Siebel Analytics Enterprise Data Warehouse Implementation Guide

Version 7.7.2, Rev. A
September 2004

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

Contents

Chapter 4: Initializing and Populating the Siebel Analytics Enterprise Data Warehouse

Chapter 5: Planning Your Warehouse Configuration Project

Chapter 9: Configuring the Receivables Module

Chapter 10: Configuring the Profitability Module

Chapter 11: Configuring the Sales Module

Chapter 12: Configuring the Service Module

Chapter 13: Configuