of predefined data that are installed with the SAS Financial Management software. The predefined source tables are the SAS_ tables in the **Products** ⇨ **SAS Financial Management** ⇨ **SAS Supplied FM** folder on the **Folders** tab of SAS Data Integration Studio.



There is one destination Data Mart, the SAS Financial Management Data Mart.

The destination Data Mart is a MySQL database. The predefined source tables and the staging area tables are all SAS tables.

The complete path that the data follows consists of the following main steps:

1. Using jobs that you write, extract data from your source systems and load it into the appropriate SAS Financial Management staging tables.

   For example, in order to supply base accounting data to SAS Financial Management, you must run a job that extracts data from your financial accounting system and loads it into the GL_TRANSACTION_SUM table.

   This step does not apply to the predefined source tables.

2. Load the data from the staging area into the appropriate Data Mart.

   You can always perform this step by running the appropriate job in SAS Data Integration Studio. For some categories of data that are used by SAS Financial Management, you can also perform this step inside SAS Financial Management Studio. For example, to load base accounting data into the Data Mart, you can do either of the following:

   - In SAS Data Integration Studio, run a job fm_1100_load_base_data or fm_1100_load_base_data_unlock_periods.
   - In the Periods workspace of SAS Financial Management Studio, run the Load New Data wizard.

Data Encodings

The MySQL databases that hold solution data must be set up at installation time to use the UTF-8 encoding.

Unless you are using double-byte SAS with a UTF-8 SAS session encoding, jobs that load data from the staging area to the Data Mart must transcode the data from the SAS session encoding to UTF-8. Conversely, jobs that export data from the Data Mart to staging tables must transcode the data from UTF-8 to the SAS session encoding.

Note: To facilitate this transcoding, your SAS session encoding must be specified at installation time.

Moving Data from Its Source to the Staging Tables

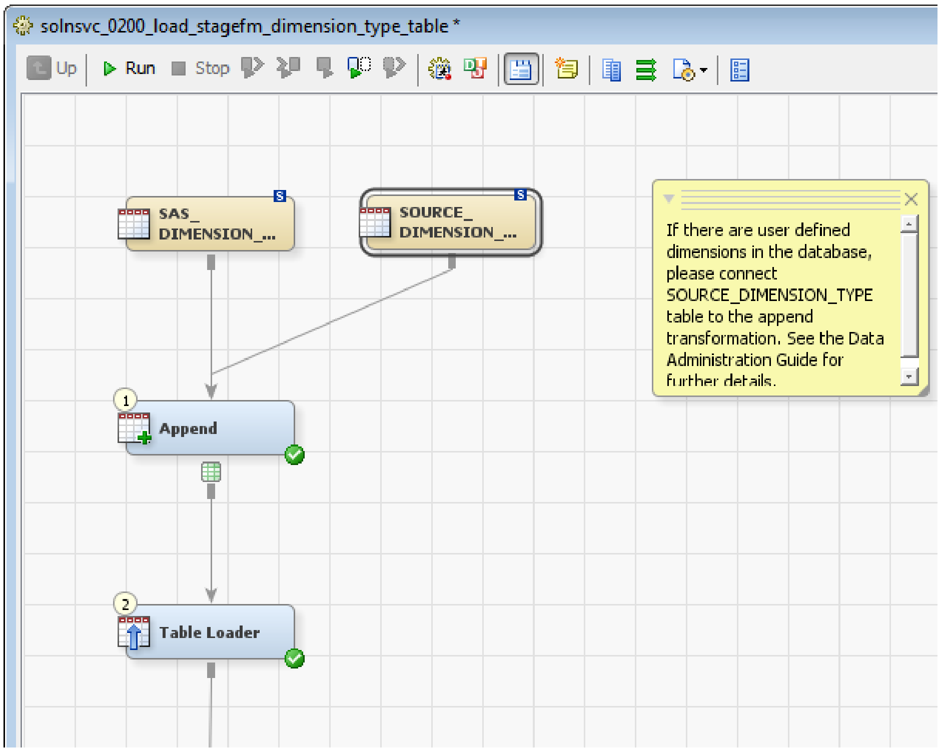
The trip from source to staging table has one form when the source is outside the SAS environment and another form when the source is a SAS_ table of predefined data:

- If the data source is outside the SAS environment, then you can load the appropriate staging table from the data source in any way that you want. For example, you can write a separate job to load each staging table or you can write jobs that load groups of related staging tables. You can run the jobs in any order. You can store the jobs in any folder. The one requirement is that your jobs must place the right data in the right columns of the right staging tables. If they achieve that result, then the jobs that load the staging area tables from the staging tables can perform the next step.

  If one of your data sources is SAP, then you can use the SAS Financial Management Adapter for SAP. See *SAS Financial Management Adapter for SAP: User's Guide*.

  If you are running SAS under 64-bit Windows and the source files are on a machine running 32-bit Windows, then you must use SAS PC Files Server to configure the data sources. For instructions, see “Post-Configuration Steps” in the *SAS Financial Management: System Administration Guide*.

- If the data source is a table that is supplied by SAS (SAS_ table) of predefined data, then the job that loads the corresponding staging area table does all the work. The job loads the data from the SAS_ source table to the corresponding staging area table. The jobs that handle predefined data begin at the SAS_ source table, as illustrated here by the job that loads dimension types:



If there are user defined dimensions in the database, please connect SOURCE_DIMENSION_TYPE table to the append transformation. See the Data Administration Guide for further details.

As this process diagram shows, these jobs enable you to supplement the predefined data with additional data from another source. You never need to write an additional job.

Before you can write a job to extract data from an external source and load it into a staging table, you must understand all the data columns in the relevant staging table. For each column, you must either determine the data source or else verify that it is appropriate to leave the column empty. For example, in order to write a job to load the GL_TRANSACTION_SUM table, you must understand all the data columns in the GL_TRANSACTION_SUM table, as explained in Chapter 15, "Loading Base Data into a Financial Cycle," on page 87.

This manual discusses the structure of some of the staging tables. The *SAS Financial Management: Data Model Reference* shows the structure of every staging area table.

In writing a job to extract data from an external source and load it into a staging table, you might be able to use the following:

- The User-Written Code transformation.
- Registering tables. Right-click a metadata folder and select **Register Tables** to register your data sources in the metadata repository, so that they show up as icons in SAS Data Integration Studio. You can then use the icons in the Process Designer.

Overview of Other Data Pathways

The following categories of data travel over pathways that do not involve the staging area:

- User and user group data travels instead through the metadata repository. It is loaded first into the metadata repository, and then into the Data Mart from the metadata repository. The jobs that load this data into the Data Mart are in the **Products** ⇨ **SAS**

Financial Management ⇨ **5.3 Jobs** folder on the **Folders** tab of SAS Data Integration Studio. There are three of these jobs:

- solnsvc_1300_load_users loads the user definitions.
- solnsvc_1400_load_groups loads the group definitions.
- solnsvc_1500_load_user_x_group loads the information about which users belong to which groups.

For details, see Chapter 5, "Loading Users and User Groups," on page 21.

- Dimensions can be created in the staging area or directly in the Data Mart. For more information, see Chapter 6, "Creating a Dimension," on page 23.

The following categories of data travel through the staging area, but give you a choice of two or more ways to load them from the staging area into the Data Mart:

- Driver rates. See Chapter 13, "Loading Driver Rates into a SAS Financial Management Driver Rate Set," on page 73.
- Members and hierarchies for an existing dimension. See Chapter 7, "Loading Members and Hierarchies into a Dimension," on page 29.
- Exchange rates. See Chapter 12, "Loading Exchange Rates into a SAS Financial Management Exchange Rate Set," on page 65.
- Base accounting data. See Chapter 15, "Loading Base Data into a Financial Cycle," on page 87.

Extending the Staging Area

You can extend the staging area in two general ways:

- Add more tables to the SAS Financial Management staging area.
- Add columns to installed tables.

In general, if you extend the staging area, the additional data cannot be loaded into a predefined Data Mart. In order to make use of the additional data, you must load it into tables in a separate location that is accessible by an appropriate application. There are two important exceptions:

- You can add custom dimension types whose members can be used to qualify financial accounting data for SAS Financial Management or metric values that can be displayed in a SAS Strategy Management scorecard. Each custom dimension type is supported by a set of four additional staging area tables and four corresponding jobs. The data in these additional tables can be loaded into the Data Mart in the same way as data for the basic dimension types. For a detailed discussion of the process of adding a dimension type, see Chapter 10, "Adding a Dimension Type," on page 55.
- You can add a column that represents a custom property to the primary member table of any dimension type. You can load the values of a custom member property into the Data Mart by following the procedure that is described in Chapter 8, "Registering Member Properties So That They Are Loaded into the SAS Financial Management Data Mart," on page 45.

If you add staging area tables that are to be used to load non-data-mart tables, be sure to do all of the following:

1. Create the staging area tables.

2. Create the non-data-mart target tables if they do not already exist.

   Note: Do not write an application that accesses staging area tables.

3. Right-click a metadata folder and select **Register Tables** to register the metadata of all the new tables, the staging area tables, and the target tables.
4. Create jobs that load the staging tables.
5. Create jobs that load the non-data-mart target tables from the staging area tables.

The result of the preceding set of steps is a data pathway to the non-data-mart target tables that is analogous to the main data pathway to the data marts. The main data pathway is described in "Overview of the Main Data Pathway" on page 11.

If you add columns to existing staging area tables that are to be used to load non-data-mart target tables, be sure to do all of the following:

1. Add the columns to the staging area tables.
2. Use **Tools** ⇨ **Update Table Metadata** to register the metadata of all the modified tables, including staging area tables.
3. Modify the jobs that load the staging tables.
4. Create the non-data-mart target tables, if they do not already exist.

   Note: Do not write an application that accesses staging area tables.

5. Right-click a metadata folder and select **Register Tables** to register the metadata of the new non-data-mart target tables.
6. Create jobs that load the non-data-mart target tables from the staging area tables.

If you add a column to a member table for the purpose of loading an additional member property into the Data Mart, do the first five of the preceding steps in order to provide for the trip into the staging area, and then provide for the trip from the staging area to the Data Mart as described in Chapter 8, "Registering Member Properties So That They Are Loaded into the SAS Financial Management Data Mart," on page 45.

Chapter 4

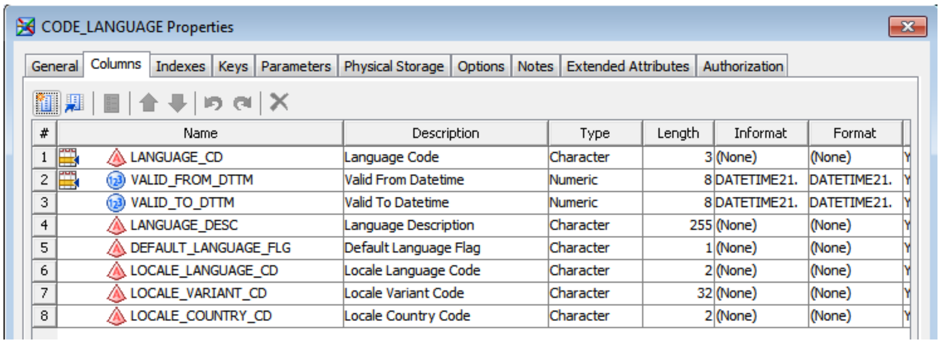
Loading Language Codes and Data Locale Codes

Overview of Languages and Data Locales

Language codes and data locale codes are used to identify the language in which associated textual data is expressed. The language codes are used in the staging tables. The data locale codes are used in the Data Mart. You can view them by selecting **Tools** ⇨ **Data Locales** in the Dimensions workspace of SAS Financial Management Studio.

Loading the Staging Table for Language and Locale Data

You must write and run a job that loads all the language codes and data locale codes that your site requires into the CODE_LANGUAGE table:



#	Name	Description	Type	Length	Informat	Format
1	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
2	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.
3	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.
4	LANGUAGE_DESC	Language Description	Character	255	(None)	(None)
5	DEFAULT_LANGUAGE_FLG	Default Language Flag	Character	1	(None)	(None)
6	LOCALE_LANGUAGE_CD	Locale Language Code	Character	2	(None)	(None)
7	LOCALE_VARIANT_CD	Locale Variant Code	Character	32	(None)	(None)
8	LOCALE_COUNTRY_CD	Locale Country Code	Character	2	(None)	(None)

Each record in this table defines a language code and a three-part data locale code. Each record also associates the language code with the data locale code. In populating this table, note the following:

- Load only those languages and data locales that are used by your data. If all your data is in a single data locale, then you need to load only one record into this table.
- Language Code is the language code that is used in staging tables.

  In general, this language code should be one of the two-character codes in the ISO0639_LANGUAGE_CD column of the SAS_LANGUAGE_ISO0639 table. You need to make an exception only if you need two or more records that represent variants of the same language. For example, if you have a record for French as used in France and another record for French as used in Canada, then you might use language codes **`frf`** and **`frc`**, respectively.

  Do not use the same language code in two records.
- Language Description is a description of the language or language variant that Language Code designates.

  For example, you might specify **`French`** or **`Canadian French`**.
- Default Language Flag must be Y for exactly one record and N for all other records. Y marks the language code for the language that is used in all the primary member tables. Be careful to coordinate the language that you mark here as the default language with the language that you use in the primary member tables. For a detailed discussion of primary and secondary member tables, see "Moving Member and Hierarchy Data from Its Source to the Staging Tables" on page 30.
- Locale Language Code and Locale Country Code work together to identify the Data Mart data locale that is associated with the staging area language code.
  - Locale Language Code must be one of the two-character codes in the ISO0639_LANGUAGE_CD column of the SAS_LANGUAGE_ISO0639 table.
  - Locale Country Code must be one of the two-character codes in the ISO3166_COUNTRY_CD column of the SAS_COUNTRY_ISO3166 table.

  In many cases, Locale Language Code can be the same two-character code as Language Code, and the other locale columns can remain empty.

  In general, the Data Mart data locale in the record that has a Default Language Flag of Y should be the data locale that is set in the Data Mart by the SAS Financial Management installation.

  Do not use the same combination of locale language code and locale country code in two records.

Data Locales with Predefined Text

The installed software includes the names and descriptions of the predefined dimension types and predefined dimensions in all the following data locales:

- da (Danish)
- de (German)
- en (English)
- es (Spanish)
- fr (French)
- it (Italian)

- ja (Japanese)
- ko (Korean)
- pl (Polish)
- ru (Russian)
- zh_CN (simplified Chinese)
- zh_TW (traditional Chinese)

If you load any of these data locales into CODE_LANGUAGE and carry them through into the Data Mart, then the associated predefined text are available in SAS Financial Management Studio.

Loading Data Locale Codes into the Data Mart

To load data locale codes from the CODE_LANGUAGE table into the Data Mart, run the solnsvc_1200_import_locales job. On the **Folders** tab, this job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder. On the **Inventory** tab, it is in the **Job** folder.

Run the job and then review the log.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Chapter 5

Loading Users and User Groups

Overview of Users and Groups

Definitions of users and groups are maintained in the metadata repository.

The SAS Financial Management installation procedure defines a number of default users, groups, and roles (see the *SAS Financial Management: System Administration Guide*). You must define all the users at your site as well as their group and role memberships. You can provide this information through a bulk-load process or interactively though SAS Management Console.

Whenever a user logs in to the solution software, the authentication process consults the user data in the metadata repository. However, there are other uses of the user data that require it to be present in the data mart. Whenever changes are made to the user data in the metadata repository, the user data in the data mart must be updated to reflect those same changes.

Ways to Load User and Group Data

The following jobs load data from the metadata repository to the data mart:

- solnsvc_1300_load_users
- solnsvc_1400_load_groups
- solnsvc_1500_load_user_x_group

These jobs are on the **Folders** tab in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder of SAS Data Integration Studio.

Best practice is to run these three jobs according to a regular schedule. For example, you might schedule a batch job to run each night. The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

The user account in which these jobs run must have Read and Write permissions to the ***`SAS-config-dir`*** **`\Lev1`** directory on the metadata server.

There is a stored process that includes all three of these jobs. For information about running the Import Users and Groups stored process, see “Assigning Groups and Roles” in the *SAS Financial Management: System Administration Guide*.

Chapter 6

Creating a Dimension

Dimension Types, Dimensions, Hierarchies, and Members

Before you perform any task that involves dimension types, dimensions, hierarchies, or members, make sure that you understand how these four concepts are related.

A dimension type represents a category of information. Examples are ACCOUNT, CURRENCY, and TIME—three of the predefined dimension types.

Each dimension type can contain many dimensions. Each dimension contains members and at least one hierarchy that is built from some or all of its members. The dimensions within a dimension type are like folders that enable you to separate the hierarchies and members into different groups.

Two dimension types—CURRENCY and ANALYSIS—can have only flat, single-level hierarchies. All other dimension types can and typically do have multi-level hierarchies. Here are two examples:

- The members of an ACCOUNT dimension are the accounts from a general ledger chart of accounts. In a typical account hierarchy, Liabilities, Current Liabilities, and Accounts Payable are on different levels, as are Assets, Current Assets, and Inventory.
- The members of a TIME dimension are time periods of different lengths. In a typical time hierarchy, years, quarters, and months are on different levels.

This chapter is about creating a new, empty dimension. You create a dimension within a dimension type, which must already exist. If you need to create a dimension type, see Chapter 10, “Adding a Dimension Type,” on page 55. After you create a dimension, you must place members and hierarchies in it. Chapter 7, “Loading Members and Hierarchies into a Dimension,” on page 29 explains how to do that.

Ways to Create a Dimension

A dimension is defined by a single dimension code, but it can have names and descriptions in any number of data locales. There are three ways to create a dimension:

- In the Dimensions workspace of SAS Financial Management Studio, select **New Dimension** and use the New Dimension wizard. Do this for each dimension that you need to create.

  If you are using several data locales, use the **Identification** tab of the dimension properties window in the Dimensions workspace of SAS Financial Management Studio to add names and descriptions in data locales other than the current data locale.

- In SAS Data Integration Studio, run the solnsvc_2200_create_dimension job or a job that you create that uses the create_dimension transformation. Do this for each dimension that you need to create.

  If you are using several data locales, use the **Identification** tab of the dimension properties window in the SAS Solutions Dimension Editor or the Dimensions workspace of SAS Financial Management Studio to add names and descriptions in data locales other than the current data locale.

- In SAS Data Integration Studio, define the dimension in the APP_DIMENSION staging table. Move the dimension definition from the staging area to the Data Mart through the standard series of steps.

  With this method, you can define any number of dimensions in any number of data locales all at once by placing all the necessary specifications in the APP_DIMENSION table.

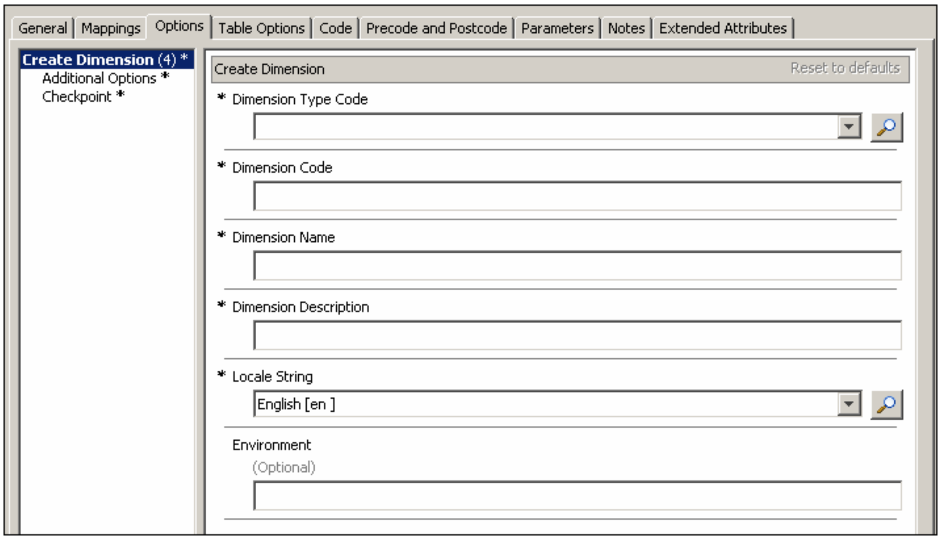
The relative convenience of these three methods can vary with the number of dimensions that you are creating and the number of data locales for each dimension.

Using the Create Dimension Transformation

Make a copy of the solnsvc_2200_create_dimension job.

Continue as follows:

1. Double-click the job to display its process diagram.
2. In the process diagram, select the Create Dimension transformation, and then select **Properties** from the pop-up menu.
3. In the Properties window, select the **Options** tab:



4. Provide values for the following options:

   - **Dimension Type Code** is the code of the dimension type within which the new dimension is created. To check the spelling of dimension type codes, use the Dimensions workspace of SAS Financial Management Studio.

   - **Dimension Code** is a unique code that identifies the new dimension. You must use this code whenever you load members and hierarchies into the dimension (as explained in "Moving Member and Hierarchy Data from the Staging Area to the Data Mart" on page 42). The code can be up to 32 characters. If you use a DBCS Unicode server and you specify more than 10 characters, then the code will be truncated.

   - **Dimension Name** identifies the new dimension in a way that users will find useful. The name can be up to 50 characters. If you use a DBCS Unicode server and you use more than 16 characters, then the name will be truncated.

     The name that you supply here are associated with the data locale that you specify with the Locale String option. After you create the dimension, you can enter names and descriptions for other data locales using the Dimensions workspace of SAS Financial Management Studio.

   - **Dimension Description** identifies the new dimension in a way that users will find useful. The description can be up to 255 characters. If you use a DBCS Unicode server and you use more than 85 characters, then the description will be truncated.

     The name and description that you supply here are associated with the data locale that you specify with the Locale String option. After you create the dimension, you can enter names and descriptions for other data locales using the Dimensions workspace of SAS Financial Management Studio.

   - **Locale String** is a string that identifies the data locale that is associated with the name and description that you specify with the **Dimension Name** and **Dimension Description** options. The string concatenates the language and country components of the data locale, using underscores as separators. Here are some examples of well-formed data locale strings:

     - `en` has only a language component.
     - `en_US` concatenates language and country components.
     - `zh_CN` concatenates language and country components.

The data locale that you specify here must already be defined in the Data Mart at the time that you run this job. For details about loading data locale codes, see Chapter 4, “Loading Language Codes and Data Locale Codes,” on page 17.

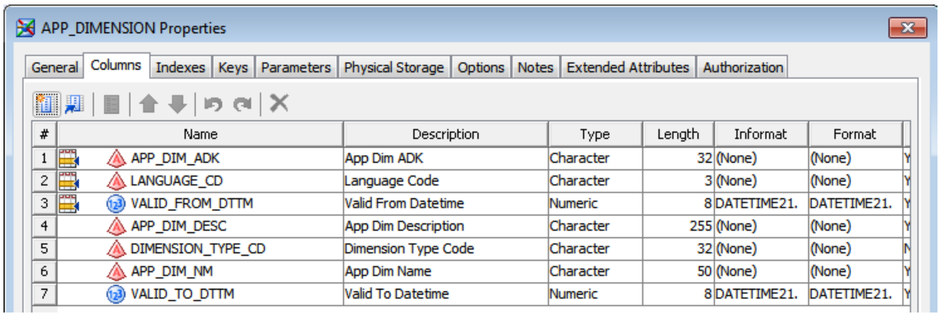
5. Run the job and then review the log.

The job can run only if SAS Remote Services and the managed servers are running on the Middle Tier Server. See *SAS Financial Management: System Administration Guide*.

Starting from a Staging Table

To create one or more dimensions using a staging table:

1. Write and run a job to load the necessary data into the APP_DIMENSION table:



APP_DIMENSION Properties

#	Name	Description	Type	Length	Informat	Format	
1	APP_DIM_ADK	App Dim ADK	Character	32	(None)	(None)	Y
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)	Y
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.	Y
4	APP_DIM_DESC	App Dim Description	Character	255	(None)	(None)	Y
5	DIMENSION_TYPE_CD	Dimension Type Code	Character	32	(None)	(None)	N
6	APP_DIM_NM	App Dim Name	Character	50	(None)	(None)	Y
7	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.	Y

On the **Folders** tab, this table is in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder.

In building records for this table, note the following:

- App Dim ADK is a unique code that will identify the new dimension. You must use this code whenever you load members and hierarchies into the dimension (as explained in “Moving Member and Hierarchy Data from the Staging Area to the Data Mart” on page 42).

- Language Code is a Language Code value that is in the CODE_LANGUAGE table. The data locale that the specified language code is associated with in the CODE_LANGUAGE table is the data locale that is associated with the name and description that you specify in the App Dim Name and App Dim Description columns.

- App Dim Name and App Dim Description identify the new dimension in a way that users will find useful.

  The name and description that you supply here is associated with the data locale that you specify indirectly with the Language Code. To specify names and descriptions in several data locales for the same dimension, create several records that have the same App Dim ADK value but different Language Code values.

- Dimension Type Code is the code of the dimension type within which the new dimension is created. To check the spelling of dimension type codes, use the Dimensions workspace of SAS Financial Management Studio.

- Valid From Datetime and Valid To Datetime define the lifespan of the record.

2. Run the solnsvc_2100_create_application_dimension job to load the dimension definitions into the Data Mart.

On the **Folders** tab, this job is located in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The data locales for which you are loading dimension names and descriptions must already be defined in the Data Mart at the time that you run this job. For details about loading data locale codes, see Chapter 4, “Loading Language Codes and Data Locale Codes,” on page 17.

3. Run the job and then review the log.

   The job can run only if SAS Remote Services and the managed servers are running on the Middle Tier Server. See *SAS Financial Management: System Administration Guide*.

Chapter 7

Loading Members and Hierarchies into a Dimension

Ways to Modify the Content of a Dimension

There are two ways to modify the members and hierarchies of a dimension, interactively or via an ETL job. Each approach has two variations, as seen in the following table.

Table 7.1 *Modifying the Content of a Dimension*

Method	Details
Interactive: editing members and hierarchies directly in the Data Mart	SAS Financial Management Studio: In the Dimensions workspace, select **Members** and use the Members window to work with members and hierarchies.

Method	Details
In SAS Data Integration Studio, using the staging area	Load the members and hierarchies into the staging area from a third-party software product or another external source.
	Load the staging area with members and hierarchies that you exported from a SAS Financial Management system. This approach is part of the content promotion facility for SAS Financial Management. You can copy the members and hierarchies of a dimension from a development system to a test system, or from a test system to a production system, by exporting them from the source system as explained in Chapter 9, "Exporting and Promoting Members and Hierarchies," on page 49 and then loading them into the target system as explained in this chapter.

This chapter discusses the process of loading members and hierarchies through the staging area. You can use this method for only one dimension per dimension type. Therefore, for any dimension type that includes more than one dimension, you must choose which dimension to supply with members and hierarchies through the staging area. All other dimensions that belong to the same dimension type must be supplied with members and hierarchies interactively, using the SAS Financial Management Studio.

If you are using the GL_TRANSACTION_SUM table to supply financial accounting data to SAS Financial Management (as described in Chapter 15, "Loading Base Data into a Financial Cycle," on page 87), then every member that is used in the GL_TRANSACTION_SUM table must belong to a dimension that you load through the staging area. This criterion can help you determine which dimensions to supply with members and hierarchies through the staging area and which dimensions to supply with members and hierarchies interactively.

For any dimension type that you are not using to describe financial accounting data in the GL_TRANSACTION_SUM table, you can choose to supply the most complex dimension with members and hierarchies through the staging area and populate the simpler dimensions interactively.

Moving Member and Hierarchy Data from Its Source to the Staging Tables

Tables for Each Dimension Type

To load members and hierarchies into a given dimension, use the set of tables for the dimension type to which that dimension belongs. You can use the staging area to load members and hierarchies into only one dimension per dimension type.

On the **Folders** tab in SAS Data Integration Studio, the staging tables that you must use to load members and hierarchies into a dimension are in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder. This folder contains a set of tables for each dimension type. For most dimension types, there are four tables. For the currency and item category dimension types, there are only three tables. The set of staging tables for a dimension type includes the following:

- The primary member table is the table that is specified in the TABLE_NM column in table DIMENSION_TYPE. For example, the primary member table for the organization dimension type is INTERNAL_ORG.

  For most dimension types, the primary member table must contain a row for each member that you are loading, with text in the staging area default language. For the currency and item category dimension types, the primary member table must contain all the member records that you are loading, regardless of language. The currency and item category primary member tables have a Language Code column. This column identifies the language used in each record. The primary member tables for other dimension types do not have a Language Code column. Therefore, all their records must use the staging area default language.

  The columns of the primary member tables differ from one dimension type to another, because the members of different dimension types are characterized by different properties. The sections of this chapter on the account dimension type, the organization dimension type, and the time dimension type include illustrations of the primary member tables for those dimension types.

  You can add other columns that represent custom properties to any primary member table, as explained in "Overview of Member Properties " on page 45.

- The secondary member table is the table that is specified in the NLS_TABLE_NM column in table DIMENSION_TYPE. For example, the secondary member table for the organization dimension type is INTERNAL_ORG_NLS.

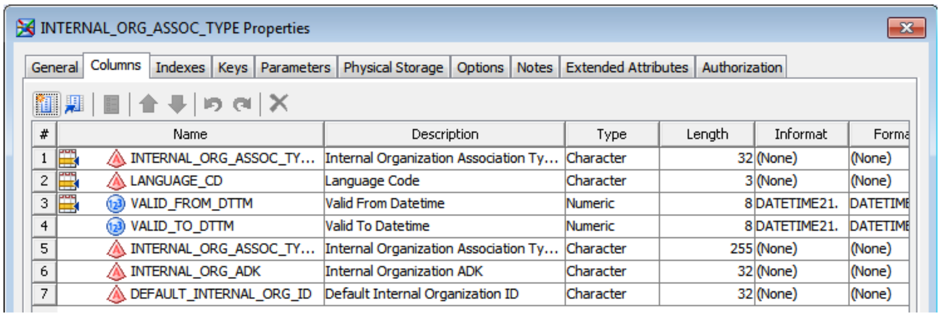
  For most dimension types, the secondary member table is the table in which to place any member records that use languages other than the default staging area language. For the currency and item category dimension types, there is no secondary member table because their primary member tables can accommodate records in all languages.

  If you are loading member records in only one language, then you can ignore the secondary member table. If you use a secondary member table, then any member that you place in it must also be in the associated primary member table.

- The hierarchy identification table is the table that is specified in the ASSOC_TYPE_TABLE_NM column in the table DIMENSION_TYPE. For example, the hierarchy identification table for the organization dimension is INTERNAL_ORG_ASSOC_TYPE.

  The hierarchy identification table must contain a row for each hierarchy that you are loading into the target dimension. If you are loading hierarchy descriptions in more than one language, then this table must contain additional rows that describe the hierarchies in the other languages.

  The hierarchy identification tables have the same column structure for all dimension types, because identifying a hierarchy involves the same considerations regardless of dimension type. Some of the column names differ from table to table to reflect the different dimension types, but the number, order, and characteristics of the columns are the same. Here are the columns of the INTERNAL_ORG_ASSOC_TYPE table:

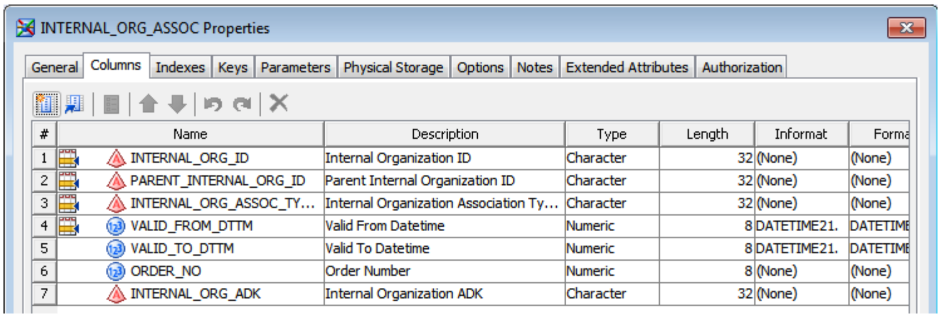


#	Name	Description	Type	Length	Informat	Forma
1	INTERNAL_ORG_ASSOC_TY…	Internal Organization Association Ty…	Character	32	(None)	(None)
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME
4	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME
5	INTERNAL_ORG_ASSOC_TY…	Internal Organization Association Ty…	Character	255	(None)	(None)
6	INTERNAL_ORG_ADK	Internal Organization ADK	Character	32	(None)	(None)
7	DEFAULT_INTERNAL_ORG_ID	Default Internal Organization ID	Character	32	(None)	(None)

- The hierarchy structure table is the table that is specified in the ASSOC_TABLE_NM column in the table DIMENSION_TYPE. For example, the hierarchy structure table for the organization dimension is INTERNAL_ORG_ASSOC.

  The hierarchy structure table must contain a row for each parent-child relationship within each hierarchy that you are loading into the target dimension. Each row of this table identifies a member, its parent member, and the hierarchy that the relationship is a part of. It also specifies the display position of the member in a fully expanded display of the hierarchy in the Dimensions workspace of SAS Financial Management Studio.

  The hierarchy structure tables have the same column structure for all dimension types because detailing a hierarchical structure involves the same considerations regardless of dimension type. Some of the column names differ from table to table to reflect the different dimension types, but the number, order, and characteristics of the columns are the same. Here are the columns of the INTERNAL_ORG_ASSOC table:



#	Name	Description	Type	Length	Informat	Forma
1	INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)	(None)
2	PARENT_INTERNAL_ORG_ID	Parent Internal Organization ID	Character	32	(None)	(None)
3	INTERNAL_ORG_ASSOC_TY…	Internal Organization Association Ty…	Character	32	(None)	(None)
4	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME
5	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME
6	ORDER_NO	Order Number	Numeric	8	(None)	(None)
7	INTERNAL_ORG_ADK	Internal Organization ADK	Character	32	(None)	(None)

In loading these tables, there are many points to keep in mind. Points that apply across all or most dimension types are discussed in "Requirements for All or Most Dimension Types" on page 32. Points that are specific to a particular dimension type are discussed in subsequent sections.

Requirements for All or Most Dimension Types

For any dimension type, the data that goes into the member tables, the hierarchy identification table, and the hierarchy structure table must meet the following conditions:

- If the primary member table has a Roll Up to Parent Flag column, then this column must have a value of either **Y** or **N**. In SAS Financial Management Studio, **Y** corresponds to selecting the **This member rolls up into its parent** check box on the **General** tab of the Properties window for a selected member. **N** corresponds to not selecting this check box. If you do not specify a value, the software provides the default value of **Y** when the data is loaded into the Data Mart.
- The hierarchy identification table must contain at least one record.

- In the hierarchy identification table, you can either specify a default member in each record or leave this column blank. If you leave the column blank, then a default member is designated automatically for each hierarchy when the hierarchies are loaded into the Data Mart. The automatically designated default member is the member in the first record of the hierarchy structure table that describes the relevant hierarchy and that makes a member its own parent.

  All the default members that you specify must also be in the primary member table. If you have several records for the same hierarchy in different languages, then either specify the same default member in all of them or leave them all blank.

- A member can be used in a hierarchy only if it is in the dimension to which the hierarchy belongs. In other words, any member that is in a parent-child record in the hierarchy structure table must also be in the primary member table.

- In the subset of the hierarchy structure table that describes a given hierarchy as of a given moment, each member that occurs as either a parent or a child must occur as a child in exactly one record:

  - If the member has a parent in that hierarchy at that moment, then that one record indicates which member is its parent.

  - If the member has no parent in that hierarchy at that moment, then that one record names the member as its own parent. This is how top-level members are identified.

- The Order Number column of the hierarchy structure table holds integers that determine the top-to-bottom display order of each parent's children in SAS Financial Management. Among each parent's children, the child with the lowest order number is displayed first, the child with the next lowest order number is displayed second, and so on.

  One approach to assigning order numbers is to assign a unique order number to every record in the table. Another approach is to start a fresh count for the children of each parent. The first approach gives the software more information than it needs, because the hierarchical structure already determines that each member will be displayed as subordinate to its parent. However, you might find that a table with unique order numbers is easier to maintain than a table that reuses the same low numbers many times.

  With either approach, it is not necessary to use consecutive integers. For example, by numbering initially with multiples of ten you can provide room to insert new members without having to renumber old members.

  If you leave this column blank in all the records of a hierarchy structure table, then the software assigns default order numbers that reflect the order of the records in the table.

  Note: The Order Number column does not affect the display of organization hierarchies in SAS Human Capital Management.

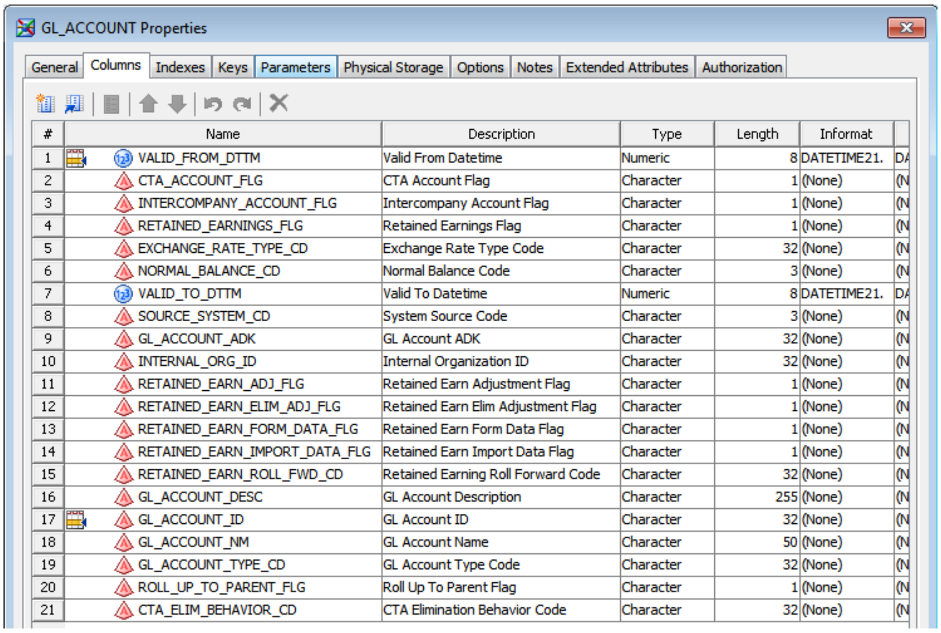
- The records of the hierarchy structure table can occur in any order, but it is a good idea to load this table so that the records are grouped by hierarchy.

- Each table includes a Valid From Datetime column and a Valid To Datetime column, which define the lifespans of its records.

Special Requirements for the Account Dimension Type

The Account Primary Member Table

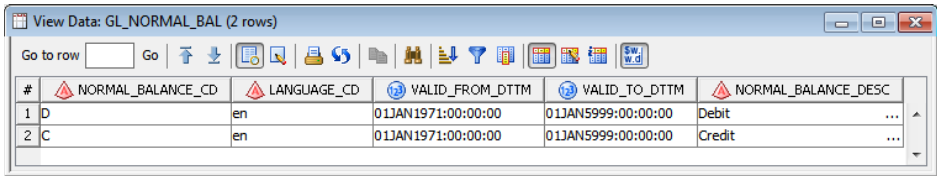
Each member of an account dimension has properties that correspond to the columns of the GL_ACCOUNT table:



#	Name	Description	Type	Length	Informat	
1	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DA
2	CTA_ACCOUNT_FLG	CTA Account Flag	Character	1	(None)	(N
3	INTERCOMPANY_ACCOUNT_FLG	Intercompany Account Flag	Character	1	(None)	(N
4	RETAINED_EARNINGS_FLG	Retained Earnings Flag	Character	1	(None)	(N
5	EXCHANGE_RATE_TYPE_CD	Exchange Rate Type Code	Character	32	(None)	(N
6	NORMAL_BALANCE_CD	Normal Balance Code	Character	3	(None)	(N
7	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DA
8	SOURCE_SYSTEM_CD	System Source Code	Character	3	(None)	(N
9	GL_ACCOUNT_ADK	GL Account ADK	Character	32	(None)	(N
10	INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)	(N
11	RETAINED_EARN_ADJ_FLG	Retained Earn Adjustment Flag	Character	1	(None)	(N
12	RETAINED_EARN_ELIM_ADJ_FLG	Retained Earn Elim Adjustment Flag	Character	1	(None)	(N
13	RETAINED_EARN_FORM_DATA_FLG	Retained Earn Form Data Flag	Character	1	(None)	(N
14	RETAINED_EARN_IMPORT_DATA_FLG	Retained Earn Import Data Flag	Character	1	(None)	(N
15	RETAINED_EARN_ROLL_FWD_CD	Retained Earning Roll Forward Code	Character	32	(None)	(N
16	GL_ACCOUNT_DESC	GL Account Description	Character	255	(None)	(N
17	GL_ACCOUNT_ID	GL Account ID	Character	32	(None)	(N
18	GL_ACCOUNT_NM	GL Account Name	Character	50	(None)	(N
19	GL_ACCOUNT_TYPE_CD	GL Account Type Code	Character	32	(None)	(N
20	ROLL_UP_TO_PARENT_FLG	Roll Up To Parent Flag	Character	1	(None)	(N
21	CTA_ELIM_BEHAVIOR_CD	CTA Elimination Behavior Code	Character	32	(None)	(N

For each record in this table, note the following:

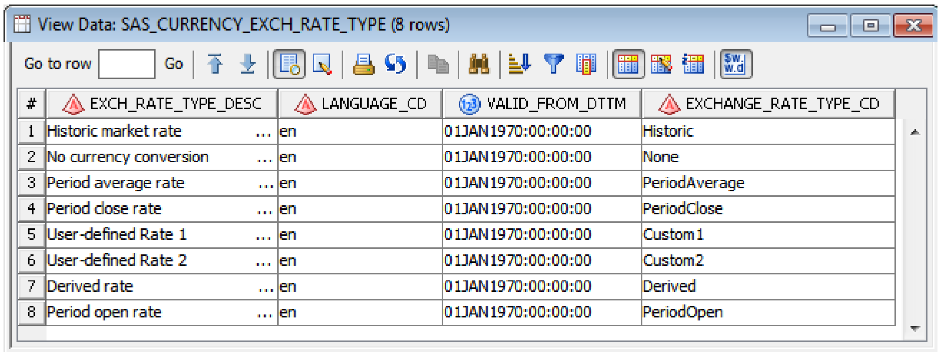
- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- Ignore these columns. They are not used:
  - CTA Account Flag
  - CTA Elimination Behavior Code
  - GL Account ADK
  - Internal Organization ID
  - Retained Earnings Flag
- Intercompany Account Flag must be Y or N. In SAS Financial Management Studio, these values correspond to selecting or not selecting the **Intercompany** check box on the **Account Details** tab of the Properties window for a selected account.
- Normal Balance Code must be C (credit) or D (debit). These values are predefined in table SAS_GL_NORMAL_BAL:



#	NORMAL_BALANCE_CD	LANGUAGE_CD	VALID_FROM_DTTM	VALID_TO_DTTM	NORMAL_BALANCE_DESC
1	D	en	01JAN1971:00:00:00	01JAN5999:00:00:00	Debit
2	C	en	01JAN1971:00:00:00	01JAN5999:00:00:00	Credit

In SAS Financial Management Studio, these values correspond to the **Credit** and **Debit** radio buttons on the **Account Details** tab of the Properties window for a selected account.

- Exchange Rate Type Code must be one of the predefined values in the EXCHANGE_RATE_TYPE_CD column of table SAS_CURRENCY_EXCH_RATE_TYPE:

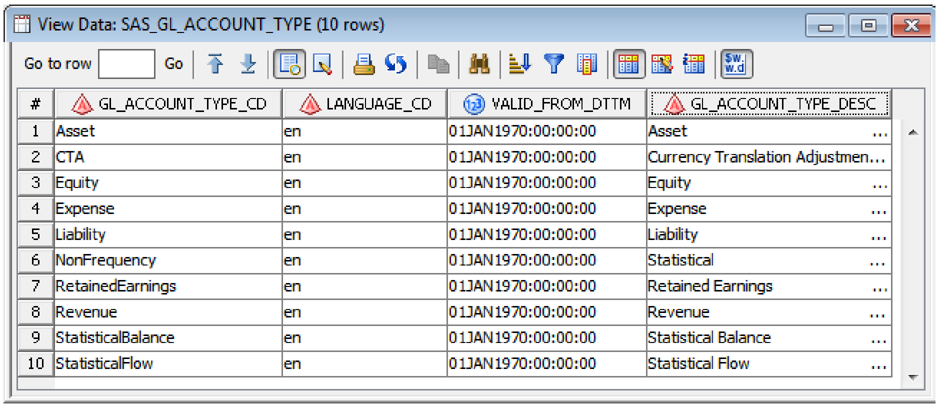


View Data: SAS_CURRENCY_EXCH_RATE_TYPE (8 rows)

#	EXCH_RATE_TYPE_DESC	LANGUAGE_CD	VALID_FROM_DTTM	EXCHANGE_RATE_TYPE_CD
1	Historic market rate …	en	01JAN1970:00:00:00	Historic
2	No currency conversion …	en	01JAN1970:00:00:00	None
3	Period average rate …	en	01JAN1970:00:00:00	PeriodAverage
4	Period close rate …	en	01JAN1970:00:00:00	PeriodClose
5	User-defined Rate 1 …	en	01JAN1970:00:00:00	Custom1
6	User-defined Rate 2 …	en	01JAN1970:00:00:00	Custom2
7	Derived rate …	en	01JAN1970:00:00:00	Derived
8	Period open rate …	en	01JAN1970:00:00:00	PeriodOpen

In SAS Financial Management Studio, these values correspond to the available values for the **Exchange rate type** field on the **Account Details** tab of the Properties window for a selected account.

- Account Type Code must be one of the predefined values in the GL_ACCOUNT_TYPE_CD column of the SAS_GL_ACCOUNT_TYPE table:

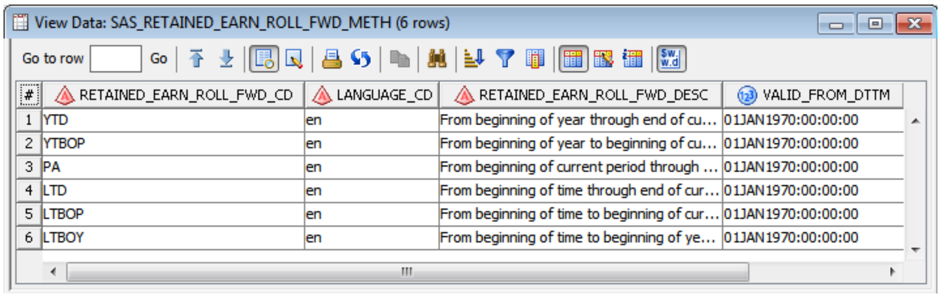


View Data: SAS_GL_ACCOUNT_TYPE (10 rows)

#	GL_ACCOUNT_TYPE_CD	LANGUAGE_CD	VALID_FROM_DTTM	GL_ACCOUNT_TYPE_DESC
1	Asset	en	01JAN1970:00:00:00	Asset …
2	CTA	en	01JAN1970:00:00:00	Currency Translation Adjustmen…
3	Equity	en	01JAN1970:00:00:00	Equity …
4	Expense	en	01JAN1970:00:00:00	Expense …
5	Liability	en	01JAN1970:00:00:00	Liability …
6	NonFrequency	en	01JAN1970:00:00:00	Statistical …
7	RetainedEarnings	en	01JAN1970:00:00:00	Retained Earnings …
8	Revenue	en	01JAN1970:00:00:00	Revenue …
9	StatisticalBalance	en	01JAN1970:00:00:00	Statistical Balance …
10	StatisticalFlow	en	01JAN1970:00:00:00	Statistical Flow …

In SAS Financial Management Studio, these values correspond to the available values for the **Account type** field on the **Account Details** tab of the Properties window for a selected account.

- If Account Type Code is CTA, then Exchange Rate Type Code must not be Historic or Derived or None.
- If Account Type Code is RetainedEarnings, then Exchange Rate Type Code must be None and Retained Earnings Roll Forward Code requires a value. The value must be one of the predefined values in the SAS_RETAINED_EARN_ROLL_FWD_METH table:


View Data: SAS_RETAINED_EARN_ROLL_FWD_METH (6 rows)

#	RETAINED_EARN_ROLL_FWD_CD	LANGUAGE_CD	RETAINED_EARN_ROLL_FWD_DESC	VALID_FROM_DTTM
1	YTD	en	From beginning of year through end of cu…	01JAN1970:00:00:00
2	YTBOP	en	From beginning of year to beginning of cu…	01JAN1970:00:00:00
3	PA	en	From beginning of current period through …	01JAN1970:00:00:00
4	LTD	en	From beginning of time through end of cur…	01JAN1970:00:00:00
5	LTBOP	en	From beginning of time to beginning of cur…	01JAN1970:00:00:00
6	LTBOY	en	From beginning of time to beginning of ye…	01JAN1970:00:00:00

In SAS Financial Management Studio, these values correspond to the available values for the **Roll-forward method** field on the **Account Details** tab of the Properties window for a selected account. This field is on the tab only for accounts of the Retained Earnings account type.

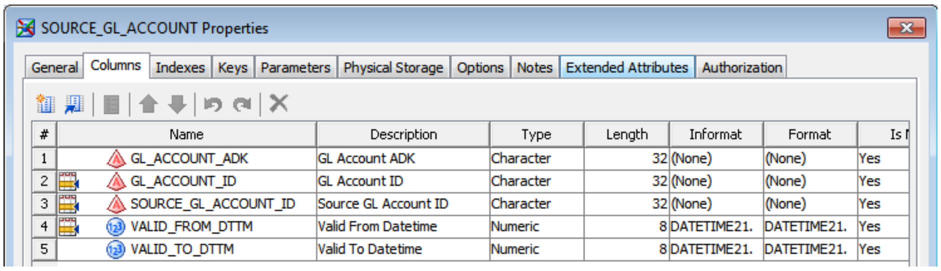
- If Account Type Code is RetainedEarnings, then the four retained earnings flag columns require values. For each of these columns, the flag value must be either Y or N. In addition, the flag value must be Y for at least one column.

  In SAS Financial Management Studio, these flag columns correspond to the four **Basis data** check boxes on the **Account Details** tab of the Properties window for a selected account. These check boxes are on the tab only for accounts of the Retained Earnings account type. The flag values determine which members of the Source hierarchy to include in the crossings that contribute to the value of the account.

The Table of RE and CTA Source Accounts

For the Account dimension type, there is a fifth staging table in addition to the standard four. The fifth staging table is named SOURCE_GL_ACCOUNT. It is located in the **Products ⇨ SAS Financial Management ⇨ StageFM** folder.

If you use the GL_ACCOUNT table to load any accounts of the Retained Earnings or CTA account types, then you must specify the source accounts for each Retained Earnings or CTA account, using the SOURCE_GL_ACCOUNT table:


SOURCE_GL_ACCOUNT Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	Format	Is
1	GL_ACCOUNT_ADK	GL Account ADK	Character	32	(None)	(None)	Yes
2	GL_ACCOUNT_ID	GL Account ID	Character	32	(None)	(None)	Yes
3	SOURCE_GL_ACCOUNT_ID	Source GL Account ID	Character	32	(None)	(None)	Yes
4	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes
5	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes

For each record in this table, note the following:

- GL Account ID must have the same value as the GL Account ID of the record in GL_ACCOUNT for which you are specifying a source account.
- Source GL Account ID must be the code of the source account that you are specifying.
- If the account type of GL Account ID is CTA, then the account type of Source GL Account ID must be one of the following:
  - Asset
  - Equity
  - Liability

- Retained Earnings

- If the account type of GL Account ID is Retained Earnings, then the account type of Source GL Account ID must be one of the following:
  - Asset
  - Equity
  - Expense
  - Liability
  - Revenue
- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- Ignore GL Account ADK. It is not used.

Account Type and Exchange Rate Type Constraints on Account Hierarchies

The following account types form a group known as *balance accounts*:

- Asset
- Equity
- Liability
- Statistical Balance

The following account types form a group known as *flow accounts*:

- Expense
- Revenue
- Statistical Flow

The following exchange rate types form a group known as *complex exchange rate types*:

- Derived (DER)
- Historic (HIS)

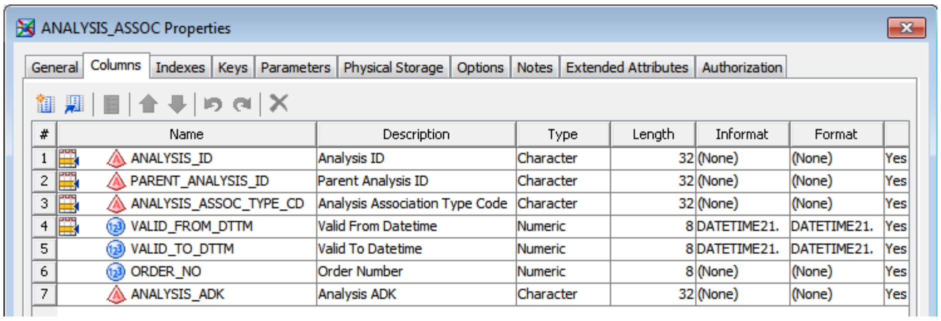
All the other exchange rate types are known as *simple exchange rate types*.

In using the GL_ACCOUNT_ASSOC table to define the parent/child relationships for an account hierarchy, you must observe the following constraints that involve account types and exchange rate types:

- The parent of a balance account, a Retained Earnings account, or a CTA account must be a balance account that uses a simple exchange rate type.
- The parent of a flow account must be a flow account that uses a simple exchange rate type.
- A child of a balance account that uses a simple exchange rate type must be either a balance account, a Retained Earnings account, or a CTA account.
- A child of a flow account that uses a simple exchange rate type must be a flow account.
- Retained Earnings accounts, CTA accounts, and all accounts that use complex exchange rate types must not have children.
- A Statistical (STA) account must have neither children nor a parent.

Special Requirements for the Analysis Dimension Type

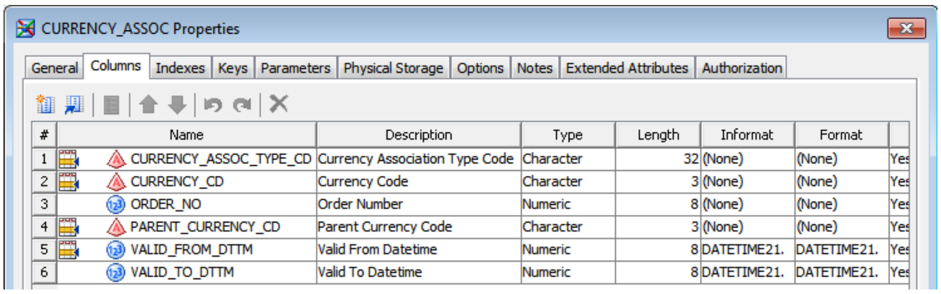
In the Analysis dimension type, every hierarchy must be flat. Every record that you load into the ANALYSIS_ASSOC hierarchy structure table must have the same analysis member code in the Analysis ID and Parent Analysis ID columns:



#	Name	Description	Type	Length	Informat	Format	
1	ANALYSIS_ID	Analysis ID	Character	32	(None)	(None)	Yes
2	PARENT_ANALYSIS_ID	Parent Analysis ID	Character	32	(None)	(None)	Yes
3	ANALYSIS_ASSOC_TYPE_CD	Analysis Association Type Code	Character	32	(None)	(None)	Yes
4	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes
5	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes
6	ORDER_NO	Order Number	Numeric	8	(None)	(None)	Yes
7	ANALYSIS_ADK	Analysis ADK	Character	32	(None)	(None)	Yes

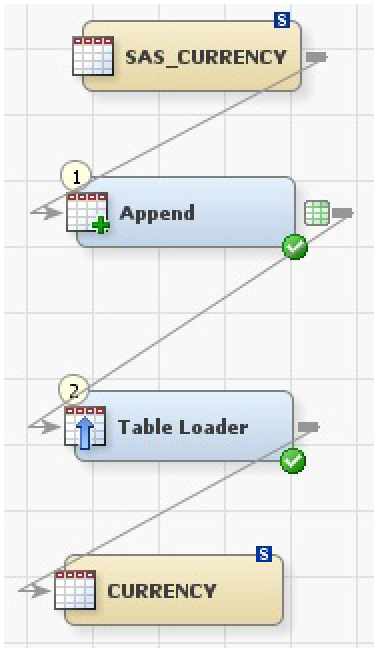
Special Requirements for the Currency Dimension Type

In the Currency dimension type, every hierarchy must be flat. Every record that you load into the CURRENCY_ASSOC hierarchy structure table must have the same currency code in the Currency Code and Parent Currency Code columns:



#	Name	Description	Type	Length	Informat	Format	
1	CURRENCY_ASSOC_TYPE_CD	Currency Association Type Code	Character	32	(None)	(None)	Yes
2	CURRENCY_CD	Currency Code	Character	3	(None)	(None)	Yes
3	ORDER_NO	Order Number	Numeric	8	(None)	(None)	Yes
4	PARENT_CURRENCY_CD	Parent Currency Code	Character	3	(None)	(None)	Yes
5	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes
6	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.	Yes

The CURRENCY table is loaded from the SAS_CURRENCY table of predefined data by the solnsvc_0210_load_stagefm_currency_table job:



This is the only dimension staging table for which you do not need to write your own job.

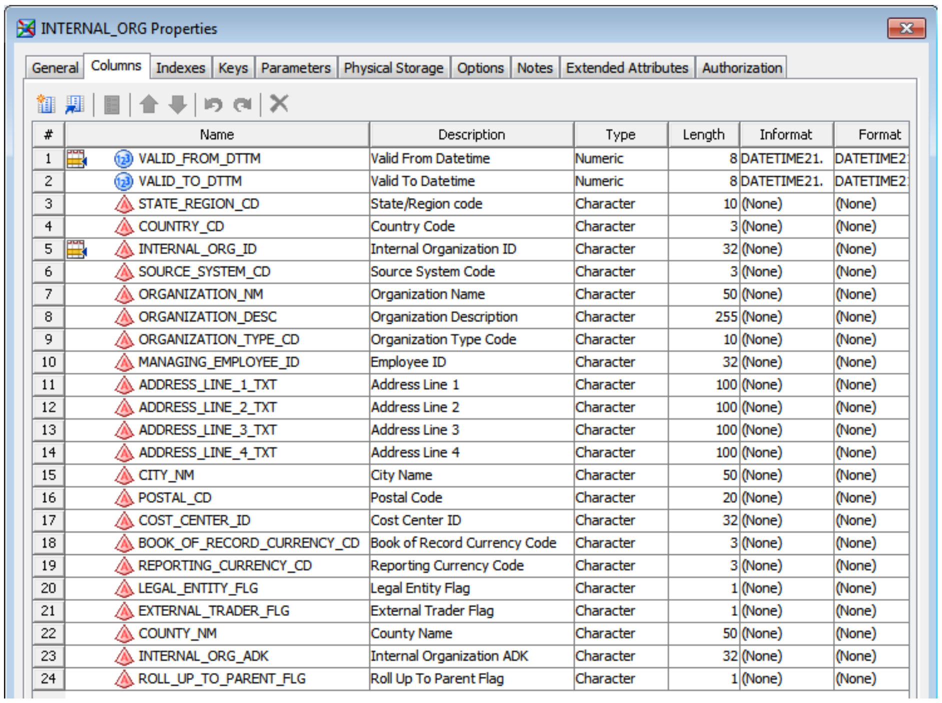
Special Requirements for the Organization Dimension Type

The INTERNAL_ORG table must contain two special members, which are not visible in the solution software. One special member is defined by an Internal Organization ID of ALL. The other special member is defined by an Internal Organization ID of EXT.

The ALL and EXT members must be part of every hierarchy that is defined in the INTERNAL_ORG_ASSOC table. In every organization hierarchy, ALL must be the unique top member, and EXT must be a leaf that is directly under ALL. The formal constraints are as follows:

- ALL must not have a parent. This is indicated by a record in which ALL is its own parent.
- ALL must be the only member of the hierarchy that does not have a parent.
- ALL must be the parent of EXT.
- EXT must not be the parent of any member.

Each member of an internal organization dimension has properties that correspond to the columns of the INTERNAL_ORG table:



#	Name	Description	Type	Length	Informat	Format
1	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME2
2	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME2
3	STATE_REGION_CD	State/Region code	Character	10	(None)	(None)
4	COUNTRY_CD	Country Code	Character	3	(None)	(None)
5	INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)	(None)
6	SOURCE_SYSTEM_CD	Source System Code	Character	3	(None)	(None)
7	ORGANIZATION_NM	Organization Name	Character	50	(None)	(None)
8	ORGANIZATION_DESC	Organization Description	Character	255	(None)	(None)
9	ORGANIZATION_TYPE_CD	Organization Type Code	Character	10	(None)	(None)
10	MANAGING_EMPLOYEE_ID	Employee ID	Character	32	(None)	(None)
11	ADDRESS_LINE_1_TXT	Address Line 1	Character	100	(None)	(None)
12	ADDRESS_LINE_2_TXT	Address Line 2	Character	100	(None)	(None)
13	ADDRESS_LINE_3_TXT	Address Line 3	Character	100	(None)	(None)
14	ADDRESS_LINE_4_TXT	Address Line 4	Character	100	(None)	(None)
15	CITY_NM	City Name	Character	50	(None)	(None)
16	POSTAL_CD	Postal Code	Character	20	(None)	(None)
17	COST_CENTER_ID	Cost Center ID	Character	32	(None)	(None)
18	BOOK_OF_RECORD_CURRENCY_CD	Book of Record Currency Code	Character	3	(None)	(None)
19	REPORTING_CURRENCY_CD	Reporting Currency Code	Character	3	(None)	(None)
20	LEGAL_ENTITY_FLG	Legal Entity Flag	Character	1	(None)	(None)
21	EXTERNAL_TRADER_FLG	External Trader Flag	Character	1	(None)	(None)
22	COUNTY_NM	County Name	Character	50	(None)	(None)
23	INTERNAL_ORG_ADK	Internal Organization ADK	Character	32	(None)	(None)
24	ROLL_UP_TO_PARENT_FLG	Roll Up To Parent Flag	Character	1	(None)	(None)

In building records for this table, note the following:

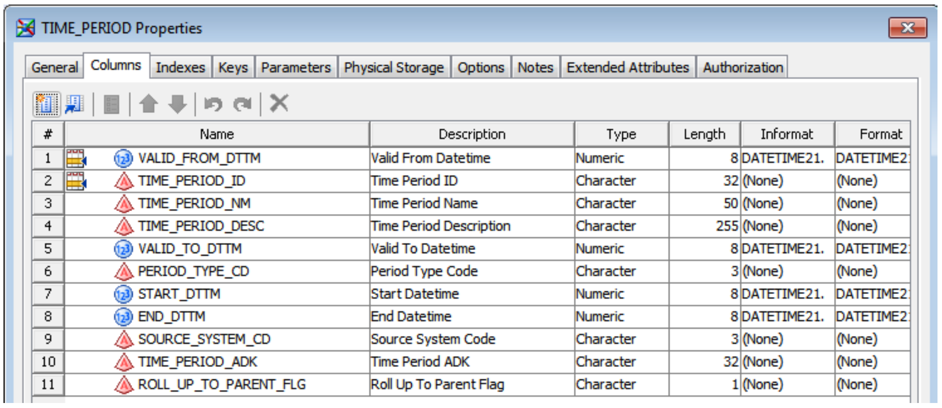
- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- Employee ID must have a value if you are using SAS Human Capital Management. Otherwise, leave this column blank. Each value that you use must be defined in the EMPLOYEE table.
- Reporting Currency Code corresponds to the Functional Currency property in SAS Financial Management Studio. You must provide a valid currency code for each organization, including ALL and EXT.

  If you are not using SAS Financial Management, then you can specify any currency code for each organization.
- Book of Record Currency Code is not used.
- Legal Entity Flag corresponds to the Reporting Entity property in SAS Financial Management Studio. Use Y for any organization that is a reporting entity and N for any organization that is not a reporting entity. For ALL and EXT, use N.

  If you are not using SAS Financial Management, then you can leave this column bank. In this case, the value N is supplied automatically for every record.
- The columns that contain geographical and address information are used by SAS Human Capital Management but not by SAS Financial Management or SAS Strategy Management. If you are not using SAS Human Capital Management, then leave these columns blank.
- Internal Organization ADK is not used.

Special Requirements for the Time Dimension Type

Each member of a time dimension has properties that correspond to the columns of the TIME_PERIOD table:



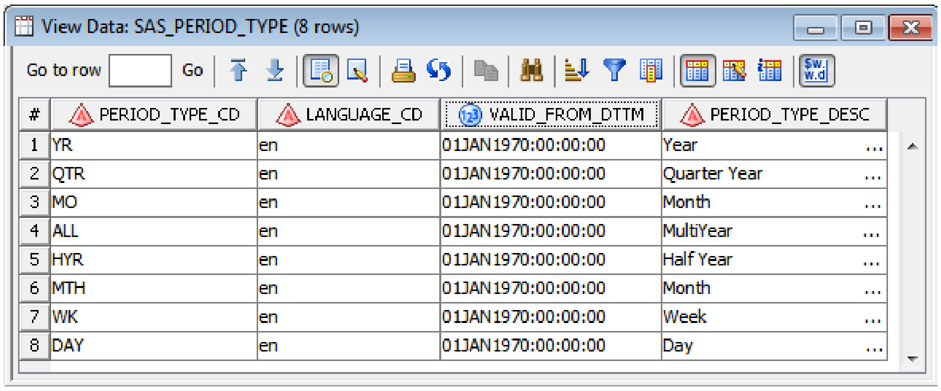
TIME_PERIOD Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	Format
1	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME2
2	TIME_PERIOD_ID	Time Period ID	Character	32	(None)	(None)
3	TIME_PERIOD_NM	Time Period Name	Character	50	(None)	(None)
4	TIME_PERIOD_DESC	Time Period Description	Character	255	(None)	(None)
5	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME2
6	PERIOD_TYPE_CD	Period Type Code	Character	3	(None)	(None)
7	START_DTTM	Start Datetime	Numeric	8	DATETIME21.	DATETIME2
8	END_DTTM	End Datetime	Numeric	8	DATETIME21.	DATETIME2
9	SOURCE_SYSTEM_CD	Source System Code	Character	3	(None)	(None)
10	TIME_PERIOD_ADK	Time Period ADK	Character	32	(None)	(None)
11	ROLL_UP_TO_PARENT_FLG	Roll Up To Parent Flag	Character	1	(None)	(None)

For each record in this table, note the following:

- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- Start Date and End Date define the time period that the member represents. You must place counts of seconds from January 1, 1960:00:00:00 in both these columns. This is so even though the solution software shows only calendar dates. Do not put counts of days from January 1, 1960 in these columns.
- Period Type Code must be one of the codes in the SAS_PERIOD_TYPE table:



View Data: SAS_PERIOD_TYPE (8 rows)

Go to row Go

#	PERIOD_TYPE_CD	LANGUAGE_CD	VALID_FROM_DTTM	PERIOD_TYPE_DESC
1	YR	en	01JAN1970:00:00:00	Year ...
2	QTR	en	01JAN1970:00:00:00	Quarter Year ...
3	MO	en	01JAN1970:00:00:00	Month ...
4	ALL	en	01JAN1970:00:00:00	MultiYear ...
5	HYR	en	01JAN1970:00:00:00	Half Year ...
6	MTH	en	01JAN1970:00:00:00	Month ...
7	WK	en	01JAN1970:00:00:00	Week ...
8	DAY	en	01JAN1970:00:00:00	Day ...

- Ignore Time Period ADK. It is not used.

Users Tab Data

For dimensions of every dimension type except analysis, currency, and time, the member Properties window in SAS Financial Management Studio includes a **Users** tab. You can load the user-member associations that can be viewed and edited with this tab.

These user-member associations can serve a useful purpose. In SAS Financial Management, a user who has a **Users** tab user-member association with a certain dimension member is authorized to enter data into any planning form that is assigned to that dimension member.

To load **Users** tab information, use the APP_USER_X_MEMBER staging table. For information about the columns of this table, see *SAS Financial Management: Data Model Reference*.

The APP_USER_X_MEMBER table is also used when you promote **Users** tab information from one SAS Financial Management system to another. For detailed information about promoting dimension content, see Chapter 9, “Exporting and Promoting Members and Hierarchies,” on page 49.

Security Tab Data

For dimensions of every dimension type, the Properties window in SAS Financial Management Studio includes a **Security** tab. You can load the security for the dimension type, hierarchies, dimension members, and custom properties for this dimension, as well as security for the dimension itself that can be viewed and edited with this tab.

These user-member and group-member associations control Read access to the data that is associated with the relevant members in SAS Financial Management reports. The online Help for the **Security** tab contains examples of the security structures that you can build. For more information, see the *SAS Financial Management: User's Guide*.

To load **Security** tab user-member associations, use the APP_USER_ACTIONS staging table. To load **Security** tab group-member associations, use the APP_GROUP_ACTIONS staging table. For information about the columns of these tables, see *SAS Financial Management: Data Model Reference*.

The APP_USER_ACTIONS and APP_GROUP_ACTIONS tables are also used when you promote **Security** tab information from one SAS Financial Management system to another. For detailed information about promoting dimension content, see Chapter 9, “Exporting and Promoting Members and Hierarchies,” on page 49.

Moving Member and Hierarchy Data from the Staging Area to the Data Mart

Overview of Moving Member and Hierarchy Data from the Staging Area to the Data Mart

You can load members and hierarchies into a dimension in the Data Mart using either one of the following:

- a SAS Data Integration Studio job that uses the Import Dimension transformation
- the Load Dimension wizard in the Dimensions workspace of SAS Financial Management Studio

Typically, you can handle your dimensions in any order. The only exception is that you must load currencies into a currency dimension before you load organizations into an organization dimension.

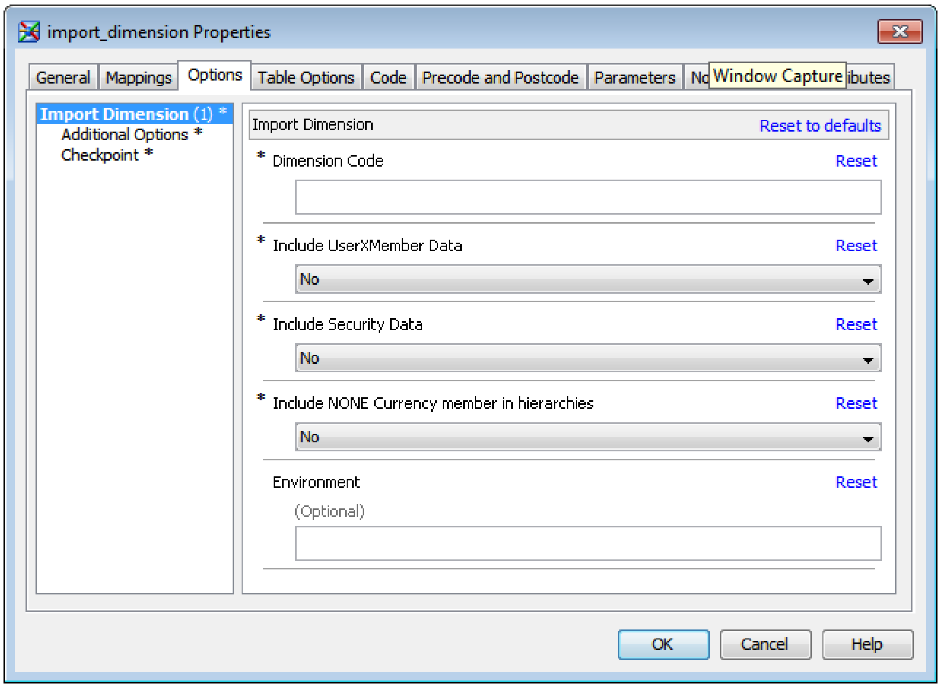
The data locales for which you are loading member and hierarchy names and descriptions must be defined in the Data Mart before you load the member and hierarchy data. For details about loading data locales, see Chapter 4, “Loading Language Codes and Data Locale Codes,” on page 17.

Using a Job

To use a SAS Data Integration Studio job, first prepare the job in the following way:

1. In the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder on the **Folders** tab, make a copy of the solnsvc_3200_load_dimension job.
2. In the process diagram, select the Import Dimension transformation, and then select **Properties** from the pop-up menu.

3. Select the **Options** tab:



* Dimension Code — Reset

* Include UserXMember Data: No — Reset

* Include Security Data: No — Reset

* Include NONE Currency member in hierarchies: No — Reset

Environment (Optional) — Reset

4. Provide values for the following options:

   - **Dimension Code** is the code of the target dimension. You can look this up in the Dimensions workspace of SAS Financial Management Studio.
   - **Include UserXMember Data** is a Yes/No flag. Select **Yes** in order to import the user-member associations that can be viewed in SAS Financial Management Studio on the **Users** tab of the Properties window. Select **No** if you are not importing this information.

     If you select **Yes**, then all information of this type for the target dimension is deleted from the Data Mart before the new information is imported.
   - **Include Security Data** is a Yes/No flag. Select **Yes** to import the security for the dimension type, hierarchies, dimension members, and custom properties for this dimension, as well as security for the dimension itself. You can view the security settings in SAS Financial Management Studio on the **Security** tab of the Properties window for each of these objects. Select **No** if you are not importing this information.

     If you select **Yes**, then all information of this type for the target dimension is deleted from the Data Mart before the new information is imported.

5. Save the job.

Run the job and then review the log. The log lists the location of an HTML report of the results.

The job can run only if SAS Remote Services and the managed servers are running on the Middle-Tier Server. See *SAS Financial Management: System Administration Guide*.

Using the Load Dimension Wizard

To use the Load Dimension wizard in the Dimensions workspace of SAS Financial Management Studio:

1. Select the target dimension from the displayed list of dimensions.
2. Select **Load Dimension** to launch the Load Dimension wizard.
3. Proceed through the Load Dimension wizard, referring to the online Help as necessary.

When the load process is complete, a window appears from which you can view an HTML report of the results.

Summary of Results

Whether you load members and hierarchies using a job or using the Load Dimension wizard, the results are the same:

- All the staging area data in the dimension-type-specific tables for the relevant dimension type is loaded. This includes the data in the primary and secondary member tables, the hierarchy identification table, and the hierarchy structure table. For an account dimension, it also includes the data in the SOURCE_GL_ACCOUNT table.
- Each member that you load replaces the existing member that has the same code, if there is one. Any existing member that is not replaced by a newly loaded member remains in the target dimension.
- For each member that you load, any associated formula data is also loaded. Associated **Security** tab data and **User** tab data is loaded only if you set the relevant flags to **Y**. Any dimension that is used in a formula must be loaded before the dimension with which the formula is associated.
- Each hierarchy that you load replaces the entire existing hierarchy that has the same code, if there is one. Any existing hierarchy that is not replaced by a newly loaded hierarchy remains in the target dimension.

Chapter 8

Registering Member Properties So That They Are Loaded into the SAS Financial Management Data Mart

Overview of Member Properties

Some of the columns in a primary member table contain information that is common across all or most dimension types. Other columns contain information that is specific to the dimension type in question. The columns that contain dimension-type-specific information represent member *properties*. Examples are Account Type for the account dimension type and Functional Currency for the organization dimension type. The generic columns represent member attributes that are not classified as member properties. Examples are Code, Name, Description, Valid from Datetime, Valid to Datetime, and Roll Up to Parent Flag.

When you load members into a dimension in the Data Mart, the information that is loaded includes all the generic columns and the values of those member properties that are registered to be loaded. Many but not all member properties are preregistered in the software. You can register more member properties, including member properties that you add to the staging area and member properties that are predefined in the staging area but not preregistered.

Member Properties That Are Preregistered

For the account and time dimension types, all predefined member properties are preregistered.

For the organization dimension type, the following predefined member properties are preregistered:

- Reporting Currency Code
- Legal Entity Flag

Defining New Member Properties

For any dimension type, you can define new member properties in the staging area by doing the following:

1. Add a column for the new property to the relevant source primary member table.
2. Modify the job that you wrote to load the source primary member table so that it loads values into the new column.

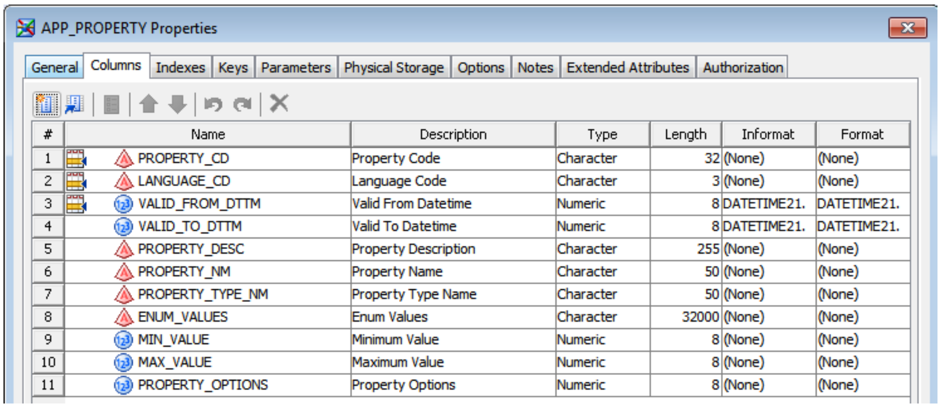
Note: After you add a column to a table, right-click the table name and select **Update Metadata**.

For example, to define a new member property for the Account dimension type, add a column for the new property to the GL_ACCOUNT table, and modify the job that loads this table.

Registering Member Properties

To register a predefined member property or a member property that you have added to the staging area, do the following:

1. In the APP_PROPERTY table, add a row that describes the property. This table is in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder on the **Folders** tab. It has the following columns:



#	Name	Description	Type	Length	Informat	Format
1	PROPERTY_CD	Property Code	Character	32	(None)	(None)
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.
4	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.
5	PROPERTY_DESC	Property Description	Character	255	(None)	(None)
6	PROPERTY_NM	Property Name	Character	50	(None)	(None)
7	PROPERTY_TYPE_NM	Property Type Name	Character	50	(None)	(None)
8	ENUM_VALUES	Enum Values	Character	32000	(None)	(None)
9	MIN_VALUE	Minimum Value	Numeric	8	(None)	(None)
10	MAX_VALUE	Maximum Value	Numeric	8	(None)	(None)
11	PROPERTY_OPTIONS	Property Options	Numeric	8	(None)	(None)

Each row that you add to this table must satisfy the following constraints:

- The Property Code column must contain the same value as the Property Code column of the corresponding record in the APP_MEMBER_PROPERTY_MAP table. Do not use any of the following reserved property codes:

```
AccountBehavior
AccountType
BalanceType
BasisData
BookCurrency
EndDate
ExchangeRateType
Formula
```

```
FormulaId
FormulaPrecedence
FormulaScope
FormulaType
FunctionalCurrency
Intercompany
Level
ReportingEntity
RollForwardMethod
SourceAccounts
StartDate
TotalAfterImport
```

- The Property Type Name column identifies the data type of the property's values. It must contain one of the following strings:

  ```
  boolean
  date
  double
  integer
  string
  ```

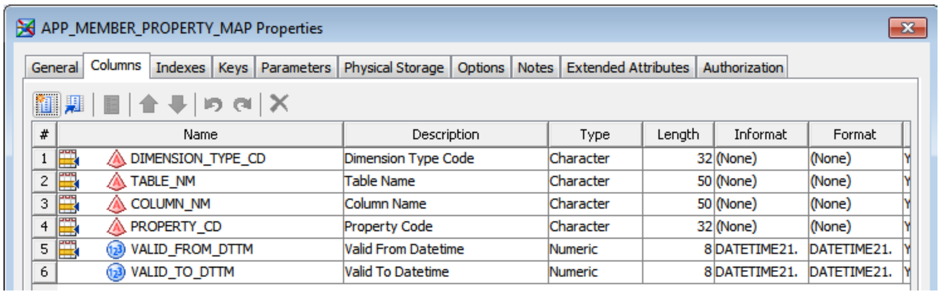
- The Enum Values column contains a string of validation values that are separated by commas (for example, red,blue,green) for string type properties.

- The Minimum Value column specifies a minimum numeric value for integer, double, or date type properties. For a date custom property, the value should be a numeric value and should have the following form: yyyymmdd.

- The Maximum Value column specifies a maximum numeric value for integer, double, or date type properties. For a date custom property, the value should be a numeric value and should have the following form: yyyymmdd.

- The Property Options specifies whether property validation active. A value of 0 specifies that property validation is inactive and a value of 1 specifies that property validation active.

2. Add to the APP_MEMBER_PROPERTY_MAP table a row that associates the property with the column that holds its values. This table is in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder. It has the following columns:

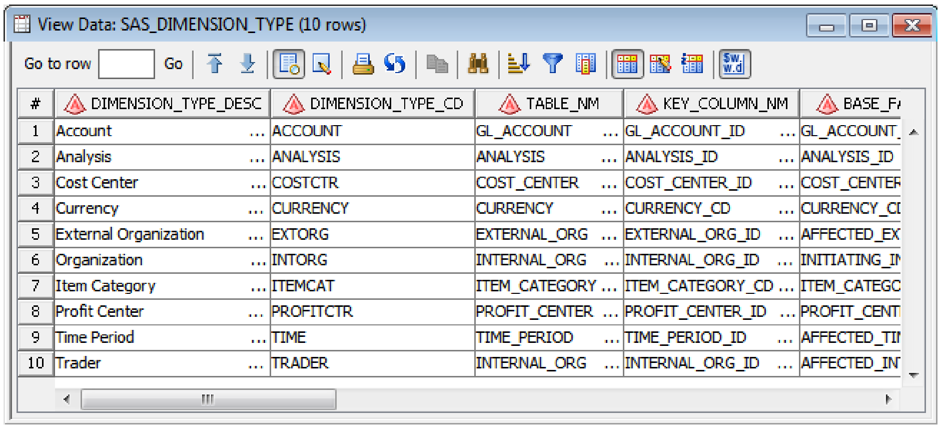


APP_MEMBER_PROPERTY_MAP Properties

#		Name	Description	Type	Length	Informat	Format	
1	✓	DIMENSION_TYPE_CD	Dimension Type Code	Character	32	(None)	(None)	Y
2	✓	TABLE_NM	Table Name	Character	50	(None)	(None)	Y
3	✓	COLUMN_NM	Column Name	Character	50	(None)	(None)	Y
4	✓	PROPERTY_CD	Property Code	Character	32	(None)	(None)	Y
5	✓	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.	Y
6		VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.	Y

Each row that you add to this table must satisfy the following constraints:

- The Dimension Type Code column must contain one of the values in the DIMENSION_TYPE_CD column of the DIMENSION_TYPE table. These values include the codes of all dimension types that you have created using the steps explained in Chapter 10, “Adding a Dimension Type,” on page 55 and the codes of all the predefined dimension types in the SAS_DIMENSION_TYPE table. Table SAS_DIMENSION_TYPE is a table that is supplied by SAS.


View Data: SAS_DIMENSION_TYPE (10 rows)

#	DIMENSION_TYPE_DESC	DIMENSION_TYPE_CD	TABLE_NM	KEY_COLUMN_NM	BASE_F
1	Account	ACCOUNT	GL_ACCOUNT	GL_ACCOUNT_ID	GL_ACCOUNT_
2	Analysis	ANALYSIS	ANALYSIS	ANALYSIS_ID	ANALYSIS_ID
3	Cost Center	COSTCTR	COST_CENTER	COST_CENTER_ID	COST_CENTER
4	Currency	CURRENCY	CURRENCY	CURRENCY_CD	CURRENCY_CI
5	External Organization	EXTORG	EXTERNAL_ORG	EXTERNAL_ORG_ID	AFFECTED_EX
6	Organization	INTORG	INTERNAL_ORG	INTERNAL_ORG_ID	INITIATING_I
7	Item Category	ITEMCAT	ITEM_CATEGORY	ITEM_CATEGORY_CD	ITEM_CATEGO
8	Profit Center	PROFITCTR	PROFIT_CENTER	PROFIT_CENTER_ID	PROFIT_CENT
9	Time Period	TIME	TIME_PERIOD	TIME_PERIOD_ID	AFFECTED_TI
10	Trader	TRADER	INTERNAL_ORG	INTERNAL_ORG_ID	AFFECTED_IN

- The Table Name column must contain the name of the primary member table for the specified dimension type. This name is in the TABLE_NM column of the DIMENSION_TYPE table.
- The Column Name column must contain the name of the column that contains the values of the property.
- The Property Code column must contain the same value as the Property Code column of the corresponding record in the APP_PROPERTY table.

Using Member Properties That You Have Registered

After additional member properties have been loaded into the Data Mart, you can view their values using either the Members view or the Hierarchies view in SAS Financial Management Studio. You can view information about the properties themselves in the Custom Properties view.

In SAS Financial Management, the following functions retrieve property values:

- the PROPERTY function in SAS Financial Management Studio
- the CDAProperty and fmProperty functions in the SAS Financial Management Add-in for Microsoft Excel

If you want the solution software to do anything else with the values of custom properties, talk to your SAS consultant about customizing the solution software.

Chapter 9

Exporting and Promoting Members and Hierarchies

Overview of Exporting Members and Hierarchies

The need to export members and hierarchies from SAS Financial Management can arise for either of two reasons.

When you export members and hierarchies, you can choose the export destination. You can also choose whether to export **Users** tab information and **Security** tab information for the exported members. Your reason for performing the export operation has implications for both choices.

Here are the two reasons and their implications:

- You have created members using the Dimensions workspace of SAS Financial Management Studio. You now want to use these members in base accounting facts to be loaded through the GL_TRANSACTION_SUM and GL_JRNL_DETAILS tables of the staging area. This requires that the members be in the appropriate stage dimension tables.

  In this case, the appropriate export destination is the staging area that serves the Data Mart that you are exporting from.

  There is no need to export **Users** tab and **Security** tab information, because this information is not used in the process of loading base accounting facts.

- You have created or modified members and hierarchies using the Dimensions workspace of SAS Financial Management Studio. You now want to promote these members and hierarchies to a test system or to a production system.

  In this case, the appropriate export destination is a Base SAS library other than the staging area that serves the Data Mart that you are exporting from. After you export the members and hierarchies to this library, you must move them to the staging area that serves the system that is your promotion target. From the staging area dimension tables for your promotion target, you load the members and hierarchies into the Data

Mart for your promotion target, as explained in "Moving Member and Hierarchy Data from the Staging Area to the Data Mart" on page 42.

Note: Before the export can occur, the tables need to exist in the export library.

Depending on how you manage **Users** tab and **Security** tab information across the two systems, you might or might not want to export **Users** tab and **Security** tab information as part of the promotion operation.

Export members and hierarchies only for those dimensions that you load through the staging area. Remember that you must choose a single dimension per dimension type to load with members and hierarchies through the staging area.

You can export the members and hierarchies of a dimension in two ways:

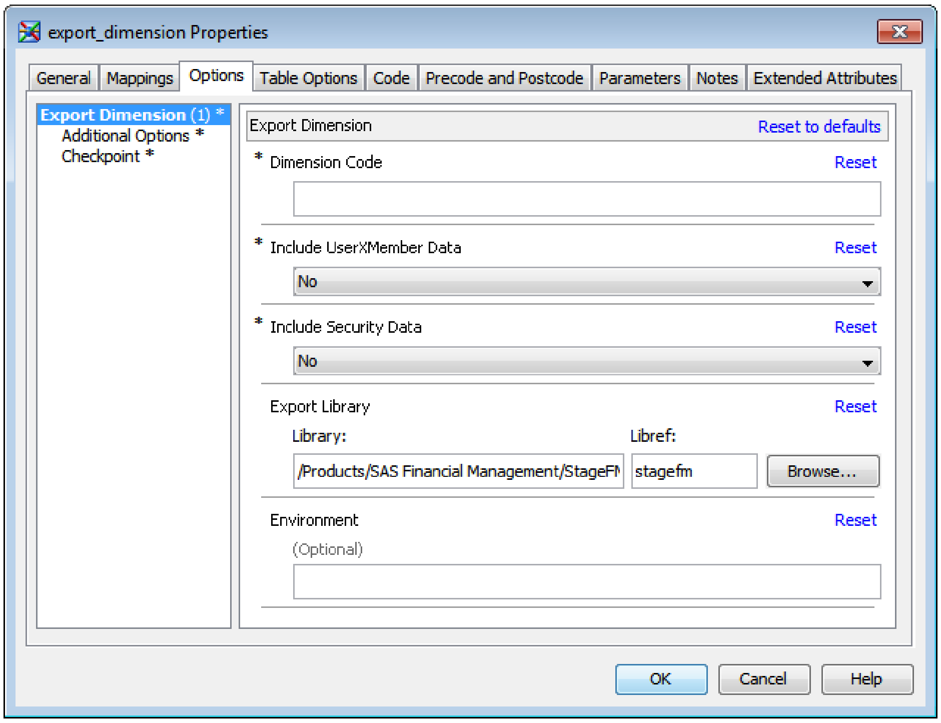
- Run a SAS Data Integration Studio job that uses the Export Dimension transformation.
- Run the Export Dimension wizard in the Dimensions workspace of SAS Financial Management Studio.

Both methods yield the same result. Both methods are available regardless of the reason for the export operation.

Using a Job to Export Members and Hierarchies

To use a SAS Data Integration Studio job, first prepare the job in the following way:

1. On the **Folders** tab, in the **Products** ⇨ **SAS Financial Managment** ⇨ **5.3 Jobs** folder, make a copy of the solnsvc_4100_export_dimension job.
2. In the process diagram, select the export_dimension transformation, and then select **Properties** from the pop-up menu. In the Properties window, select the **Options** tab:



* Dimension Code
* Include UserXMember Data: No
* Include Security Data: No
Export Library
Library: /Products/SAS Financial Management/StageFM
Libref: stagefm
Environment (Optional)

3. Provide values for the following options:

- **Dimension Code** is the code of the source dimension. You can look this up in the Dimensions workspace of SAS Financial Management Studio.
- **Include UserXMember Data** is a Yes/No flag. Select **Yes** in order to export the user-member associations that can be viewed in SAS Financial Management Studio on the **Users** tab of the member properties window. Select **No** in order to withhold these user-member associations from the exported information. **Yes** is appropriate only if you are exporting members and hierarchies in order to promote them to another system.
- **Include Security Data** is a Yes/No flag. Select **Yes** to export security for the dimension type, hierarchies, dimension members, and custom properties for this dimension, as well as security for the dimension itself. You can view the security settings in SAS Financial Management Studio on the **Security** tab of the properties window for each of these objects. Select **No** to withhold these security settings from the exported information. **Yes** is appropriate only if you are exporting members and hierarchies in order to promote them to another dimension or to another system.
- **Export Library** is the name of the Base SAS data library that you are exporting the data to. Click **Browse** to select a library. For example, select **`StageFM`** if you are exporting members and hierarchies to the staging area. If you specify a target library other than **`StageFM`**, then make sure that the target library satisfies the following conditions:
  - It is on a machine that uses the same operating system as the machine that holds the source Data Mart.
  - The Solutions Host User has operating system Read and Write access to it.
  - It contains copies of all the staging tables that are needed to receive the exported data. These include the following:
    - dimension-type-specific tables for each dimension type that you are working with. For the account dimension type, you need copies of the following five tables: GL_ACCOUNT, GL_ACCOUNT_ASSOC_TYPE, GL_ACCOUNT_ASSOC, GL_ACCOUNT_NLS, and SOURCE_GL_ACCOUNT. For most other dimension types, you need the counterparts of the first four of these tables. For the currency and item category dimension types, you need the counterparts of the first three.
    - tables that contain formula information across all dimension types that support formulas: APP_FORMULA, APP_FORMULA_TARGET, APP_FORMULA_READ_MEMBER, and APP_FORMULA_WRITE_MEMBER.
    - tables that contain **Security** tab data across all dimension types: APP_GROUP_ACTIONS and APP_USER_ACTIONS.
    - the table that contains **User** tab data across all dimension types except analysis, currency, and time (which do not support **User** tab data): APP_USER_X_MEMBER.

  Note: To define additional Base SAS libraries, use SAS Management Console.

4. Save the job.
5. Run the job and then review the log.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using the Export Dimension Wizard to Export Members and Hierarchies

To export the members and hierarchies of a selected dimension using the Export Dimension wizard:

1. In the Dimensions workspace of SAS Financial Management Studio, select the source dimension.
2. Select **Export this dimension** to launch the Export Dimension wizard.
3. Proceed through the wizard, referring to the online Help as necessary.

   If you specify an export library other than `StageFM`, then the export library must satisfy all the conditions that are listed in "Using a Job to Export Members and Hierarchies" on page 50.

Details of the Result

The two methods of exporting members and hierarchies produce the same result. Characteristics of this result include the following:

- All the data in the target dimension-type-specific tables is deleted, and then the data that you are exporting is placed in them. At the end of the process, these tables contain only the data that you have just exported.

  If the Data Mart contains member or hierarchy names and descriptions in more than one data locale, then the export includes names and descriptions in each data locale that is defined in the CODE_LANGUAGE table. For a detailed discussion of this table, see Chapter 4, "Loading Language Codes and Data Locale Codes," on page 17. The names and descriptions for the data locale that is associated with the staging area default language are exported to the primary member table. The names and descriptions for all other data locales are exported to the secondary member table. For a detailed discussion of these tables, see "Tables for Each Dimension Type" on page 30.

- All the data in the target formula tables for the dimension type that you are working with is deleted, and then the formula data that you are exporting is placed in them. At the end of the process, these tables contain only the freshly exported formula data for the dimension type that you are working with plus the previously present formula data for all other dimension types.

- If you choose to export **Security** tab data, all the data in the target **Security** tab tables for the dimension type that you are working with is deleted, and then the **Security** tab data that you are exporting is placed in them. In this case, at the end of the process, these tables contain only the freshly exported **Security** tab data for the dimension type that you are working with plus the previously present **Security** tab data for all other dimension types.

  If you choose to not export **Security** tab data, then the export operation does not change the target **Security** tab tables in any way.

- If you choose to export **User** tab data, all the data in the target **User** tab table for the dimension type that you are working with is deleted. The **User** tab data that you are exporting is placed in the table. At the end of the process, this table contains only the freshly exported **User** tab data for the dimension type that you are working with, plus the previously present **User** tab data for all other dimension types.

  If you choose to not export **User** tab data, then the export operation does not change the target **User** tab table in any way.

Possible Obstacles to Exporting a Dimension

The solnsvc_4100_export_dimension job and the Export Dimension wizard can encounter various obstacles that prevent them from successfully exporting the members and hierarchies of the selected dimension. Possible obstacles include the following:

- The Solutions Host User does not have operating system Read and Write access to the target data library.
- A target table does not exist. If the target data library is the staging area, this can happen if a table was accidentally deleted or if the staging tables for the relevant dimension type were never created. For a detailed discussion of the process of creating a dimension type, see Chapter 10, “Adding a Dimension Type,” on page 55.

  For a target data library other than the staging area, this can happen if you neglected to copy one of the necessary tables into the target library.
- A column is either misnamed or missing from a target table. This can happen if the target tables were not created correctly.
- The record for the relevant dimension type in the DIMENSION_TYPE table contains an error. This can happen if an incorrect value was placed in the record when it was created.
- One of the target tables is open and locked. This can happen if someone is working with the table.
- The CODE_LANGUAGE table does not have one and only one record marked with a Default Language Flag value of Y. For a detailed discussion of this table, see Chapter 4, “Loading Language Codes and Data Locale Codes,” on page 17.
- The CODE_LANGUAGE table does not have a record for one of the languages that is used in the member and hierarchy data that you want to export.

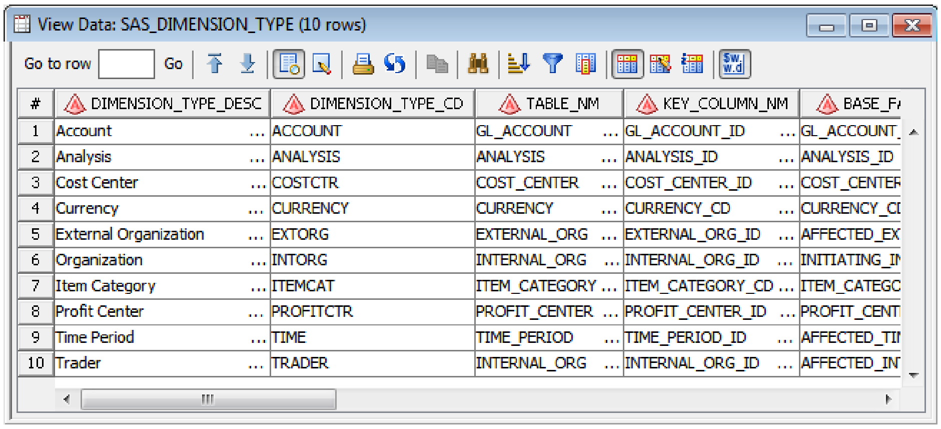
If the job or the wizard encounters any of these obstacles, an appropriate message is displayed.

Chapter 10

Adding a Dimension Type

Overview of Adding a Dimension Type

The installed SAS Financial Management software includes the dimension types that are defined in the SAS_DIMENSION_TYPE table:



View Data: SAS_DIMENSION_TYPE (10 rows)

#	DIMENSION_TYPE_DESC	DIMENSION_TYPE_CD	TABLE_NM	KEY_COLUMN_NM	BASE_F
1	Account	ACCOUNT	GL_ACCOUNT	GL_ACCOUNT_ID	GL_ACCOUNT_
2	Analysis	ANALYSIS	ANALYSIS	ANALYSIS_ID	ANALYSIS_ID
3	Cost Center	COSTCTR	COST_CENTER	COST_CENTER_ID	COST_CENTER
4	Currency	CURRENCY	CURRENCY	CURRENCY_CD	CURRENCY_C
5	External Organization	EXTORG	EXTERNAL_ORG	EXTERNAL_ORG_ID	AFFECTED_EX
6	Organization	INTORG	INTERNAL_ORG	INTERNAL_ORG_ID	INITIATING_I
7	Item Category	ITEMCAT	ITEM_CATEGORY	ITEM_CATEGORY_CD	ITEM_CATEGO
8	Profit Center	PROFITCTR	PROFIT_CENTER	PROFIT_CENTER_ID	PROFIT_CENT
9	Time Period	TIME	TIME_PERIOD	TIME_PERIOD_ID	AFFECTED_TI
10	Trader	TRADER	INTERNAL_ORG	INTERNAL_ORG_ID	AFFECTED_IN

This predefined set of dimension types might or might not meet your needs. This chapter provides instructions for adding another dimension type. You must work through the entire chapter in order, without skipping any steps. To add two or more dimension types, repeat the steps that are described here as many times as necessary.

Run the Job That Creates a New Dimension Type in the Staging Tables

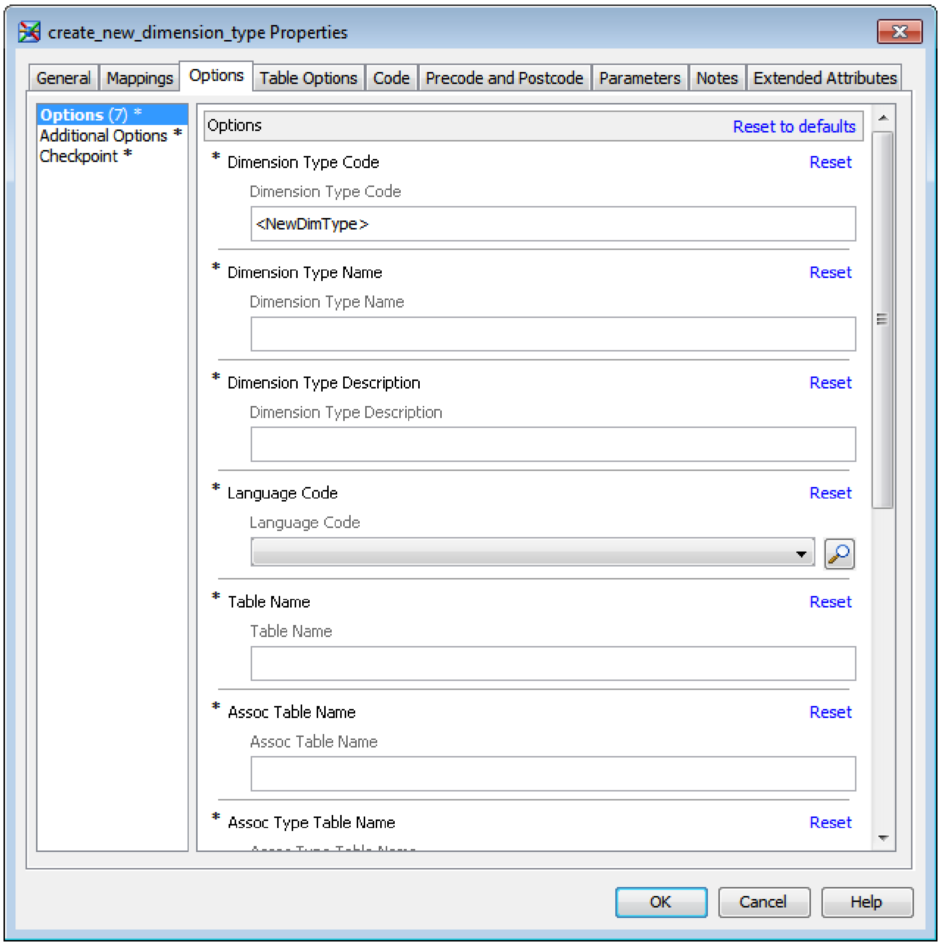
The solnsvc_0100_create_a_new_dimension_type job does the following:

- It places a row that describes a specified new dimension type in the SOURCE_DIMENSION_TYPE table. This is a supplementary source table for the solnsvc_0100_create_a_new_dimension_type job. The physical table name is SOURCE_DIMENSION_TYPE, and the metadata name is SOURCE_DIMENSION_TYPE.
- It creates the four staging tables that will be used to load members and hierarchies into a dimension that belongs to the new dimension type. For information about loading members and hierarchies into a dimension, see Chapter 7, “Loading Members and Hierarchies into a Dimension,” on page 29.
- It adds a column that holds member codes that belong to the new dimension type to the GL_TRANSACTION_SUM, GL_JRNL_DETAILS, MISC_RATE, CURRENCY_COMPLEX_EXCH_RATE, SASOP_DETAIL, and SUPP_SCHEDULE_FACT staging tables.

  This is an optional result that depends on how you set one of the job options. These tables need the additional columns only if the new dimension type is used to describe financial accounting data for SAS Financial Management.

Before you run this job, set its options as follows:

1. On the **Folders** tab, open the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.
2. Double-click the solnsvc_0100_create_a_new_dimension_type job. The process diagram for the job appears on the right.
3. In the process diagram, right-click the create_new dimension_type transformation and select **Properties** from the pop-up menu.
4. In the **Properties** window, select the **Options** tab:



* Dimension Type Code — Reset
Dimension Type Code
<NewDimType>

* Dimension Type Name — Reset
Dimension Type Name

* Dimension Type Description — Reset
Dimension Type Description

* Language Code — Reset
Language Code

* Table Name — Reset
Table Name

* Assoc Table Name — Reset
Assoc Table Name

* Assoc Type Table Name — Reset

5. On the **Options** tab, provide values for the following options:

   - **Dimension Type Code** is the code of the new dimension type. This code can be up to 32 characters long, and it can include special characters.
   - **Dimension Type Name** is the name of the new dimension type.
   - **Dimension Type Description** is the description of the new dimension type.
   - **Language Code** is one of the language codes in the CODE_LANGUAGE table. Select the appropriate language code for the dimension name and description that you have provided. For information about loading language codes, see Chapter 4, “Loading Language Codes and Data Locale Codes,” on page 17.
   - **Table Name** is the name of the primary member table for the new dimension type. To use the naming convention of the predefined dimension types, make this table name identical to the dimension type code.
   - **Assoc Table Name** is the name of the hierarchy structure table for the new dimension type. To use the naming convention of the predefined dimension types, make this table name identical to *code*_ASSOC, where *code* is the dimension type code. The table name must be 32 characters or less, and it cannot contain special characters.
   - **Assoc Type Table Name** is the name of the hierarchy identification table for the new dimension type. To use the naming convention of the predefined dimension types, make this table name identical to *code*_ASSOC_TYPE, where *code* is the dimension type code. The table name must be 32 characters or less, and it cannot contain special characters.

- **NLS Table Name** is the name of the secondary member table for the new dimension type. To use the naming convention of the predefined dimension types, make this table name identical to *code*_NLS, where *code* is the dimension type code. The table name must be 32 characters or less, and it cannot contain special characters.
- **Business ID Column Name** is the name of the column that is added to the GL_TRANSACTION_SUM, GL_JRNL_DETAILS, MISC_RATE, CURRENCY_COMPLEX_EXCH_RATE, SASOP_DETAIL, and SUPP_SCHEDULE_FACT staging tables to hold member codes that belong to the new dimension type. To use the naming convention of the predefined dimension types, make this table name identical to *code*_ID, where *code* is the dimension type code. The column name must be 32 characters or less, and it cannot contain special characters.

  If you select **No** for the **Add Dimension Type to Fact Tables** option, then leave this option blank.
- **Base Fact Column Name** is the name of the column that is added to the GL_TRANSACTION_SUM, GL_JRNL_DETAILS, MISC_RATE, CURRENCY_ COMPLEX_EXCH_RATE, SASOP_DETAIL, and SUPP_SCHEDULE_FACT tables to hold the retained keys that are generated from member codes in the staging tables. To use the naming convention of the predefined dimension types, make this table name identical to *code*_ID, where *code* is the dimension type code. The column name must be 32 characters or less, and it cannot contain special characters.

  If you select **No** for the **Add Dimension Type to Fact Tables** option, then leave this option blank.
- **Add Dimension Type to Fact Tables** is a Yes/No flag. If you select **Yes**, then a column for the new dimension type is added to the GL_TRANSACTION_SUM and GL_JRNL_DETAILS tables and their corresponding staging tables. In this case, you must specify names for these columns using the Base Fact Column Name and Business ID Column Name options. If you select **No**, then no column is added to these tables. Select **Yes** if and only if the new dimension type will be used to describe financial accounting data for use in SAS Financial Management.
- **Format/Informat for Timestamp Columns** determines the format that is used for time stamps in the four tables that will hold member and hierarchy data for the new dimension type.

6. Click **OK** to close the Properties window.
7. Select **File** ⇨ **Save**.

The following table contains a set of option values for creating a dimension type that has the code PRODUCT and that uses the naming conventions of the predefined dimensions.

Option	**Value**
Dimension Type Code	PRODUCT
Dimension Type Name	Product
Dimension Type Description	Products and product categories
Language Code	en

Option	Value
Table Name	PRODUCT
Assoc Table Name	PRODUCT_ASSOC
Assoc Type Table Name	PRODUCT_ASSOC_TYPE
NLS Table Name	PRODUCT_NLS
Business ID Column Name	PRODUCT_ID
Base Fact Column Name	PRODUCT_ID
Add Dimension Type to Fact Table	Yes

The rest of this chapter uses this set of options.

Run the job and then review the log. Select **View** ⇨ **Refresh** to refresh the metadata so that the tables for the new dimension type appear.

If you specified PRODUCT as the value of the Dimension Type Code option, then you should see the following tables in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder on the **Folders** tab:

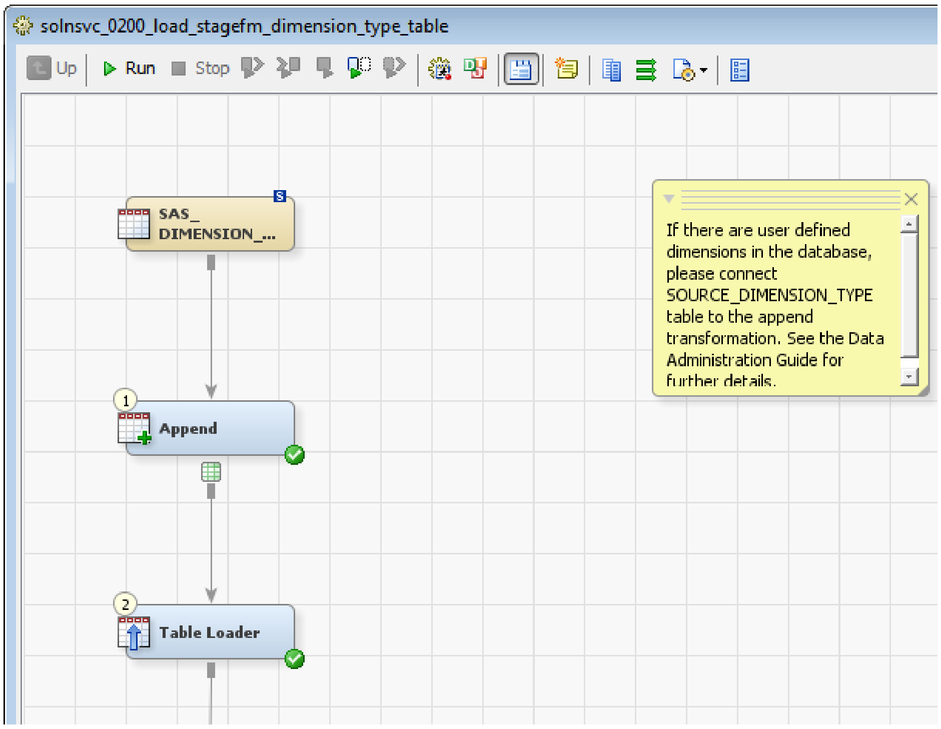
- PRODUCT
- PRODUCT_ASSOC
- PRODUCT_ASSOC_TYPE
- PRODUCT_NLS

All these tables are also visible on the **Inventory** tab, in the **Table** folder.

Run the Job That Loads a Dimension Type

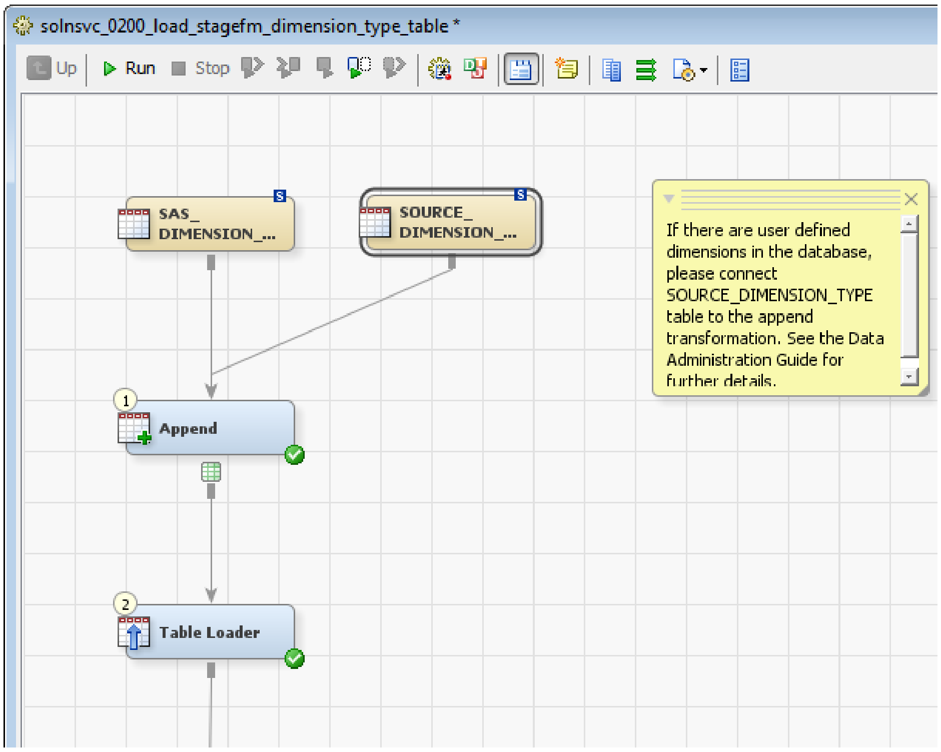
To run the solnsvc_0200_load_stagefm_dimension_type_table job:

1. In the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder on the **Folders** tab, double-click the solnsvc_0200_load_stagefm_dimension_type_table job. The process diagram appears on the right:



If there are user defined dimensions in the database, please connect SOURCE_DIMENSION_TYPE table to the append transformation. See the Data Administration Guide for further details.

2. Add the SOURCE_DIMENSION_TYPE table as a second source by dragging and dropping the table onto the process diagram and connecting it to the Append transformation:



If there are user defined dimensions in the database, please connect SOURCE_DIMENSION_TYPE table to the append transformation. See the Data Administration Guide for further details.

Make sure that the columns in the SOURCE_DIMENSION_TYPE table are mapped to the output table in the **Append** transformation.

1. In the process diagram, select the Append transformation, and then select **Properties** from the pop-up menu.

2. In the Properties window, select the **Mapping** tab.
3. Click the **Map all columns** button.
4. Click **Ok** to save your changes.

3. Run the job and then review the log. Check that all the rows of data that the solnsvc_0200_load_dimension_type_table job placed in the SOURCE_DIMENSION_TYPE table are now in the DIMENSION_TYPE table in the **Products** ⇨ **SAS Financial Management** ⇨ **StageFM** folder.

Loading New Dimension Types into the Data Mart

To load new dimension types into the Data Mart, run the solnsvc_2000_load_dimension_types job. On the **Folders** tab, this job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

Run the job and then review the log.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Creating Dimensions in a New Dimension Type

The available methods for creating a dimension are the same for new dimension types and predefined dimension types. For details, see Chapter 6, “Creating a Dimension,” on page 23.

Loading Members and Hierarchies into a Dimension That Belongs to a New Dimension Type

The procedure for loading members and hierarchies into a dimension is the same for new dimension types and predefined dimension types. For details, see Chapter 7, “Loading Members and Hierarchies into a Dimension,” on page 29.

Chapter 11
Creating a Stored Process from a SAS Data Integration Studio Job

Creating a Stored Process from a SAS Data Integration Studio Job

You can make any SAS Data Integration Studio job available as a stored process that can be run from the Document Manager. This gives a larger set of users the ability to run the code, which might be appropriate in some cases.

Before you create a stored process from a SAS Data Integration Studio job, make sure that you have made all appropriate modifications to the job. This includes specifying appropriate values for any job options.

Note: If you deploy a job as a stored process to the workspace server, then the %stpbegin and %stpend macros are commented out. If you later want to execute the resulting stored process from the portal, then you must uncomment those lines of code.

To create a stored process from a SAS Data Integration Studio job, first do the following in SAS Data Integration Studio:

1. Select the job, and then select **Open**.

   The process diagram appears in the Process Designer.

2. In the Process Designer, select the **Code** tab.

   The code for the job is displayed.

3. Select **File** ⇨ **Save to File** ⇨ **Local**.

   The Save File window appears.

4. Use the Save File window to specify the target location and name for the file that will contain the code, and then click **Save**.

   Save the file in a location such as the following: ***`SAS-config-dir`*** **`\lev1\SASApp\SASEnvironment\[Solutions Services, Financial Management, StrategyManagement]\UserDefined`**. (Create the **`UserDefined`** directory if it does not already exist.)

Next, use an appropriate editor to do the following:

1. Open the saved file.

2. At the beginning of the file, add the following statement:

   ```
   %rptinit;
   ```

3. At the end of the file, add the following statements:

   ```
   %include
   sasautos(etlstatus.sas);
   %stpend;
   ```

   The %INCLUDE statement creates a job status report, which displays the status of jobs that are executed in SAS Data Integration Studio.

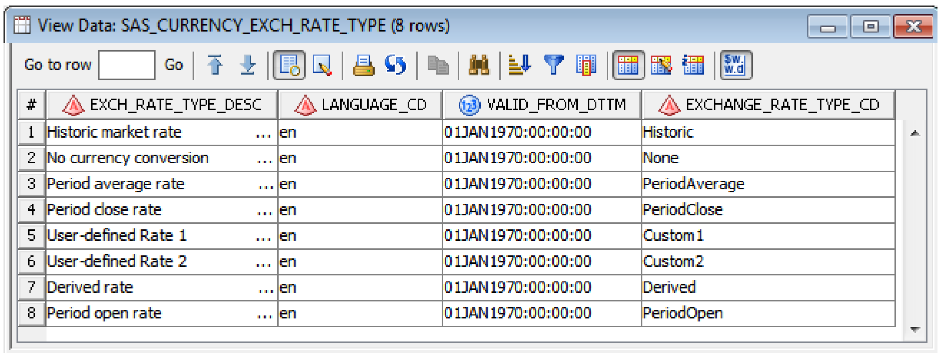
4. Save these changes and close the editor.

Register the stored process in SAS Management Console. For instructions, see “Working with Stored Processes” in the *SAS Financial Management: Customization Guide.*

Chapter 12
Loading Exchange Rates into a SAS Financial Management Exchange Rate Set

Exchange Rate Types

Every currency exchange rate belongs to one of the predefined exchange rate types in the SAS_CURRENCY_EXCH_RATE_TYPE table:


View Data: SAS_CURRENCY_EXCH_RATE_TYPE (8 rows)

#	EXCH_RATE_TYPE_DESC	LANGUAGE_CD	VALID_FROM_DTTM	EXCHANGE_RATE_TYPE_CD
1	Historic market rate ...	en	01JAN1970:00:00:00	Historic
2	No currency conversion ...	en	01JAN1970:00:00:00	None
3	Period average rate ...	en	01JAN1970:00:00:00	PeriodAverage
4	Period close rate ...	en	01JAN1970:00:00:00	PeriodClose
5	User-defined Rate 1 ...	en	01JAN1970:00:00:00	Custom1
6	User-defined Rate 2 ...	en	01JAN1970:00:00:00	Custom2
7	Derived rate ...	en	01JAN1970:00:00:00	Derived
8	Period open rate ...	en	01JAN1970:00:00:00	PeriodOpen

These exchange rate types are divided into two groups:

- All exchange rate types except Historic and Derived are known as *simple exchange rate types*. An exchange rate that belongs to a simple exchange rate type is known as a *simple exchange rate*. Simple exchange rates vary with time period but do not vary with the members of any other dimension type.
- The Historic and Derived exchange rate types are known as *complex exchange rate types*. An exchange rate that belongs to one of these complex exchange rate types is known as a *complex exchange rate*. Complex exchange rates vary with time period

and with the members of at least one other dimension type, such as account or organization.

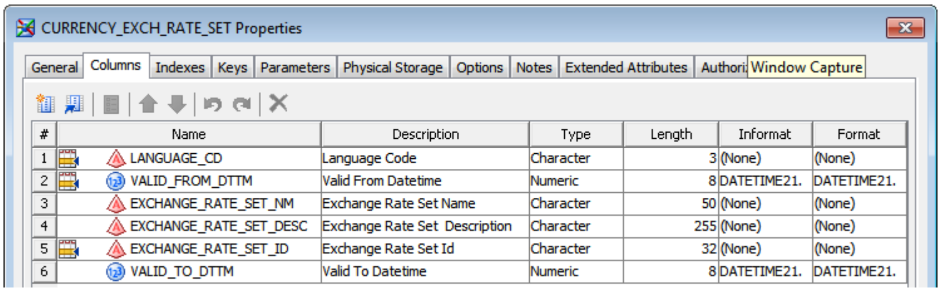
You can use SAS Data Integration Studio to load all exchange rates, both simple and complex. You can also load all exchange rates using the Load Exchange Rates wizard in the Rates workspace of SAS Financial Management Studio.

The Rates workspace of SAS Financial Management Studio also enables you to enter exchange rates manually. However, given the volume of data that is involved and the importance of avoiding errors, it is advisable to load exchange rates from a reliable source, using either SAS Data Integration Studio or the Load Exchange Rates wizard of SAS Financial Management Studio.

Exchange Rate Sets

In SAS Financial Management, every exchange rate belongs to a SAS Financial Management exchange rate set. You define SAS Financial Management exchange rate sets in the Rates workspace of SAS Financial Management Studio, where you give them codes, names, and descriptions.

In the staging area, every exchange rate belongs to a staging area exchange rate set. You define staging area exchange rate sets in the CURRENCY_EXCH_RATE_SET table:



#	Name	Description	Type	Length	Informat	Format
1	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
2	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME21.
3	EXCHANGE_RATE_SET_NM	Exchange Rate Set Name	Character	50	(None)	(None)
4	EXCHANGE_RATE_SET_DESC	Exchange Rate Set Description	Character	255	(None)	(None)
5	EXCHANGE_RATE_SET_ID	Exchange Rate Set Id	Character	32	(None)	(None)
6	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME21.

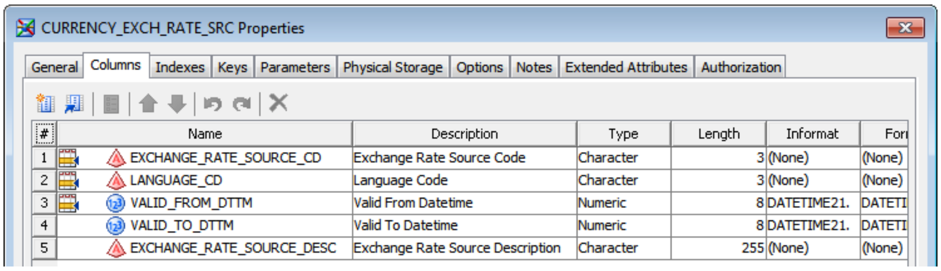
In building records for the CURRENCY_EXCH_RATE_SET table, note the following:

- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- It is advisable to maintain a one-to-one correspondence between staging area exchange rate sets and SAS Financial Management exchange rate sets, and to coordinate the codes, names, and descriptions of the corresponding pairs.

You must load the definitions of your staging area exchange rate sets into the staging area before you can load exchange rates that belong to those exchange rate sets into the staging area.

Exchange Rate Sources

Each exchange rate that you load is extracted from a certain source. You must define codes for the sources from which you extract exchange rates and load these codes into the CURRENCY_EXCH_RATE_SRC table:



CURRENCY_EXCH_RATE_SRC Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	For
1	EXCHANGE_RATE_SOURCE_CD	Exchange Rate Source Code	Character	3	(None)	(None)
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETI
4	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETI
5	EXCHANGE_RATE_SOURCE_DESC	Exchange Rate Source Description	Character	255	(None)	(None)

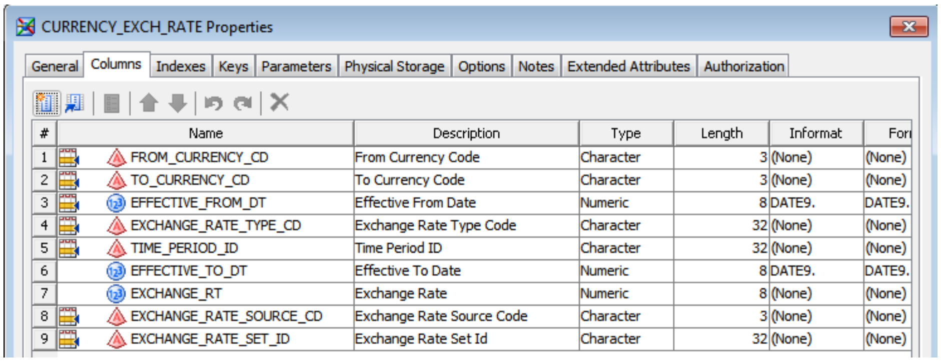
If all your exchange rates are extracted from a single source, or if you are not interested in tracking source information for exchange rates, then this table can contain a single record.

You must load the definitions of your exchange rate sources into the staging area before you can load exchange rates that belong to those exchange rate sources into the staging area.

Loading Exchange Rates into Their Staging Tables

There are two staging tables for exchange rates.

The CURRENCY_EXCH_RATE table is for simple exchange rates:

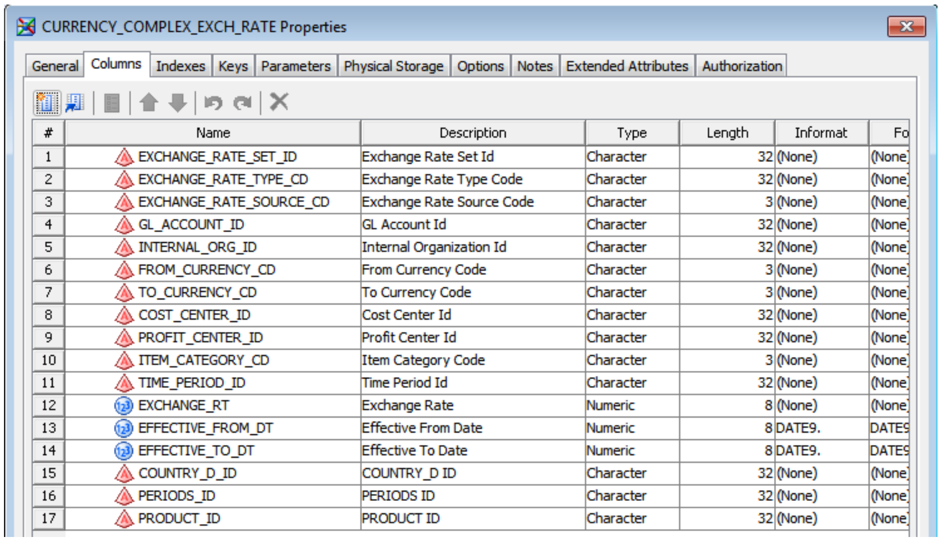


CURRENCY_EXCH_RATE Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	For
1	FROM_CURRENCY_CD	From Currency Code	Character	3	(None)	(None)
2	TO_CURRENCY_CD	To Currency Code	Character	3	(None)	(None)
3	EFFECTIVE_FROM_DT	Effective From Date	Numeric	8	DATE9.	DATE9.
4	EXCHANGE_RATE_TYPE_CD	Exchange Rate Type Code	Character	32	(None)	(None)
5	TIME_PERIOD_ID	Time Period ID	Character	32	(None)	(None)
6	EFFECTIVE_TO_DT	Effective To Date	Numeric	8	DATE9.	DATE9.
7	EXCHANGE_RT	Exchange Rate	Numeric	8	(None)	(None)
8	EXCHANGE_RATE_SOURCE_CD	Exchange Rate Source Code	Character	3	(None)	(None)
9	EXCHANGE_RATE_SET_ID	Exchange Rate Set Id	Character	32	(None)	(None)

The CURRENCY_COMPLEX_EXCH_RATE table is for complex exchange rates:



CURRENCY_COMPLEX_EXCH_RATE Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	Fo
1	EXCHANGE_RATE_SET_ID	Exchange Rate Set Id	Character	32	(None)	(None
2	EXCHANGE_RATE_TYPE_CD	Exchange Rate Type Code	Character	32	(None)	(None
3	EXCHANGE_RATE_SOURCE_CD	Exchange Rate Source Code	Character	3	(None)	(None
4	GL_ACCOUNT_ID	GL Account Id	Character	32	(None)	(None
5	INTERNAL_ORG_ID	Internal Organization Id	Character	32	(None)	(None
6	FROM_CURRENCY_CD	From Currency Code	Character	3	(None)	(None
7	TO_CURRENCY_CD	To Currency Code	Character	3	(None)	(None
8	COST_CENTER_ID	Cost Center Id	Character	32	(None)	(None
9	PROFIT_CENTER_ID	Profit Center Id	Character	32	(None)	(None
10	ITEM_CATEGORY_CD	Item Category Code	Character	3	(None)	(None
11	TIME_PERIOD_ID	Time Period Id	Character	32	(None)	(None
12	EXCHANGE_RT	Exchange Rate	Numeric	8	(None)	(None
13	EFFECTIVE_FROM_DT	Effective From Date	Numeric	8	DATE9.	DATE9
14	EFFECTIVE_TO_DT	Effective To Date	Numeric	8	DATE9.	DATE9
15	COUNTRY_D_ID	COUNTRY_D ID	Character	32	(None)	(None
16	PERIODS_ID	PERIODS ID	Character	32	(None)	(None
17	PRODUCT_ID	PRODUCT ID	Character	32	(None)	(None

For each of these tables that you need to use, write a job that extracts exchange rates from appropriate sources and loads them into the staging table.

In building records for these two exchange rate staging tables, note the following:

- To Currency Code should be the same for all records that belong to a given staging area exchange rate set, as indicated in the Exchange Rate Set ID column. When you load the data into a SAS Financial Management exchange rate set in the Data Mart, records whose To Currency Code does not match the base currency of the target exchange rate set are ignored.

- From Currency Code must be the code of the other currency that is involved in the exchange rate.

- In the CURRENCY_EXCH_RATE table, the Exchange Rate Type Code must be one of the simple exchange rate type codes in the SAS_CURRENCY_EXCH_RATE_TYPE table. In the CURRENCY_COMPLEX_EXCH_RATE table, the Exchange Rate Type Code must be one of the complex exchange rate type codes in the SAS_CURRENCY_EXCH_RATE_TYPE table. See “Exchange Rate Types” on page 65.

- Exchange Rate Set ID indicates which staging area exchange rate set the exchange rate belongs to. It must be one of the values in the CURRENCY_EXCH_RATE_SET table. See “Exchange Rate Sets” on page 66.

- Exchange Rate Source Code indicates where the exchange rate was extracted from. It must be one of the values in the CURRENCY_EXCH_RATE_SRC table. See “Exchange Rate Sources” on page 66.

- Exchange Rate is the numeric exchange rate. This must be the number by which you multiply a value expressed in the From Currency to yield the equivalent value expressed in the To Currency. For example, if the From Currency is U.S. dollars and the To Currency is Japanese yen, then the numeric exchange rate is in the neighborhood of 100, but if the From Currency is Japanese yen and the To Currency is U.S. dollars, then the numeric exchange rate is in the neighborhood of 0.01.

- Time Period ID is the code of the time period that the exchange rate applies to. The CURRENCY_COMPLEX_EXCH_RATE table has columns for member codes from other dimension types that the exchange rates depend on.

- Effective From Date is a key field that must contain a distinct date for each time period. For example, you can use the first day of each time period.

- Effective To Date is not used and should be left blank.

Loading Exchange Rates from the Staging Area to the Data Mart

Overview of Loading Exchange Rates from the Staging Area to the Data Mart

There are three ways to load exchange rates from the staging area into a SAS Financial Management exchange rate set in the Data Mart:

- Use the Load Exchange Rates wizard in SAS Financial Management Studio.
- Use a SAS Data Integration Studio job.
- Use a SAS macro.

Each time you run the SAS Data Integration Studio job, it loads exchange rates for only one combination of an exchange rate type and a time period. The Load Exchange Rates wizard and the SAS macro can handle many combinations of exchange rate types and time periods in a single run. Because there is substantial overhead associated with each run, the wizard and the SAS macro become increasingly advantageous as the number of combinations of exchange rate types and time periods increases.

When you load exchange rates into a SAS Financial Management exchange rate set by any method, all the exchange rates currently in the target exchange rate set for the specified time periods and exchange rate types are deleted before the new exchange rates are loaded.

Using the Load Exchange Rates Wizard to Load Exchange Rates into the Data Mart

To load exchange rates using the Load Exchange Rates wizard in SAS Financial Management Studio:

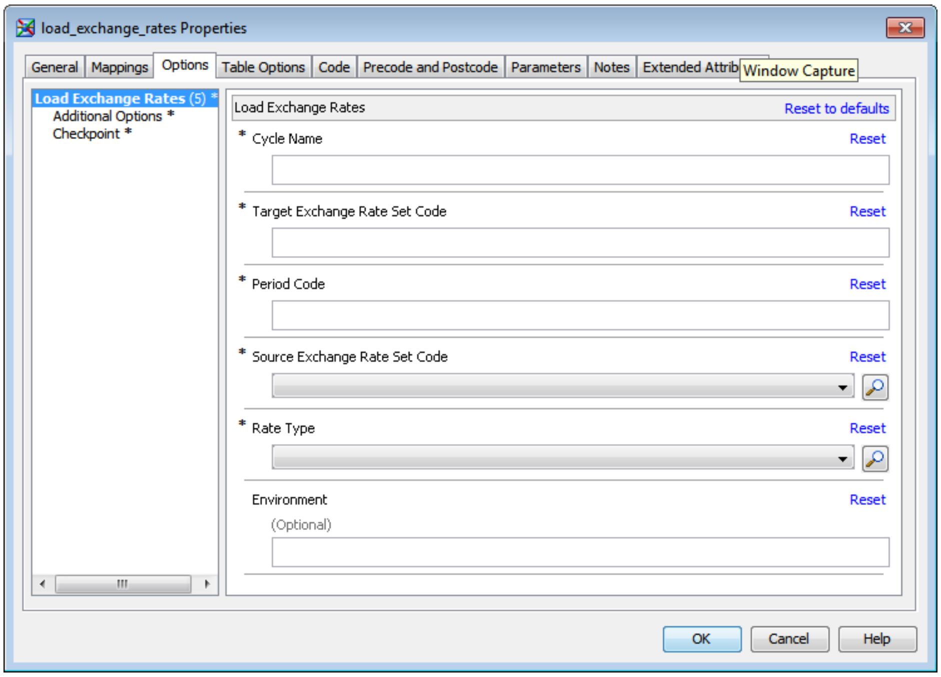
1. Select the Rates workspace.
2. In the Exchange Rate Sets view, select the exchange rate set into which you want to load exchange rates.
3. Select **Load Exchange Rates** to launch the Load Exchange Rates wizard.
4. Work through the wizard, consulting the online Help for the individual wizard pages as necessary.

Using a Job to Load Exchange Rates into the Data Mart

To load exchange rates using SAS Data Integration Studio, prepare and run a copy of the fm_1300_load_exchange_rates job. It is a good idea to create and maintain a separate, appropriately named job for each set of option values. Changing the option values of a job from time to time is possible, but it is likely to generate confusion.

On the **Folders** tab, the fm_1300_load_exchange_rates job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the load_exchange_rates transformation, which has the following options:



Provide values for the options as follows:

- **Cycle Name** is the name of the cycle to which the target exchange rate set belongs.
- **Target Exchange Rate Set Code** is the code of the target exchange rate set.
- **Period Code** is the code of the time period for which you are loading exchange rates.
- **Source Exchange Rate Set Code** is the code of the source exchange rate set in the staging area. Use the drop-down list to select a valid code.
- **Rate Type** is the exchange rate type for which you are loading exchange rates. Use the drop-down list to select a valid exchange rate type.

  If you select a simple exchange rate type, then the job gets the exchange rates from the CURRENCY_EXCH_RATE table. If you select a complex exchange rate type, then the job gets the exchange rates from the CURRENCY_COMPLEX_ EXCH_RATE table.
- **Environment** can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment "default" is used.

Run the job and then review the log. If there were errors, the log lists the location of an HTML error report.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Load Exchange Rates into the Data Mart

To load exchange rates using a SAS macro:

1. Create a SAS data set that specifies the combinations of exchange rate types and time periods for which you want to load exchange rates.
2. Run the etlxrteb.sas macro file.

Detailed instructions are inside the macro file, which is on the data-tier server.

On a Windows server, the etlxrteb.sas macro file is at the following location: **`!SASROOT\finance\sasmacro`**. On a UNIX server, the etlxrteb.sas macro file is at the following location: **`!SASROOT/sasautos`**.

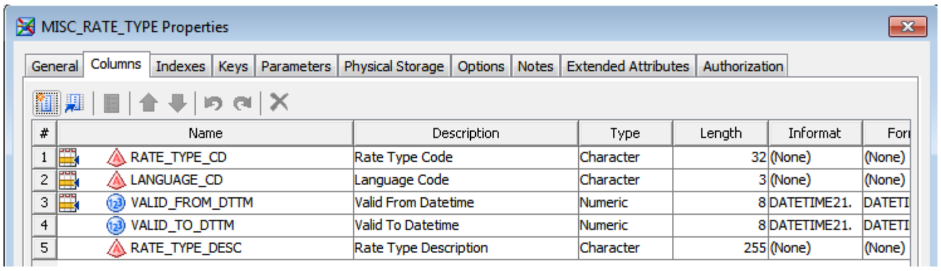
The macro can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Chapter 13

Loading Driver Rates into a SAS Financial Management Driver Rate Set

Driver Rate Types

Every driver rate that you load must belong to a driver rate type that you define in the Rates workspace of SAS Financial Management Studio.



#	Name	Description	Type	Length	Informat	For
1	RATE_TYPE_CD	Rate Type Code	Character	32	(None)	(None)
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETI
4	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETI
5	RATE_TYPE_DESC	Rate Type Description	Character	255	(None)	(None)

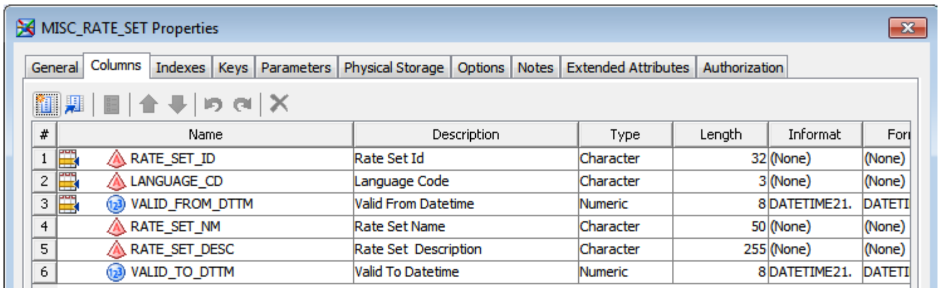
Driver rate types are related to driver rates exactly as exchange rate types are related to exchange rates. The key difference is that all exchange rate types are predefined, but no driver rate types are predefined.

You must load driver rate types from the MISC_RATE_TYPE table into the staging area before you can load driver rates that belong to those driver rate types into the staging area.

Driver Rate Sets

In SAS Financial Management, every driver rate belongs to a SAS Financial Management driver rate set. You define SAS Financial Management driver rate sets in the Rates workspace of SAS Financial Management Studio, where you give them codes, names, and descriptions.

In the staging area, every driver rate belongs to a SAS Financial Management staging area driver rate set. You define staging area driver rate sets in the MISC_RATE_SET table:


MISC_RATE_SET Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	For
1	RATE_SET_ID	Rate Set Id	Character	32	(None)	(None)
2	LANGUAGE_CD	Language Code	Character	3	(None)	(None)
3	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETI
4	RATE_SET_NM	Rate Set Name	Character	50	(None)	(None)
5	RATE_SET_DESC	Rate Set Description	Character	255	(None)	(None)
6	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETI

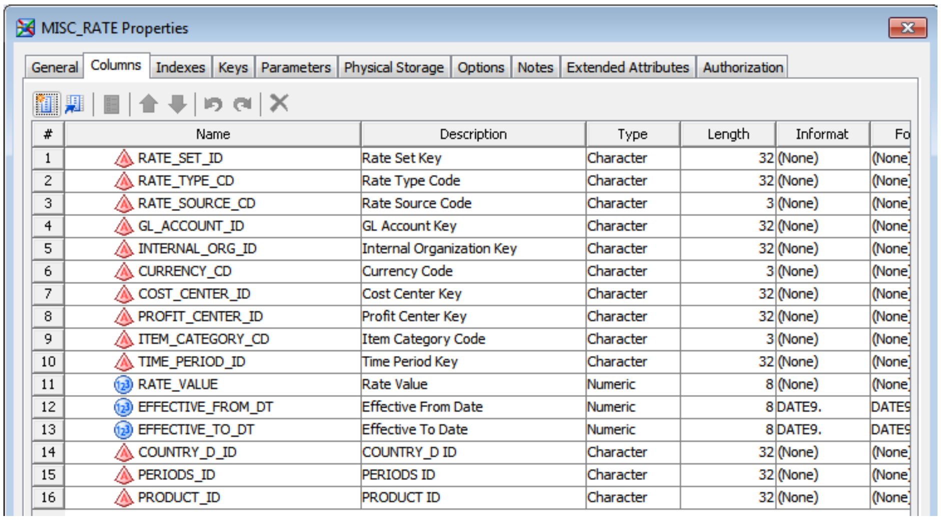
In building records for the MISC_RATE_SET table, note the following:

- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- It is advisable to maintain a one-to-one correspondence between staging area driver rate sets and SAS Financial Management driver rate sets, and to coordinate the codes, names, and descriptions of the corresponding pairs.

You must load the definitions of your staging area driver rate sets into the staging area before you can load driver rates that belong to those driver rate sets into the staging area.

Loading Driver Rates into Their Staging Table

Write a job that extracts driver rates from appropriate sources and loads them into the MISC_RATE staging table:


MISC_RATE Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	Fo
1	RATE_SET_ID	Rate Set Key	Character	32	(None)	(None
2	RATE_TYPE_CD	Rate Type Code	Character	32	(None)	(None
3	RATE_SOURCE_CD	Rate Source Code	Character	3	(None)	(None
4	GL_ACCOUNT_ID	GL Account Key	Character	32	(None)	(None
5	INTERNAL_ORG_ID	Internal Organization Key	Character	32	(None)	(None
6	CURRENCY_CD	Currency Code	Character	3	(None)	(None
7	COST_CENTER_ID	Cost Center Key	Character	32	(None)	(None
8	PROFIT_CENTER_ID	Profit Center Key	Character	32	(None)	(None
9	ITEM_CATEGORY_CD	Item Category Code	Character	3	(None)	(None
10	TIME_PERIOD_ID	Time Period Key	Character	32	(None)	(None
11	RATE_VALUE	Rate Value	Numeric	8	(None)	(None
12	EFFECTIVE_FROM_DT	Effective From Date	Numeric	8	DATE9.	DATE9
13	EFFECTIVE_TO_DT	Effective To Date	Numeric	8	DATE9.	DATE9
14	COUNTRY_D_ID	COUNTRY_D ID	Character	32	(None)	(None
15	PERIODS_ID	PERIODS ID	Character	32	(None)	(None
16	PRODUCT_ID	PRODUCT ID	Character	32	(None)	(None

In building records for this staging table, note the following:

- Rate Type Code must be one of the rate type codes that you define in the MISC_RATE_TYPE table. See "Driver Rate Types" on page 73.

- Rate Set Key indicates to which staging area driver rate set the driver rate belongs. It must be one of the values in the MISC_RATE_SET table. See "Driver Rate Sets" on page 73.
- Rate Source Code indicates where the driver rate was extracted from. Unlike the Exchange Rate Source Code column in the exchange rate staging tables, this column is not validated and can be left blank.
- Rate Value is the numeric driver rate.
- GL Account Key and the other Key columns contain member codes that the driver rate depends on. In each record, at least one of these columns must contain a member code.
- Leave the **Effective From Date** field blank. It is not used.
- Leave the **Effective To Date** field blank. It is not used.

Loading Driver Rates from the Staging Area to the Data Mart

Overview of Loading Driver Rates from the Staging Area to the Data Mart

There are three ways to load driver rates from the staging area into a SAS Financial Management driver rate set in the Data Mart:

- Use the Load Driver Rates wizard in SAS Financial Management Studio.
- Use a SAS Data Integration Studio job.
- Use a SAS macro.

Each time you run the SAS Data Integration Studio job, it loads driver rates for only one driver rate type. The Load Driver Rates wizard and the SAS macro can handle many driver rate types in a single run. Because there is substantial overhead associated with each run, the wizard and the SAS macro become increasingly advantageous as the number of driver rate types increases.

When you load driver rates into a SAS Financial Management driver rate set by any method, all the driver rates currently in the target driver rate set for the specified driver rate types are deleted before the new driver rates are loaded.

Using the Load Driver Rates Wizard to Load Driver Rates into the Data Mart

To load driver rates using the Load Driver Rates wizard in SAS Financial Management Studio:

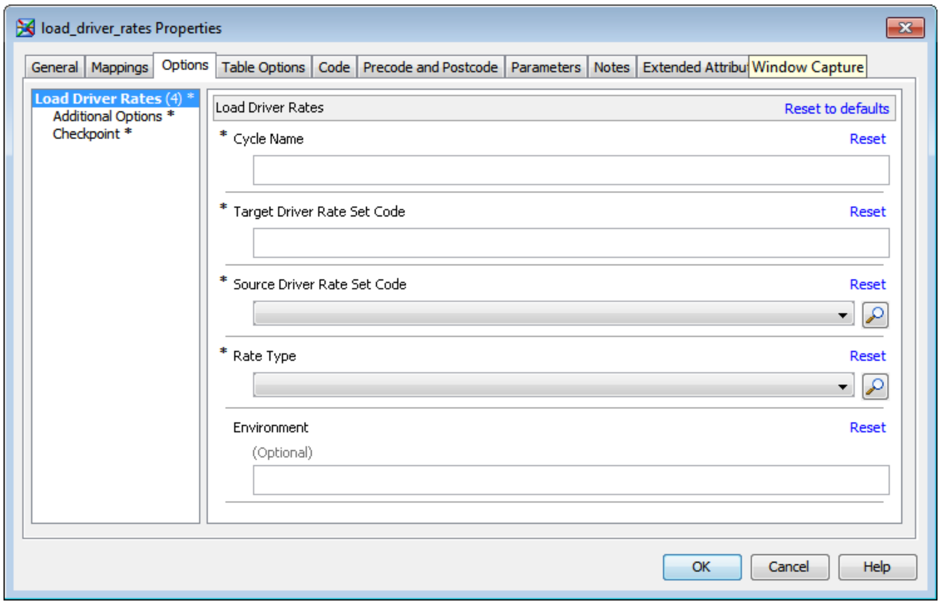
1. Select the Rates workspace.
2. In the Driver Rate Sets view, select the driver rate set into which you want to load driver rates.
3. Select **Load Driver Rates** to launch the Load Driver Rates wizard.
4. Work through the wizard, consulting the online Help for the individual wizard pages as necessary.

Using a Job to Load Driver Rates into the Data Mart

To load driver rates using SAS Data Integration Studio, prepare and run a copy of the fm_1500_load_driver_rates job. It is a good idea to create and maintain a separate, appropriately named job for each set of option values. Changing the option values of a job from time to time is possible, but it is likely to generate confusion.

On the **Folders** tab, the fm_1500_load_driver_rates job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the load_driver_rates transformation, which has the following options:



Provide values for the options as follows:

- **Cycle Name** is the name of the cycle to which the target driver rate set belongs.
- **Target Driver Rate Set Code** is the code of the target driver rate set.
- **Source Driver Rate Set Code** is the code of the source driver rate set in the SAS Financial Management staging area. Use the drop-down list to select a valid code.
- **Rate Type** is the driver rate type for which you are loading driver rates. Use the drop-down list to select a valid driver rate type.
- **Environment** can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment “default” is used.

Run the job and then review the log. If there were errors, the log lists the location of an HTML error report.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Load Driver Rates into the Data Mart

To load driver rates using a SAS macro:

1. Create a SAS data set that specifies the driver rate types for which you want to load driver rates.
2. Run the etldrteb.sas macro file.

Detailed instructions are inside the macro file, which is on the data-tier server.

On a Windows server, the etldrteb.sas macro file is at the following location: `!SASROOT\finance\sasmacro`. On a UNIX server, the etldrteb.sas macro file is at the following location: `!SASROOT/sasautos`.

The macro can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Chapter 14

Loading Cell Protection Rules for a Model

About Cell Protection Rules

You can protect cell crossings in a data-entry form by creating one or more rules that apply to the dimensions in the data-entry table.

Cells are protected against the following actions:

- manual data entry
- spread
- automatic allocation (applies only to financial forms in a bottom-up workflow)

However, the values of these protected cells can still change as the result of indirect actions, including the following:

- calculations
- changes in the values of descendants that roll up to the protected cell
- changes in cell protection rules
- changes in previous periods when frequency is To Date (for example, Year To Date or Quarter To Date)
- data that is loaded via SAS Data Integration Studio jobs
- data that was seeded from other models
- rules-based adjustments and allocations

- manual adjustments

Cell protection is applied in the following order:

1. Rules that are defined in a model. These rules are inherited by every form set that uses the model.
2. Rules that are defined in a form template. These rules, as well as the rules from the model, are inherited by all forms in the form set.
3. Cell protection that is set in a data-entry form. This protection applies only to the form in which it is defined and applies only to financial forms. You must set form-based cell protection in Microsoft Excel, but the protected cells are visible (and honored) in the Web-based Form Editor as well.

A form cannot override the protection that was set in the form set or the model, and a form set cannot override the protection that was set in the model. For example, if the model rules protect a specific crossing, the form set and its forms cannot unprotect it. However, both the form template and individual forms can define additional cell protection.

Loading Cell Protection Rules for a Model

You can load cell protection rules for a model using any of these methods:

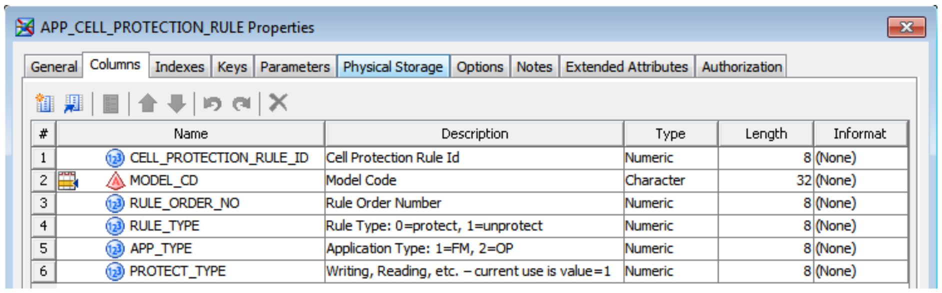
- In SAS Data Integration Studio, you can run a job that loads the cell protection rules for the selected model from the staging tables to the Data Mart.
- You can invoke a macro that accomplishes the same thing as the SAS Data Integration Studio job.
- In the Models workspace of SAS Financial Management Studio, you can select a model and select **Show cell protection rules**. A worksheet opens in Microsoft Excel, with the Cell Protection window open. In that window, you can define cell protection rules for the model. For details, see the online Help for the Excel add-in or “Working with Forms and Form Sets” in the *SAS Financial Management: User's Guide*.

  Note: If you subsequently load the Data Mart database via an ETL job or a macro, the rules that you defined in Excel are deleted.

Loading Cell Protection Rules into Their Staging Tables

There are two staging tables for cell protection.

The APP_CELL_PROTECTION_RULE table defines the rules.

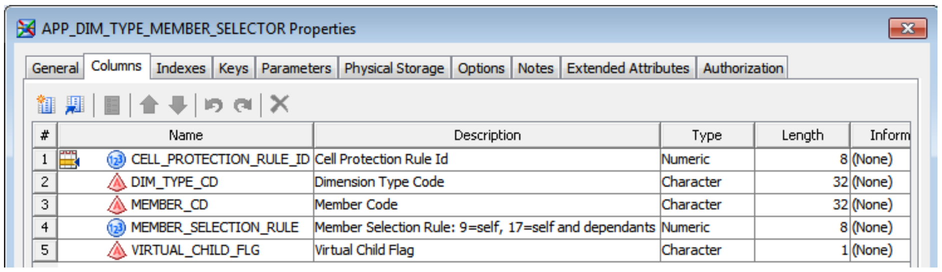


#	Name	Description	Type	Length	Informat
1	CELL_PROTECTION_RULE_ID	Cell Protection Rule Id	Numeric	8	(None)
2	MODEL_CD	Model Code	Character	32	(None)
3	RULE_ORDER_NO	Rule Order Number	Numeric	8	(None)
4	RULE_TYPE	Rule Type: 0=protect, 1=unprotect	Numeric	8	(None)
5	APP_TYPE	Application Type: 1=FM, 2=OP	Numeric	8	(None)
6	PROTECT_TYPE	Writing, Reading, etc. – current use is value=1	Numeric	8	(None)

The APP_CELL_PROTECTION_RULE table has the following columns:

Column	Description
CELL_PROTECTION_RULE_ID	The rule ID. It must correspond to CELL_PROTECTION_RULE_ID in the APP_DIM_TYPE_MEMBER_SELECTOR table, in a one-to-many relationship.
MODEL_CD	The model code.
RULE_ORDER_NO	The sequence (starting with 1) in which rules are applied for this model.
RULE_TYPE	The type of rule: 0 (protect) or 1 (unprotect).
APP_TYPE	A value of 1 means that the rule applies to financial planning models; a value of 2, operational planning models.
PROTECT_TYPE	This column is reserved for future use. It should have a value of 1.

The APP_DIM_TYPE_MEMBER_SELECTOR table selects the members to which each rule applies. Each rule can apply to one or more members of one or more dimensions.



#	Name	Description	Type	Length	Inform
1	CELL_PROTECTION_RULE_ID	Cell Protection Rule Id	Numeric	8	(None)
2	DIM_TYPE_CD	Dimension Type Code	Character	32	(None)
3	MEMBER_CD	Member Code	Character	32	(None)
4	MEMBER_SELECTION_RULE	Member Selection Rule: 9=self, 17=self and dependants	Numeric	8	(None)
5	VIRTUAL_CHILD_FLG	Virtual Child Flag	Character	1	(None)

The APP_DIM_TYPE_MEMBER_SELECTOR table has the following columns:

Column	Description
CELL_PROTECTION_RULE_ID	The rule ID. It must correspond to the CELL_PROTECTION_RULE_ID column in table APP_CELL_PROTECTION_RULE.

Column	Description
DIM_TYPE_CD	The dimension type code.
MEMBER_CD	The member code.
MEMBER_SELECTION_RULE	The following lists the valid values for this column: • 0: The rule does not apply to the member or any of its descendants. • 2: The rule applies only to the leaf descendants. • 4: The rule applies only to the immediate subordinate members. • 8: The rule applies only to all of the subordinate members. • 9: The rule applies only to the specified member. • 11: The rule applies to the member and the leaf descendants. • 13: The rule applies to the member and all of its immediate subordinate members. • 17: The rule applies to the member and all of its descendants.
VIRTUAL_CHILD_FLG	Specifies whether the member is a virtual child. Valid values are N and Y.

Loading Cell Protection Rules from the Staging Tables to the Data Mart

Overview

There are two ways to load cell protection rules for a model from the staging tables into the Data Mart:

- Use a SAS Data Integration Studio job.
- Use a SAS macro.

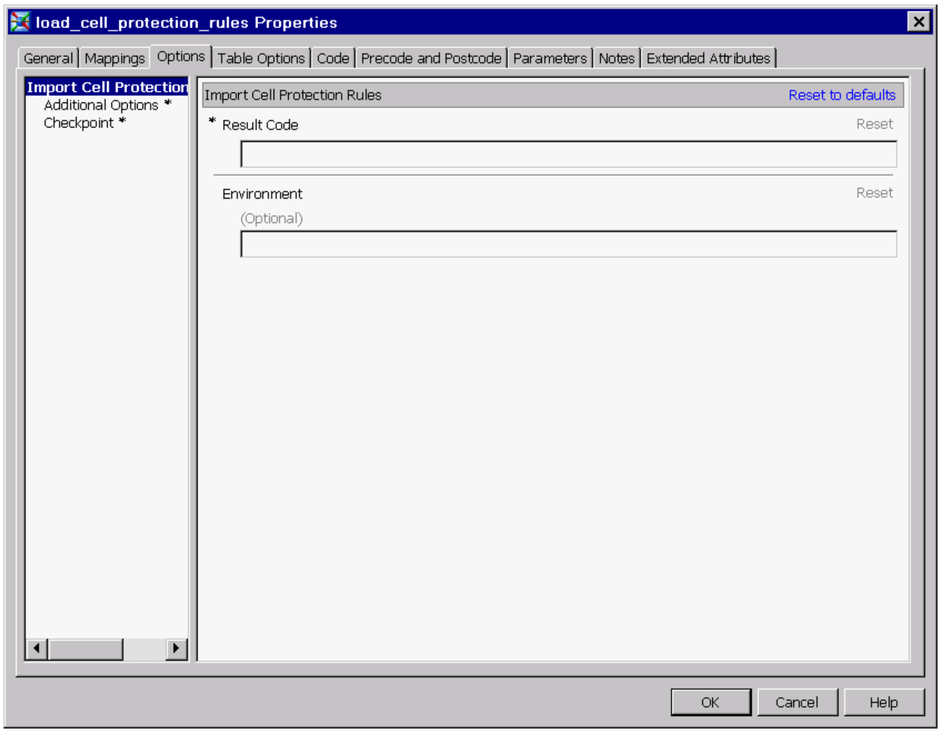
Each time you run the SAS Data Integration Studio job or the macro, it loads cell protection rules for the specified model. It deletes any rules that previously existed for that model and loads the rules that are defined in the staging tables.

Using a Job to Load Cell Protection Rules

To load cell protection rules using SAS Data Integration Studio, prepare and run a copy of the fm_2100_import_cell_protection_rules job. It is a good idea to create and maintain a separate, appropriately named job for each set of option values. Changing the option values of a job from time to time is possible, but it is likely to generate confusion.

On the **Folders** tab, the fm_2100_import_cell_protection_rules job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the load_cell_protection_rules transformation, which has the following options:



Provide values for the options as follows:

- **Result Code**: the code for the model whose rules you are loading.
- (Optional) **Environment**: the name of the middle-tier environment (for authentication purposes). The default value is **`default`**.

Run the job and then review the log. If there were errors, the log lists the location of an HTML error report.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Load Cell Protection Rules

The %ETLLDCPR macro loads cell protection rules for a specified model from the staging tables to the Data Mart. After loading the staging tables, run the macro, which has the following syntax:

ETLLDCPR(resultCode, <environment>)

resultCode
is the model code. Only rules for the specified model are loaded.

environment
is the name of the middle-tier environment (for authentication purposes). If you do not specify an environment, a value of **`default`** is used.

You can invoke the macro from an interactive SAS session, or you can write a stored process that calls the macro. In either case, SAS Remote Services and the managed servers must be running on the middle-tier server. See *SAS Financial Management: System Administration Guide*. On a Windows server, the etlldcpr.sas macro file is located at: `!SASROOT\finance\sasmacro`. On a UNIX server, the etlldcpr.sas macro file is located at: `!SASROOT/sasautos`.

Exporting Cell Protection Rules

Overview

There are two ways to export cell protection rules for a model:

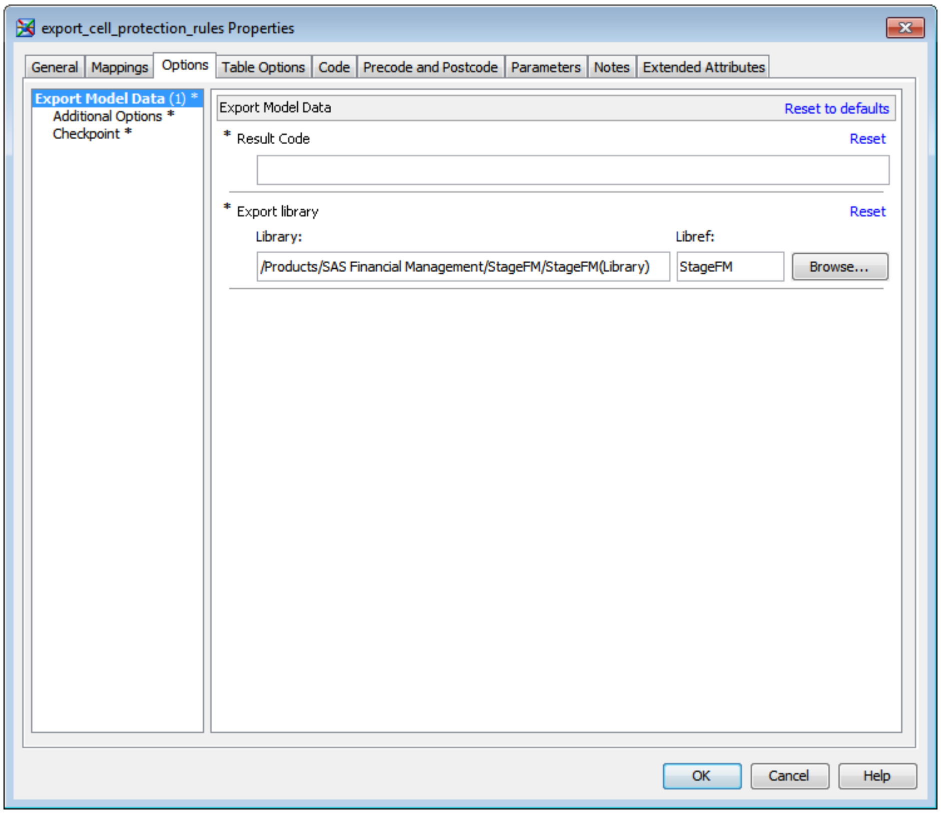
- Use a SAS Data Integration Studio job.
- Use a SAS macro.

Using a Job to Export Cell Protection Rules

To export cell protection rules using SAS Data Integration Studio, prepare and run a copy of the fm_2300_export_cell_protection_rules job.

On the **Folders** tab, the fm_2300_export_cell_protection_rules job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the export_cell_protection_rules transformation, which has the following options:



Export Model Data

* Result Code

* Export library

Library: /Products/SAS Financial Management/StageFM/StageFM(Library)

Libref: StageFM

Provide values for the options as follows:

- **Result Code**: the code for the model whose rules you are loading.
- **Export library**: select from the available data libraries the target data library where the cell protection rules will be exported. The default value is the StageFM library.

Run the job and then review the log.

Using a SAS Macro to Export Cell Protection Rules

The %ETLCPREX macro exports cell protection rules for a specified model from the Data Mart to a specified target data library. Run the macro, which has the following syntax:

ETLCPREX(resultCode, <exportLib>)

resultCode
is the model code. Only rules for the specified model are loaded.

exportLib
specifies the target data library for the cell protection rules to be exported.

You can invoke the macro from an interactive SAS session, or you can write a stored process that calls the macro. Before you run the macro from a SAS session, the target data library has to be assigned in the SAS session. On a Windows server, the etlcprex.sas macro file is located at: **`!SASROOT\finance\sasmacro`**. On a UNIX server, the etlcprex.sas macro file is located at: **`!SASROOT/sasautos`**.

Chapter 15

Loading Base Data into a Financial Cycle

Overview of Loading Base Financial Data

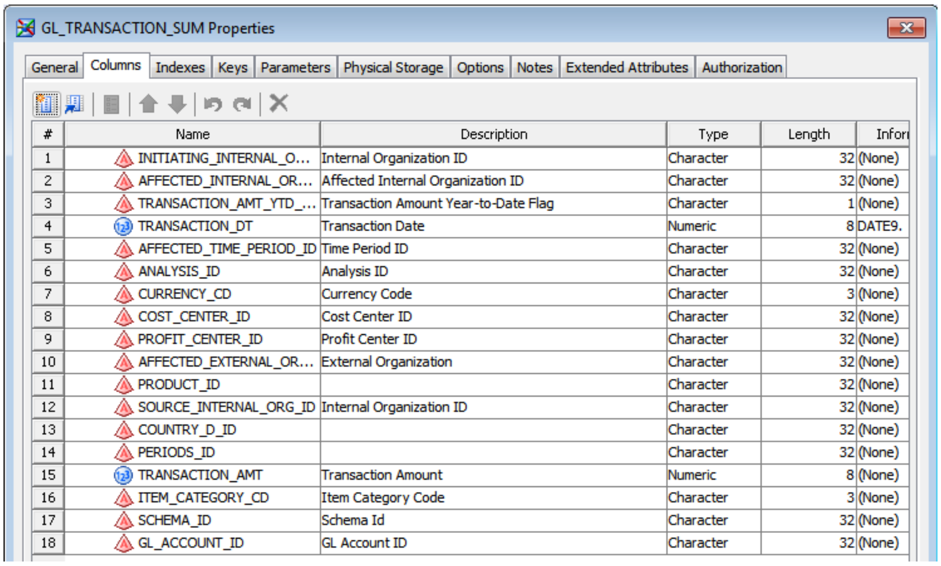
Base financial data is loaded into a financial cycle in SAS Financial Management. The main source of base financial data is general ledger account balances. A secondary and optional source is journal data.

If you load both general ledger data and journal data, be careful to avoid loading journal data that is also reflected in the general ledger data that you are loading. That would lead to double-counting of the same financial transactions.

Loading Base Financial Data from Its Source to the Staging Tables

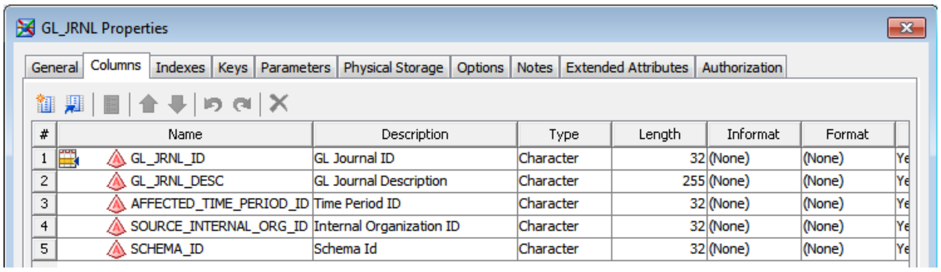
Overview of the Base Financial Data Staging Tables

For general ledger data, write a job to extract the data from its source and load it into the GL_TRANSACTION_SUM table:

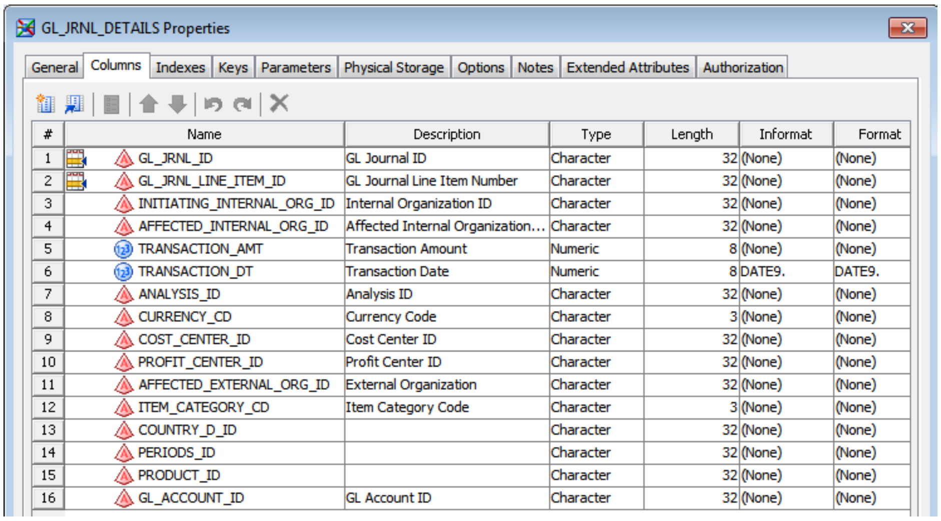


#	Name	Description	Type	Length	Infor
1	INITIATING_INTERNAL_O…	Internal Organization ID	Character	32	(None)
2	AFFECTED_INTERNAL_OR…	Affected Internal Organization ID	Character	32	(None)
3	TRANSACTION_AMT_YTD_…	Transaction Amount Year-to-Date Flag	Character	1	(None)
4	TRANSACTION_DT	Transaction Date	Numeric	8	DATE9.
5	AFFECTED_TIME_PERIOD_ID	Time Period ID	Character	32	(None)
6	ANALYSIS_ID	Analysis ID	Character	32	(None)
7	CURRENCY_CD	Currency Code	Character	3	(None)
8	COST_CENTER_ID	Cost Center ID	Character	32	(None)
9	PROFIT_CENTER_ID	Profit Center ID	Character	32	(None)
10	AFFECTED_EXTERNAL_OR…	External Organization	Character	32	(None)
11	PRODUCT_ID		Character	32	(None)
12	SOURCE_INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)
13	COUNTRY_D_ID		Character	32	(None)
14	PERIODS_ID		Character	32	(None)
15	TRANSACTION_AMT	Transaction Amount	Numeric	8	(None)
16	ITEM_CATEGORY_CD	Item Category Code	Character	3	(None)
17	SCHEMA_ID	Schema Id	Character	32	(None)
18	GL_ACCOUNT_ID	GL Account ID	Character	32	(None)

For journal data, write a job to extract the data from its source and load it into the GL_JRNL and GL_JRNL_DETAILS tables:



#	Name	Description	Type	Length	Informat	Format	
1	GL_JRNL_ID	GL Journal ID	Character	32	(None)	(None)	Ye
2	GL_JRNL_DESC	GL Journal Description	Character	255	(None)	(None)	Ye
3	AFFECTED_TIME_PERIOD_ID	Time Period ID	Character	32	(None)	(None)	Ye
4	SOURCE_INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)	(None)	Ye
5	SCHEMA_ID	Schema Id	Character	32	(None)	(None)	Ye



#	Name	Description	Type	Length	Informat	Format
1	GL_JRNL_ID	GL Journal ID	Character	32	(None)	(None)
2	GL_JRNL_LINE_ITEM_ID	GL Journal Line Item Number	Character	32	(None)	(None)
3	INITIATING_INTERNAL_ORG_ID	Internal Organization ID	Character	32	(None)	(None)
4	AFFECTED_INTERNAL_ORG_ID	Affected Internal Organization…	Character	32	(None)	(None)
5	TRANSACTION_AMT	Transaction Amount	Numeric	8	(None)	(None)
6	TRANSACTION_DT	Transaction Date	Numeric	8	DATE9.	DATE9.
7	ANALYSIS_ID	Analysis ID	Character	32	(None)	(None)
8	CURRENCY_CD	Currency Code	Character	3	(None)	(None)
9	COST_CENTER_ID	Cost Center ID	Character	32	(None)	(None)
10	PROFIT_CENTER_ID	Profit Center ID	Character	32	(None)	(None)
11	AFFECTED_EXTERNAL_ORG_ID	External Organization	Character	32	(None)	(None)
12	ITEM_CATEGORY_CD	Item Category Code	Character	3	(None)	(None)
13	COUNTRY_D_ID		Character	32	(None)	(None)
14	PERIODS_ID		Character	32	(None)	(None)
15	PRODUCT_ID		Character	32	(None)	(None)
16	GL_ACCOUNT_ID	GL Account ID	Character	32	(None)	(None)

Note that the column layout for general ledger data is similar to the column layout for journal data. In the case of general ledger data, the GL_TRANSACTION_SUM table contains all the columns. In the case of journal data, the GL_JRNL table identifies journal entries, each of which can include several data records in the GL_JRNL_DETAILS table. GL_JRNL contains the columns that must have the same value for all the data records that belong to a given journal entry.

GL Journal ID must have a unique value for each record in GL_JRNL. You can generate the unique values in any way that you find convenient. In GL_JRNL_DETAILS, the combination of GL Journal ID and GL Journal Line Item Number must be unique for each record.

The Schema ID column in GL_TRANSACTION_SUM and GL_JRNL is not used. Leave this column blank.

Columns That Hold Members

The following columns must contain a valid member code in every record because they represent dimension types that are automatically included in every financial cycle:

- Initiating Internal Organization ID (Organization)
- Affected Internal Organization ID (Trader)
- GL Account ID
- Analysis ID
- Currency Code
- Time Period ID

The SOURCE_INTERNAL_ORG_ID column must contain a valid member code in every record. This column indicates the organization that is the source of the data record. In many cases, this is the same organization that you place in the INITIATING_INTERNAL_ORG_ID column, which indicates the organization that the record describes.

The member codes in a record must satisfy the following constraints:

- The Organization code must not be ALL or EXT.
- The Organization code and the Trader code must be different.
- If the account that is specified in the GL Account ID column has a value of *N* for its Intercompany Account Flag, then the value of Affected Internal Organization ID (Trader) must be EXT. (This constraint applies only if you load the data into a cycle for which the "Non-intercompany accounts must be associated with the external trading member" property is set.)
- If the specified account has a value of *Y* for its Intercompany Account Flag, then the value of Affected Internal Organization ID (Trader) must not be EXT. (This constraint applies only if you load the data into a cycle for which the "Intercompany accounts must be associated with an intercompany trading partner" property is set.)

Leave the following columns empty if these dimension types are not used to describe the data that you are loading:

- Cost Center ID
- Profit Center ID
- External Organization
- Item Category Code

If you add other dimension types to your data model, then they are represented by additional columns in GL_TRANSACTION_SUM and GL_JRNL_DETAILS that are not shown here. You must provide valid member codes for any dimension type that is included in the cycle that is the destination of the data. For a discussion of adding dimension types, see Chapter 10, “Adding a Dimension Type,” on page 55.

Columns That Specify the Numeric Values

Overview of Columns That Specify the Numeric Values

In both GL_TRANSACTION_SUM and GL_JRNL_DETAILS, the Transaction Amount column holds the base numeric values.

In GL_TRANSACTION_SUM, the interpretation of the Transaction Amount values is affected by the Transaction Amount Year-to-Date Flag column. For each record, you must load this column with either a *Y* or an *N*. If you leave the Transaction Amount Year-to-Date Flag column empty, then the record is ignored.

GL_JRNL_DETAILS does not have a Transaction Amount Year-to-Date Flag column. Every record in that table is processed in the same manner as an *N* record in GL_TRANSACTION_SUM.

The explanation of the *Y/N* choice for the Transaction Amount Year-to-Date Flag column follows.

Setting the Year-To-Date Flag

For a Revenue or Expense account, the value that is stored in the Data Mart must represent the revenue received or expense incurred during the designated time period. For an Asset, Liability, or Equity account, the value that is stored in the Data Mart must represent the change in the value of the asset, liability, or equity item from the previous time period to the designated time period. SAS Financial Management computes the values of Asset, Liability, and Equity accounts by summing up a history of stored changes in value. All the numeric values that are stored in the Data Mart are called *period activity* values.

For each record of GL_TRANSACTION_SUM, if you load the period activity value that is required by the Data Mart into the Transaction Amount column, then you should place *N* in the Transaction Amount Year-to-Date Flag column. Thus, use *N* for the flag in the following cases:

- The record concerns a Revenue account and the Transaction Amount is the revenue received during the designated time period.
- The record concerns an Expense account and the Transaction Amount is the expense incurred during the designated time period.
- The record concerns an Asset, Liability, or Equity account and the Transaction Amount is the change in the value of the asset, liability, or equity item from the previous time period to the designated time period.

You should use *Y* for the flag in the following cases:

- The record concerns a Revenue account and the Transaction Amount is the cumulative year-to-date revenue through the designated time period.
- The record concerns an Expense account and the Transaction Amount is the cumulative year-to-date expense through the designated time period.
- The record concerns an Asset, Liability, or Equity account and the Transaction Amount is the value of the asset, liability, or equity item in the designated time period.

For Statistical accounts, the Year-to-Date Flag is ignored, but still you must specify either *Y* or *N*.

How Year-To-Date Transaction Amounts Are Processed

For Statistical accounts, transaction amounts are always loaded without change into the Data Mart, whether the Year-to-Date Flag is *Y* or *N*.

For all other account types, if the year-to-date flag is *Y*, then the period activity value that is placed in the Data Mart is generally calculated as the Transaction Amount in GL_TRANSACTION_SUM for the same time period minus the Transaction Amount in GL_TRANSACTION_SUM for the previous time period. For example, a March year-to-date transaction amount of 100 and a February year-to-date transaction amount of 94 together yield a March period activity value of 6 in the Data Mart.

There are two important exceptions to this rule. A year-to-date Transaction Amount in GL_TRANSACTION_SUM is carried forward without change to the Data Mart if either of the following conditions is true:

- GL_TRANSACTION_SUM does not contain a corresponding record for the previous time period.
- The record concerns a Revenue or Expense account and the designated time period is the first period of a fiscal year, as determined by the relevant time hierarchy.

Note that the difference between the year-to-date values for two consecutive time periods can be calculated only if the table contains records for both time periods. This is so, even if the year-to-date value for one of the time periods is zero. If you set the year-to-date flag to *Y*, then be sure to include records for an unbroken sequence of time periods, including records with a Transaction Amount of zero where necessary.

How Multiple Records for the Same Combination of Members Are Processed

It is likely that your staging tables will contain at most one record for a given combination of members. However, this is not a requirement. You can create as many data records as you want for the same combination of members. You can even create a mix of year-to-date and non-year-to-date data records for the same combination of members. That would be pointless and confusing in most cases, but the software will handle it.

Suppose that you create many data records for the same combination of members, possibly including a mix of year-to-date and non-year-to-date records. The period activity values that are loaded into the Data Mart are computed as follows:

1. All year-to-date transaction amounts for a given combination of members are summed, yielding a net year-to-date amount for that combination of members.
2. The net year-to-date amount for one time period is subtracted from the net year-to-date amount for the following time period, yielding a period activity value that is based solely on the year-to-date amounts.
3. All non-year-to-date transaction amounts for a given combination of members are summed, yielding a period activity value for that combination of members that is based solely on non-year-to-date amounts.
4. For each combination of members, the period activity value that is based solely on year-to-date amounts is added to the period activity value that is based solely on non-year-to-date amounts. This yields the final period activity value that is loaded into the Data Mart.

Loading Base Financial Data from the Staging Area to the Data Mart

Overview of Loading Base Financial Data from the Staging Area to the Data Mart

When you load base financial data from the staging area into a financial cycle in the Data Mart, the three relevant staging area tables (GL_TRANSACTION_SUM, GL_JRNL, and GL_JRNL_DETAILS) always participate. You cannot load only general ledger data or only journal data unless one of the tables contains no relevant records.

There are three ways to load base financial data from the staging area into a financial cycle in the Data Mart:

- Use the Load New Data wizard in SAS Financial Management Studio.
- Use a SAS Data Integration Studio job.
- Use a SAS macro.

Using the Load New Data Wizard to Load Base Financial Data into the Data Mart

To load base financial data using SAS Financial Management Studio:

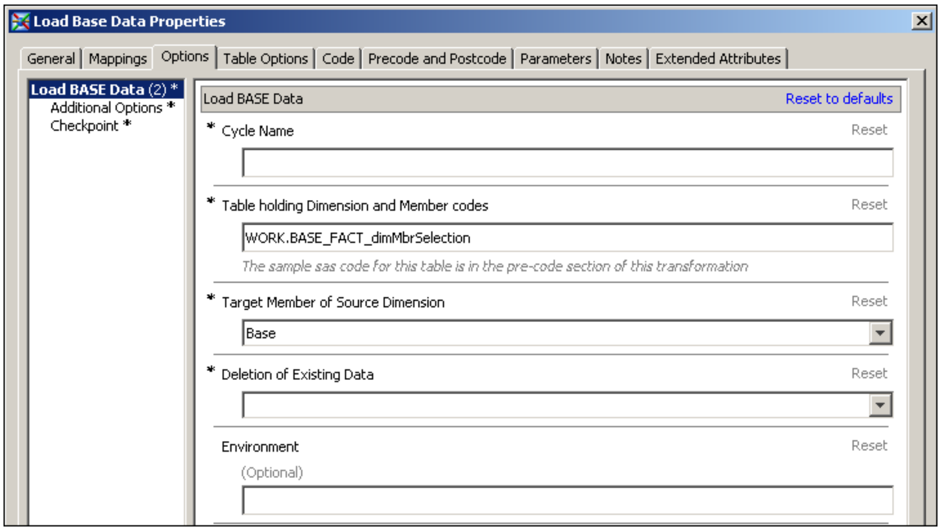
1. Open the financial cycle into which you want to load the data.
2. Select the Periods workspace.
3. In the Periods view, select the period or periods for which you want to load data.
4. Select **Load New Data** to launch the Load New Data wizard.
5. Work through the wizard, consulting the online Help for the individual wizard pages as necessary.

Using a Job to Load Base Financial Data into the Data Mart

To load base financial data using SAS Data Integration Studio, run either the fm_1100_load_base_data job or the fm_1100_load_base_data_unlock_periods job. The former job cannot load data into locked periods. The latter job unlocks any locked target periods, loads the data, and then locks again the periods that it unlocked. Otherwise, the two jobs are identical.

On the **Folders** tab, these jobs are in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the Load Base Data (or Load Base Data Unlock Periods) transformation, which has the following options:


Load Base Data Properties

General | Mappings | Options | Table Options | Code | Precode and Postcode | Parameters | Notes | Extended Attributes

Load BASE Data (2) *
Additional Options *
Checkpoint *

Load BASE Data — Reset to defaults

* Cycle Name — Reset

* Table holding Dimension and Member codes — Reset
WORK.BASE_FACT_dimMbrSelection
The sample sas code for this table is in the pre-code section of this transformation

* Target Member of Source Dimension — Reset
Base

* Deletion of Existing Data — Reset

Environment — Reset
(Optional)

Provide values for the options as follows:

- **Cycle Name**: the name of the SAS Financial Management cycle into which you are loading the data.
- **Table Holding Dimension and Member Codes**: the name of a SAS data set that specifies the dimension and member combinations to load data for. The **Precode** region on the **Precode and Postcode** tab contains sample code that builds a SAS data set with the required layout. By default, the job uses the SAS data set that is built by the Precode program. Do one of the following:
  - On the **Precode and Postcode** tab, make sure that the **Precode** check box is selected. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that is built by the precode. Modify the precode to build the table that you need. In this case, the precode runs before the job, and then the job uses the SAS data set that the precode builds.
  - On the **Precode and Postcode** tab, make sure that the **Precode** check box is not selected. Build the SAS data set that you need using a means other than the precode. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that you built for this purpose.
- **Target Member of Source Dimension**: select **Base** or **BaseForm** from the drop-down list.

  If you select **Base**, then relevant general ledger data in GL_TRANSACTION_SUM becomes associated with the Base member of the Source hierarchy.

  If you select **BaseForm**, then relevant general ledger data in GL_TRANSACTION_SUM becomes associated with the BaseForm member of the Source hierarchy.

  Relevant journal data in GL_JRNL and GL_JRNL_DETAILS becomes associated with the BaseJourn member of the Source hierarchy, no matter what you select here.
- **Deletion of Existing Data**: select **Replace All** or **Replace Matching** from the drop-down list.

  If you select **Replace All**, then you must select either **Yes** or **No** for **Preserve Data Entered via Web Form**.

  If you select **Replace Matching**, then you must select either **Yes** or **No** for **Ignore Currency Dimension**. The four possible combinations specify the four available deletion policies:

- Replace all, preserve form data — Data is deleted from all the crossings that you have specified as eligible to receive data, but not including data that was entered through forms or stored computed values of driver formulas. In each case where data is loaded to a crossing that already has form-entered data or stored driver-formula values, the result is additive. It is possible that data will be deleted from some crossings that do not receive data in the load operation.
- Replace all, do not preserve form data — Data is deleted from all the crossings that you have specified as eligible to receive data, without exception. Data that was loaded previously, data that was entered through forms, and stored computed values of driver formulas are all deleted. It is possible that data will be deleted from some crossings that do not receive data in the load operation.
- Replace matching, ignore currency dimension — Data is deleted only from crossings that match a crossing in the data that you are loading. A crossing in the new data matches a crossing in the existing data if the two records match member-for-member in every dimension except Source and Currency. The Source and Currency members can match or not.
- Replace matching, do not ignore currency dimension — Data is deleted only from crossings that match a crossing in the data that you are loading. A crossing in the new data matches a crossing in the existing data if either of the following conditions is satisfied:
  - The two crossings match member-for-member in every dimension except Source. The Source member can match or not.
  - The existing record was created through form data entry and the two crossings match member-for-member in every dimension except Source and Currency. The Source and Currency members can match or not.

- Environment can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment “default” is used.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Load Base Financial Data into the Data Mart

To load base financial data using a SAS macro:

1. Create a SAS data set that specifies the member combinations to load data for.
2. Run the etlldfcp.sas macro file.

Detailed instructions are inside the macro file, which is on the data-tier server.

On a Windows server, the etlldfcp.sas macro file is at the following location: `!SASROOT\finance\sasmacro`. On a UNIX server, the etlldfct.sas macro file is at the following location: `!SASROOT/sasautos`.

The macro can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Which Records Are Loaded?

A staging area record that contains base financial data is loaded only if the following conditions are met:

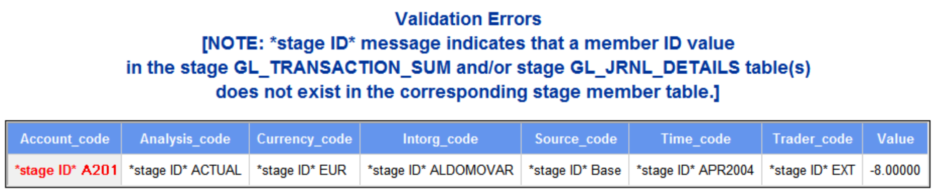
- The record contains a member for each dimension type (except Source) that is used by the target cycle.
- The record contains a member for no dimension type that is not used by the target cycle.
- Within each dimension type that is used by the record and the target cycle, the member in the record belongs to the dimension that is used by the target cycle.
- The record belongs to the subset of records that is defined in the wizard, job, or SAS macro that loads the data. In other words, for each dimension type for which one or more members are specified in the wizard, job, or SAS macro, one of those members is in the record.

Checking for Errors

After you run a job that uses the Load Base Data transformation, review the log. If there were errors, then the job is terminated and the log lists the location of an HTML error report. If the SAS macro is terminated, then the log lists the location of an HTML error report. Error reports for the job, the macro, and the Load New Data wizard are all available in SAS Financial Management Studio from the **History** page of the Properties window for the target cycle.

An error report is produced if any record violates any one of the following constraints:

- In the GL_TRANSACTION_SUM table, INITIATING_INTERNAL_ORG_ID must not be ALL or EXT.
- In the GL_TRANSACTION_SUM table, INITIATING_INTERNAL_ORG_ID and AFFECTED_INTERNAL_ORG_ID (Trader) must be different.
- If the account that is specified in the GL Account ID column of GL_TRANSACTION_SUM has a value of *N* for Intercompany Account Flag in the GL_ACCOUNT table, then the value of AFFECTED_INTERNAL_ORG_ID (Trader) in GL_TRANSACTION_SUM must be EXT. (This constraint applies only if you load the data into a SAS Financial Management cycle for which the “Non-intercompany accounts must be associated with the external trading member” property is set.)
- If the account that is specified in the GL Account ID column of GL_TRANSACTION_SUM has a value of *Y* for Intercompany Account Flag in the GL_ACCOUNT table, then the value of AFFECTED_INTERNAL_ORG_ID (Trader) in GL_TRANSACTION_SUM must not be EXT. (This constraint applies only if you load the data into a SAS Financial Management cycle for which the “Intercompany accounts must be associated with an intercompany trading partner” property is set.)
- Member IDs are also validated when they are imported into the Data Mart. A data record that has an ID value that does not also exist in the corresponding member table appears in the error report as follows:



Validation Errors
[NOTE: *stage ID* message indicates that a member ID value in the stage GL_TRANSACTION_SUM and/or stage GL_JRNL_DETAILS table(s) does not exist in the corresponding stage member table.]

Account_code	Analysis_code	Currency_code	Intorg_code	Source_code	Time_code	Trader_code	Value
stage ID A201	*stage ID* ACTUAL	*stage ID* EUR	*stage ID* ALDOMOVAR	*stage ID* Base	*stage ID* APR2004	*stage ID* EXT	-8.00000

Chapter 16

Loading Base Data into an Operational Cycle

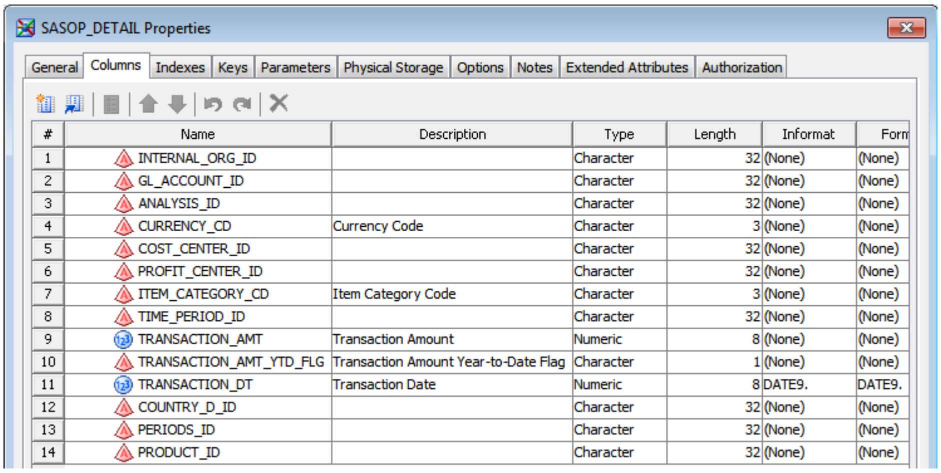
Overview of Loading Base Operational Data

Base operational data is loaded into an operational cycle in SAS Financial Management. Base operational data can come from a variety of transactional systems. Two typical examples of operational data are sales by product and employee salary history.

Moving Base Operational Data from Its Source to Its Staging Table

Overview of the Base Operational Data Staging Table

Write a job to extract the data from its source and load it into the SASOP_DETAIL table:



SASOP_DETAIL Properties

General | Columns | Indexes | Keys | Parameters | Physical Storage | Options | Notes | Extended Attributes | Authorization

#	Name	Description	Type	Length	Informat	Form
1	INTERNAL_ORG_ID		Character	32	(None)	(None)
2	GL_ACCOUNT_ID		Character	32	(None)	(None)
3	ANALYSIS_ID		Character	32	(None)	(None)
4	CURRENCY_CD	Currency Code	Character	3	(None)	(None)
5	COST_CENTER_ID		Character	32	(None)	(None)
6	PROFIT_CENTER_ID		Character	32	(None)	(None)
7	ITEM_CATEGORY_CD	Item Category Code	Character	3	(None)	(None)
8	TIME_PERIOD_ID		Character	32	(None)	(None)
9	TRANSACTION_AMT	Transaction Amount	Numeric	8	(None)	(None)
10	TRANSACTION_AMT_YTD_FLG	Transaction Amount Year-to-Date Flag	Character	1	(None)	(None)
11	TRANSACTION_DT	Transaction Date	Numeric	8	DATE9.	DATE9.
12	COUNTRY_D_ID		Character	32	(None)	(None)
13	PERIODS_ID		Character	32	(None)	(None)
14	PRODUCT_ID		Character	32	(None)	(None)

Columns That Hold Members

The following columns must contain a valid member code in every record because they represent dimension types that are automatically included in every operational cycle:

- Internal Organization ID

  The Internal Organization ID must not be ALL or EXT.

- GL Account ID
- Analysis ID
- Currency Code
- Time Period ID

Leave the following columns empty if these dimension types are not used to describe the data that you are loading:

- Cost Center ID
- Profit Center ID
- Item Category Code

If you add other dimension types to your data model, then they are represented by additional columns in SASOP_DETAIL that are not shown here. You must provide valid member codes for any dimension type that is included in the cycle that is the destination of the data. For a discussion of adding dimension types, see Chapter 10, “Adding a Dimension Type,” on page 55.

Columns That Specify the Numeric Values

Overview of Columns That Specify the Numeric Values

The Transaction Amount column holds the base numeric values.

The interpretation of the Transaction Amount values is affected by the Transaction Amount Year-to-Date Flag column. For each record, you must load this column with either a *Y* or an *N*. If you leave the Transaction Amount Year-to-Date Flag column empty, then the record is ignored.

The explanation of the *Y/N* choice for the Transaction Amount Year-to-Date Flag column follows.

Setting the Year-To-Date Flag

For a Revenue or Expense account, the value that is stored in the Data Mart must represent the revenue received or expense incurred during the designated time period. For an Asset, Liability, or Equity account, the value that is stored in the Data Mart must represent the change in the value of the asset, liability, or equity item from the previous time period to the designated time period. SAS Financial Management computes the values of Asset, Liability, and Equity accounts by summing up a history of stored changes in value. All the numeric values that are stored in the Data Mart are called *period activity* values.

For each record of SASOP_DETAIL, if you load the period activity value that is required by the Data Mart into the Transaction Amount column, then you should place *N* in the Transaction Amount Year-to-Date Flag column. Thus, use *N* for the flag in the following cases:

- The record concerns a Revenue account, and the Transaction Amount is the revenue received during the designated time period.
- The record concerns an Expense account and the Transaction Amount is the expense incurred during the designated time period.
- The record concerns an Asset, Liability, or Equity account and the Transaction Amount is the change in the value of the asset, liability, or equity item from the previous time period to the designated time period.

You should use *Y* for the flag in the following cases:

- The record concerns a Revenue account and the Transaction Amount is the cumulative year-to-date revenue through the designated time period.
- The record concerns an Expense account and the Transaction Amount is the cumulative year-to-date expense through the designated time period.
- The record concerns an Asset, Liability, or Equity account, and the Transaction Amount is the value of the asset, liability, or equity item in the designated time period.

For Statistical accounts, the Year-to-Date Flag is ignored, but still you must specify either *Y* or *N*.

How Year-To-Date Transaction Amounts Are Processed

For Statistical accounts, transaction amounts are always loaded without change into the Data Mart, whether the Year-to-Date Flag is *Y* or *N*.

For all other account types, if the year-to-date flag is *Y*, then the period activity value that is placed in the Data Mart is generally calculated as the Transaction Amount in SASOP_DETAIL for the same time period minus the Transaction Amount in SASOP_DETAIL for the previous time period. For example, a March year-to-date transaction amount of 100 and a February year-to-date transaction amount of 94 together yield a March period activity value of 6 in the Data Mart.

There are two important exceptions to this rule. A year-to-date Transaction Amount in SASOP_DETAIL is carried forward without change to the Data Mart if either of the following conditions is true:

- SASOP_DETAIL does not contain a corresponding record for the previous time period.

- The record concerns a Revenue or Expense account, and the designated time period is the first period of a fiscal year, as determined by the relevant time hierarchy.

Note that the difference between the year-to-date values for two consecutive time periods can be calculated only if the table contains records for both time periods. This is so, even if the year-to-date value for one of the time periods is zero. If you set the year-to-date flag to *Y*, then be sure to include records for an unbroken sequence of time periods, including records with a Transaction Amount of zero where necessary.

How Multiple Records for the Same Combination of Members Are Processed

It is likely that your staging tables will contain at most one record for a given combination of members. However, this is not a requirement. You can create as many data records as you want for the same combination of members. You can even create a mix of year-to-date and non-year-to-date data records for the same combination of members. That would be pointless and confusing in most cases, but the software can handle it.

Suppose that you create many data records for the same combination of members, possibly including a mix of year-to-date and non-year-to-date records. The period activity values that are loaded into the Data Mart are computed in this sequence:

1. All year-to-date transaction amounts for a given combination of members are summed. The sum is a net year-to-date amount for that combination of members.
2. The net year-to-date amount for one time period is subtracted from the net year-to-date amount for the following time period. This computation yields a period activity value that is based solely on the year-to-date amounts.
3. All non-year-to-date transaction amounts for a given combination of members are summed, yielding a period activity value for that combination of members that is based solely on non-year-to-date amounts.
4. For each combination of members, the period activity value that is based solely on year-to-date amounts is added to the period activity value that is based solely on non-year-to-date amounts. This yields the final period activity value that is loaded into the Data Mart.

Moving Base Operational Data from the Staging Area to the Data Mart

Overview of Moving Base Operational Data from the Staging Area to the Data Mart

There are three ways to load base operational data from the staging area into an operational cycle in the Data Mart:

- Use the Load Operational Data wizard in SAS Financial Management Studio.
- Use a SAS Data Integration Studio job.
- Use a SAS macro.

Using the Load Operational Data Wizard to Load Base Operational Data into the Data Mart

To load base operational data using SAS Financial Management Studio:

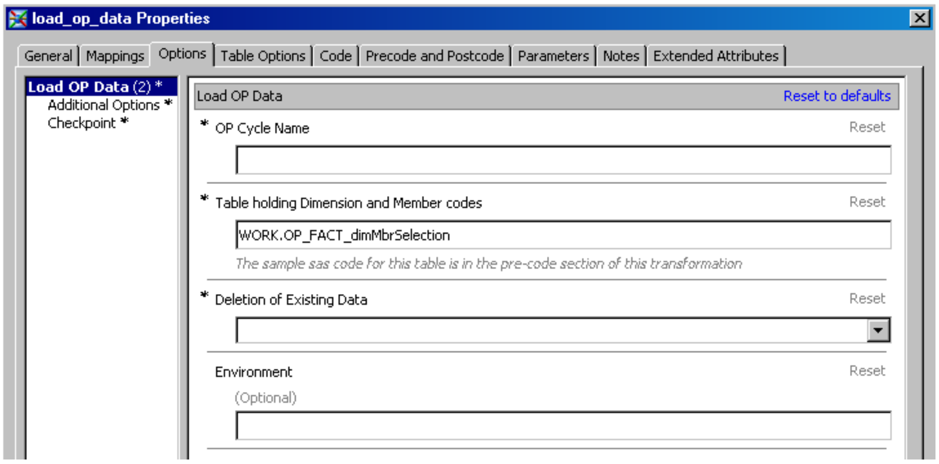
1. Open the operational cycle into which you want to load the data.
2. Select the Periods workspace.
3. In the Periods view, select the period or periods for which you want to load data.
4. Select **Load New Data** to launch the Load New Data wizard.
5. Work through the wizard, consulting the online Help for the individual wizard pages as necessary.

Using a Job to Load Base Operational Data into the Data Mart

To load base operational data using SAS Data Integration Studio, run the oplan_3000_load_op_data job.

On the **Folders** tab, this job is in the **Products** ⇨ **SAS Operational Planning for Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the load_op_data transformation, which has the following options:



* OP Cycle Name

* Table holding Dimension and Member codes: WORK.OP_FACT_dimMbrSelection

The sample sas code for this table is in the pre-code section of this transformation

* Deletion of Existing Data

Environment (Optional)

Specify values for the options as follows:

- **OP Cycle Name** is the name of the SAS Financial Management cycle into which you are loading the data.
- **Table holding Dimension and Member codes** is the name of a SAS data set that specifies the member combinations to load data for. The **Precode** region on the **Precode and Postcode** tab contains sample code that builds a SAS data set with the required layout. By default, the job uses the SAS data set that is built by the Precode program. Do one of the following:
  - On the **Precode and Postcode** tab, make sure that the **Precode** check box is selected. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that is built by the precode. Modify the precode to build the table that you need. In this case, the precode runs before the job, and then the job uses the SAS data set that the precode builds.

    - On the **Precode and Postcode** tab, make sure that the **Precode** check box is not selected. Build the SAS data set that you need using a means other than the precode. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that you built for this purpose.

- **Deletion of Existing Data** is either **Replace All** or **Append – No Delete**. Select the correct value from the drop-down list.

  If you select **Replace All**, then data is deleted from all the crossings that you have specified as eligible to receive data. It is possible that data will be deleted from some crossings that do not receive data in the load operation.

  If you select **Append – No Delete**, then no data is deleted. In each case where data is loaded to a crossing that already has data, the result is additive.

- **Environment** can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment "default" is used.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Load Base Operational Data into the Data Mart

To load base operational data using a SAS macro:

1. Create a SAS data set that specifies the member combinations to load data for.
2. Run the etlldopf.sas macro file.

Detailed instructions are inside the macro file, which is on the data-tier server.

On a Windows server, the etlldopf.sas macro file is at the following location: `!SASROOT\finance\sasmacro`. On a UNIX server, the etlldopf.sas macro file is at the following location: `!SASROOT/sasautos`.

The macro can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Which Records Are Loaded?

A staging area record that contains base operational data is loaded only if all of the following conditions are met:

- The record contains a member for each dimension type (except Source) that is used by the target cycle.
- The record contains a member for no dimension type that is not used by the target cycle.
- Within each dimension type that is used by the record and the target cycle, the member in the record belongs to the dimension that is used by the target cycle.
- The record belongs to the subset of records that is defined in the wizard, job, or SAS macro that loads the data. In other words, for each dimension type for which one or more members are specified in the wizard, job, or SAS macro, one of those members is in the record.

Checking for Errors

After you run a job that uses the load_op_data transformation, review the log. If there were errors, then the job is terminated and the log lists the location of an HTML error report. If the SAS macro is terminated, then the log lists the location of an HTML error report. Error reports for the job, the macro, and the Load New Data wizard are all available in SAS Financial Management Studio from the **History** page of the Properties window for the target cycle.

An error report is produced if any record violates any one of the following constraints:

- In the SASOP_DETAIL table, Internal Organization ID must not be ALL or EXT.
- Member IDs are also validated when they are imported into the Data Mart. A data record that has an ID value that does not also exist in the corresponding member table appears in an error report.

Chapter 17

Exporting Financial Accounting Data

Overview of Exporting Accounting Data

You can export data from a selected financial model in SAS Financial Management to a designated SAS library. The target library can be the StageFM library of staging tables or any other library that you set up to receive exported data.

If the target library is StageFM, then the exported data is placed in the following tables:

- GL_TRANSACTION_SUM
- GL_JRNL
- GL_JRNL_DETAILS

If you use any other target library, then the exported data is placed in copies of these staging tables that have been put in the target library.

From the target library, you can make the exported accounting data available to other products, such as SAS Web Report Studio.

In general, you should not export data from a financial model to the staging tables and then load it into a financial cycle. That procedure works, but you can achieve the same result more easily with the Load Model Data wizard in SAS Financial Management Studio.

There are three ways to export data from a selected financial model:

- Use the Export Data Records wizard in SAS Financial Management Studio.
- Use the fm_2000_export_model_data job.
- Use a SAS macro.

Using the Export Data Records Wizard to Export Accounting Data

To export data using the Export Data Records wizard:

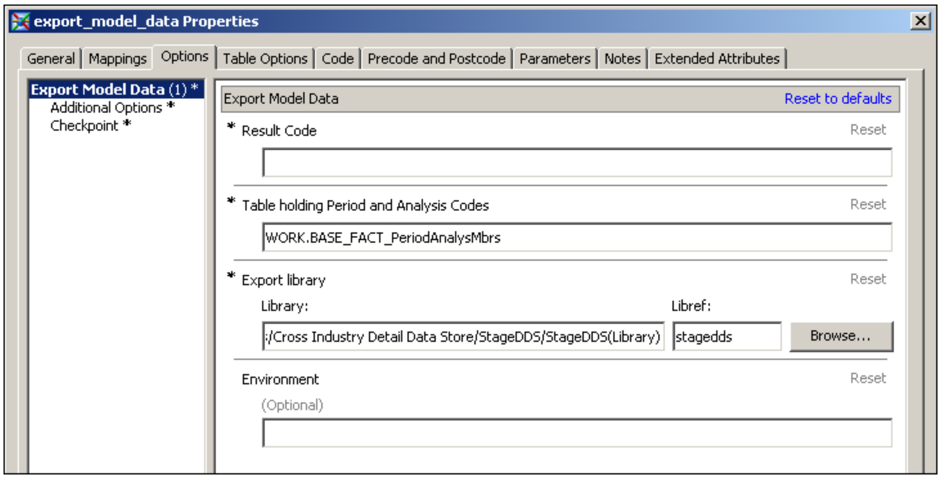
1. In SAS Financial Management Studio, open the cycle from which you want to export data.
2. In the Models workspace, select the source model.
3. Select **Export Data Records** to launch the Export Data Records wizard.
4. Work through the wizard, consulting the online Help for the individual wizard pages as necessary.

Using the Export Model Data Job to Export Accounting Data

To export data using SAS Data Integration Studio, run the fm_2000_export_model_data job.

On the **Folders** tab, this job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the export_model_data transformation, which has the following options:



* Result Code:
* Table holding Period and Analysis Codes: WORK.BASE_FACT_PeriodAnalysMbrs
* Export library
  Library: :/Cross Industry Detail Data Store/StageDDS/StageDDS(Library)
  Libref: stagedds
Environment (Optional):

Provide values for the options as follows:

- **Result Code** is the code of the model that is the source of the data.
- **Table holding Period and Analysis Codes** is the name of a SAS data set that specifies the member combinations to load data for. The **Precode** region on the **Precode and Postcode** tab contains sample code that builds a SAS data set with the required layout. By default, the job uses the SAS data set that is built by the Precode program. Do one of the following:

    - On the **Precode and Postcode** tab, make sure that the **Precode** check box is selected. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that is built by the precode. Modify the precode to build the table that you need. In this case, the precode runs before the job, and then the job uses the SAS data set that the precode builds.
    - On the **Precode and Postcode** tab, make sure that the **Precode** check box is not selected. Build the SAS data set that you need using a means other than the precode. Make sure that the name of the SAS data set that is specified for this option matches the name of the SAS data set that you built for this purpose.
- **Export library** is the SAS library that you are exporting the data to. Click **Browse** to select the target library. If you select a library other than **`StageFM`**, then make sure that the selected library satisfies the following conditions:
    - It contains copies of the GL_TRANSACTION_SUM, GL_JRNL, and GL_JRNL_DETAILS tables.
    - The Solutions Host User has operating system Read and Write access to it.
- **Environment** can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment "default" is used.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Using a SAS Macro to Export Accounting Data

To export accounting data using a SAS macro:

1. Create a SAS data set that specifies the combinations of analysis members and time periods for which you want to export accounting data.
2. Run the etlfctxb.sas macro file.

Detailed instructions are inside the macro file, which is on the data-tier server.

On a Windows server, the etlfctxb.sas macro file is at the following location: **`!SASROOT\finance\sasmacro`**. On a UNIX server, the etlfctxb.sas macro file is at the following location: **`!SASROOT/sasautos`**.

The macro can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Details of the Result

The exported data is appended to whatever data is already in the target tables. If the target tables contain data that you do not want to mix with the data that you are going to export, then you must delete that data from the target tables before you begin the export operation. To delete data from the target tables, write and run a suitable SAS program.

For each specified combination of a time period and an analysis member, the following data is exported:

- All data that is stored in the cycle that the model belongs to. This includes data that is associated with the following members of the Source hierarchy:
  - Base
  - BaseJourn
  - BaseForm
- All manual adjustments and all adjustments that are generated by adjustment rules that are part of the model. This includes data that is associated with the following members of the Source hierarchy:
  - Manual
  - Bal
  - Alloc
  - Reclass
  - CPO

Data that is associated with the BaseJourn member of the Source hierarchy is exported to the GL_JRNL and GL_JRNL_DETAILS tables or to the copies of these tables that you place in another target library. All other exported data is exported to the GL_TRANSACTION_SUM table or to the copy of this table that you place in another target library.

Many numbers that you might see in a SAS Financial Management report that is based on the selected model are not exported. Numbers that are not exported include the following:

- elimination adjustments
- the computed values of accounts that belong to the Retained Earnings and CTA account types
- the computed values of hierarchical roll-ups
- the computed values of formulas

Possible Obstacles to Exporting Accounting Data

The Export Data Records wizard, the fm_2000_export_model_data job, and the etlfctxp.sas macro file can encounter various obstacles that prevent them from successfully exporting data. Possible obstacles include the following:

- The Solutions Host User does not have operating system Read and Write access to the target data library.
- A target table does not exist. If the target data library is the staging area, this can happen if a table was accidentally deleted. For a target data library other than the staging area, this can happen if you neglected to copy one of the necessary tables into the target library.
- A column that represents a dimension type that is used by the data is either misnamed or missing from a target table. This can happen if the column was not added correctly when the dimension type was created.

- The DIMENSION_TYPE table contains an incorrect record for one of the dimension types that is used by the data. This can happen if an incorrect value was placed in the record when it was created.
- One of the target tables is open and locked. This can happen if someone is working with the table.

If any one of these obstacles is encountered, an appropriate message is displayed.

Checking for Errors

After you run a job that uses the export_model_data transformation, review the log. If there were errors, then the job is terminated and the log lists the location of an HTML error report. If the SAS macro is terminated, then the log lists the location of an HTML error report. Error reports for the job, the macro, and the Export Data Records wizard are all available in SAS Financial Management Studio from the **History** page of the Properties window for the source model.

Chapter 18

Loading a System Filter for the SAS Financial Management Add-In for Microsoft Excel

Viewing Data with the SAS Financial Management Add-In for Microsoft Excel

A user who works with a read-only table or a data-entry table in the SAS Financial Management Add-in for Microsoft Excel has many ways to manipulate the table in order to provide different views of the underlying data. For example, a user can do any of the following:

- Pivot the table, turning a row dimension into a column dimension, an unused dimension into a slicer dimension, and so on.
- Change the scope of the displayed slice of data by expanding and collapsing the hierarchies for the row and column dimensions.
- Display a different slice of data by selecting a different member of a slicer dimension.
- Remove certain rows or columns from the display.

Another option that a user might have is to apply or remove a system filter that is loaded through the staging area. To see the details of the available system filter, a user of the SAS Financial Management Add-in for Microsoft Excel selects **Filters**. If a system filter is available, it is displayed on the **System** tab of the Filters window. By selecting or deselecting the **Enable system filters** check box on that tab, a user can apply or remove the available system filter.

Users of the SAS Financial Management Add-in for Microsoft Excel have a system filter available only if a system filter is loaded through the staging area as explained in this chapter.

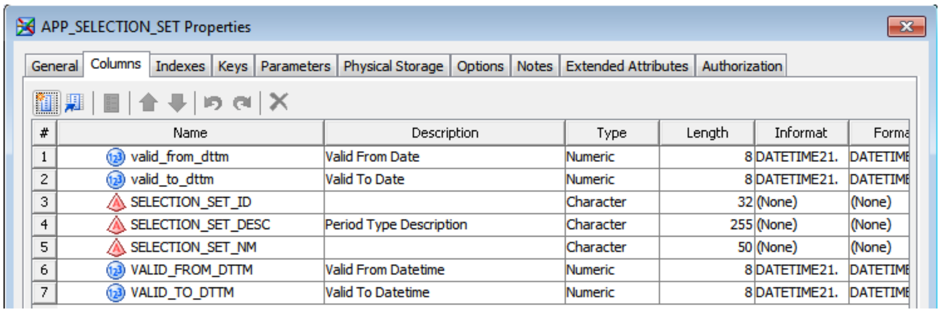
Loading System Filter Specifications into the Staging Tables

The following two staging tables are used to define a system filter:

- APP_SELECTION_SET
- APP_MBR_SELECTION_RULES

Write and run a job to load each of these tables.

Use the APP_SELECTION_SET table to define a selection set:



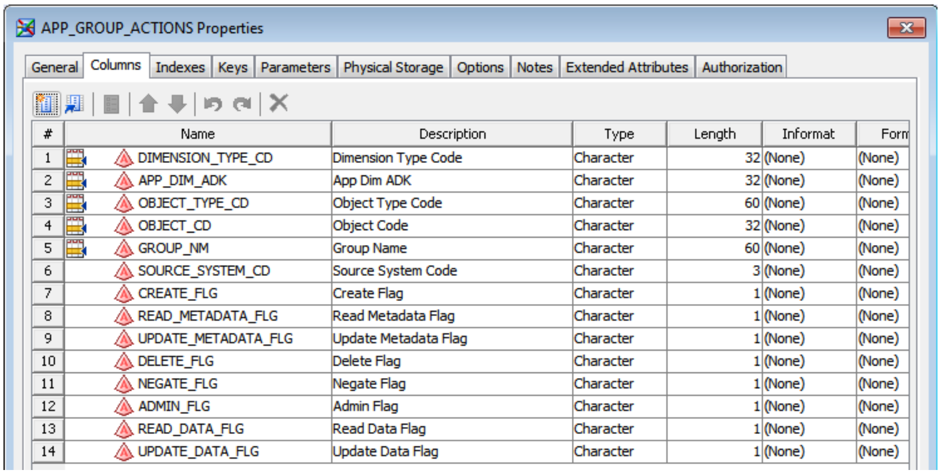
#	Name	Description	Type	Length	Informat	Forma
1	valid_from_dttm	Valid From Date	Numeric	8	DATETIME21.	DATETIME
2	valid_to_dttm	Valid To Date	Numeric	8	DATETIME21.	DATETIME
3	SELECTION_SET_ID		Character	32	(None)	(None)
4	SELECTION_SET_DESC	Period Type Description	Character	255	(None)	(None)
5	SELECTION_SET_NM		Character	50	(None)	(None)
6	VALID_FROM_DTTM	Valid From Datetime	Numeric	8	DATETIME21.	DATETIME
7	VALID_TO_DTTM	Valid To Datetime	Numeric	8	DATETIME21.	DATETIME

In building a record, note the following:

- Valid From Datetime and Valid To Datetime define the lifespan of the record.
- Selection Set ID is used in every record of the APP_MBR_SELECTION_RULES table to indicate which selection set the record belongs to.

Note: The APP_SELECTION_SET_NLS table has no use. Ignore it.

Use the APP_MBR_SELECTION_RULES table to specify the details of the system filter:



#	Name	Description	Type	Length	Informat	Form
1	DIMENSION_TYPE_CD	Dimension Type Code	Character	32	(None)	(None)
2	APP_DIM_ADK	App Dim ADK	Character	32	(None)	(None)
3	OBJECT_TYPE_CD	Object Type Code	Character	60	(None)	(None)
4	OBJECT_CD	Object Code	Character	32	(None)	(None)
5	GROUP_NM	Group Name	Character	60	(None)	(None)
6	SOURCE_SYSTEM_CD	Source System Code	Character	3	(None)	(None)
7	CREATE_FLG	Create Flag	Character	1	(None)	(None)
8	READ_METADATA_FLG	Read Metadata Flag	Character	1	(None)	(None)
9	UPDATE_METADATA_FLG	Update Metadata Flag	Character	1	(None)	(None)
10	DELETE_FLG	Delete Flag	Character	1	(None)	(None)
11	NEGATE_FLG	Negate Flag	Character	1	(None)	(None)
12	ADMIN_FLG	Admin Flag	Character	1	(None)	(None)
13	READ_DATA_FLG	Read Data Flag	Character	1	(None)	(None)
14	UPDATE_DATA_FLG	Update Data Flag	Character	1	(None)	(None)

In building records for this table, note the following:

- Selection Set ID indicates which selection set the record belongs to. Every record must belong to the single selection set that is defined in the APP_SELECTION_SET table.

- Source App Dim ID is the code of the controlling dimension.
- Source Member ID is the code of the controlling member in the controlling dimension.
- Target App Dim ID is the code of the controlled dimension.
- Target Member ID is the code of the controlled member in the controlled dimension.
- For each record, its *control condition* is satisfied by a read-only table or a data-entry table whenever one of the following is true:
  - The controlling dimension is a slicer dimension, the controlling member is selected in that slicer dimension, and the controlled dimension is either a slicer, column, or row dimension.
  - The controlling dimension is a column dimension, the controlling member is displayed in the table, and the controlled dimension is also a column dimension.
  - The controlling dimension is a row dimension, the controlling member is displayed in the table, and the controlled dimension is also a row dimension.

  If a read-only table or a data-entry table does not satisfy a record's control condition, then the record has no effect on the table.
- **Include Member Flag** must contain either N or Y, or a character 0 or 1. These values have the following significance:
  - N or 0—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter removes the controlled member from the table.
  - Y or 1—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter does not remove the controlled member from the table. (A user can still remove this member in other ways.)
- **Include Leaves Filter No** must contain a numeric 0, 1, or 2. These values have the following significance:
  - 0—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter removes from the table all leaf members that are descendants of the controlled member (children, grandchildren, and so on).
  - 1—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter removes from the table all leaf members that are remote descendants of the controlled member (grandchildren and beyond, but not children).
  - 2—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter does not remove from the table any leaf members that are descendants of the controlled member.
- **Include Rollups Filter No** must contain a numeric 0, 1, or 2. These values have the following significance:
  - 0—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter removes from the table all non-leaf members that are descendants of the controlled member (children, grandchildren, and so on).
  - 1—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter removes from the table all non-leaf members that are remote descendants of the controlled member (grandchildren and beyond, but not children).

- 2—If a read-only table or a data-entry table satisfies the record's control condition, then the system filter does not remove from the table any non-leaf members that are descendants of the controlled member.

- There are many ways in which a member of a controlled dimension can be subject to conflicting filter specifications. For example, a record with a 0 in the Include Member Flag column can specify that the member should be removed from a read-only table or a data-entry table while another record with a 1 or 2 in one of the Filter No columns can specify that the same member should not be removed from the table.

  All such conflicts are resolved according to the following simple rule. The behavior of a given member is determined by the record whose controlled member is closest to the given member in the relevant hierarchy. A record in which the given member is the controlled member always prevails, if it exists. If there is no record in which the given member is the controlled member, then a record in which the immediate parent of the given member is the controlled member always prevails, if it exists. And so on, up the hierarchy.

  This method of conflict resolution enables you to define complex filter patterns by layering a sequence of specifications. You can specify one filter condition for a certain subhierarchy, partially override it with a different filter condition for a lower subhierarchy, and partially override the override with yet another filter condition for an even lower subhierarchy.

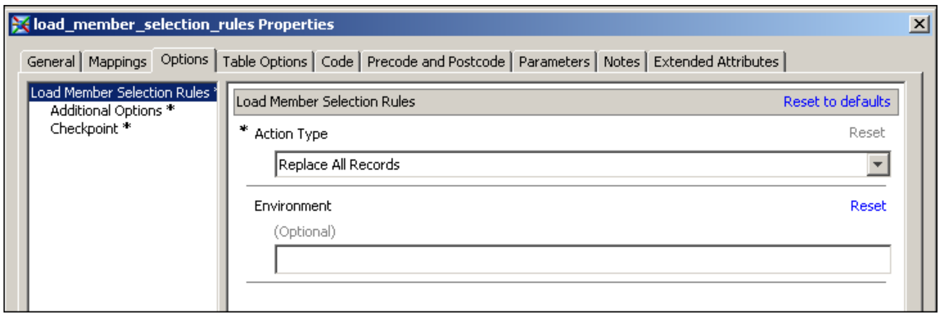
- The order of the records in the APP_MBR_SELECTION_RULES table does not affect the behavior of the resulting system filter. However, you can make the table easier to maintain by grouping the records in a logical way. For example, if you are defining a filter that has more than one pair of controlling and controlled dimensions, then it makes sense to group all the records for a given pair of dimensions together.

Moving System Filter Specifications from the Staging Area to the Data Mart

To load system filter specifications into the Data Mart, run the fm_1400_load_member_selection_rules job.

On the **Folders** tab, this job is in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder.

The job consists of the load_member_selection_rules transformation, which has the following options:



* Action Type: Replace All Records

Environment (Optional):

Provide values for the options as follows:

- **Action Type** is either **Replace All Records** or **Delete All Records**. Select the correct value from the drop-down list:
  - **Replace All Records**—Delete all the system filter records that are currently in the Data Mart, and then load into the Data Mart all the system filter records that are currently in the staging area. You must use this value if you are loading filter records.
  - **Delete All Records**—Delete all system filter records that are currently in the Data Mart, but do not load any new records. This value enables you to get rid of a previously loaded set of filter records, leaving the SAS Financial Management Add-in for Microsoft Excel without a system filter.
- **Environment** can be any environment that is defined in your EnvironmentFactory.xml file. If you leave this field empty, then the environment “default” is used.

The job can run only if SAS Remote Services and the managed servers are running on the middle-tier server. See *SAS Financial Management: System Administration Guide*.

Chapter 19
Loading Supplemental Schedule Detail and Fact Tables

About Loading Supplemental Schedule Detail and Fact Tables

A supplemental schedule is an additional table that can be added to enable users to reference detailed information outside the financial model for use in reports, forms, and so on. There are two staging tables for supplemental schedule tables:

- SUPP_SCHEDULE_DETAIL
- SUPP_SCHEDULE_FACT

Load the Supplemental Schedule Detail Table

1. On the **Folders** tab, in the **Products** ⇨ **SAS Financial Management** ⇨ **5.3 Jobs** folder, make a copy of the fm_2200_load_ss_detail job.
2. Double-click the job to display its process diagram.
3. In the process diagram, select the load_ss_detail transformation, and then select **Properties** from the pop-up menu.
4. In the Properties window, select the **Options** tab.
5. Provide a value for the **Cycle Name** option. For more information about cycles, see the *SAS Financial Management: User's Guide*.
6. Save the job.
7. Run the job and then review the log.

Load the Supplemental Schedule Fact Table

1. On the **Folders** tab, in the **Products ⇨ SAS Financial Management ⇨ 5.3 Jobs** folder, make a copy of the fm_2210_load_ss_fact job.
2. Double-click the job to display its process diagram.
3. In the process diagram, select the transformation, and then select **Properties** from the pop-up menu.
4. In the Properties window, select the **Options** tab.
5. Provide a value for the **Cycle Name** option. For more information about cycles, see the *SAS Financial Management: User's Guide*.
6. Save the job.
7. Run the job and then review the log.

Part 2

Appendixes

Appendix 1

MySQL Reserved Words

MySQL reserved words cannot be used as column names of MySQL tables. Therefore, you must not use any MySQL reserved word as a dimension code for a dimension that will be represented by a column in a metric table. The safest approach is to avoid defining any dimension code that is identical to a MySQL reserved word.

There is a complete list of MySQL reserved words at the following Web location:

`http://dev.mysql.com/doc/refman/5.0/en/reserved-words.html`

For a discussion of dimension codes, see Chapter 6, “Creating a Dimension,” on page 23.

Appendix 2

The Conform Area

Overview of the Conform Area

Data that passes through the SAS Financial Management staging area can go from the SAS Financial Management staging area to a data mart either directly or by way of an intermediate location known as the conform area. The conform area and the SAS Financial Management staging area are concatenated to constitute the CONFORM library. Jobs that load data into a data mart refer to input tables in the CONFORM library. By default, the conform area points to the same location as the SAS Financial Management staging area:

```
..Lev1\SASApp\Data\FinancialManagement\StageFM
```

To create a separate conform area, prepend the following path in the LIBNAME statement:

```
..\Lev1\SASApp\Data\FinancialManagement\ConformedDataMart
```

The ConformedDataMart folder path must be prepended to the stageFM folder path. Do not be misled by the name of the conform area directory. The conform area is not a destination data mart.

The following LIBNAME statement is an example of creating a separate conform area:

```
   LIBNAME Conform BASE
   ("C:\SAS\Config\Lev1\SASApp\Data\FinancialManagement\ConformedDataMart"
   "C:\SAS\Config\Lev1\SASApp\Data\FinancialManagement\StageFM");
```

Copying Tables to the Conform Area

Using the copy_all_stagefm_tables_to_conform_library and copy_stagefm_table_to_conform_library transformations in the Solutions Transformations folder on the **Transformations** tab, you can create jobs that copy tables to the conform area. You can recopy the tables at any time. Once you copy these tables to the conform area, the SAS Financial Management Data Mart is loaded from the most recently copied versions of these tables. If you never copy these tables to the conform area, then the SAS Financial Management Data Mart is loaded from the tables in the SAS Financial Management staging area.

Index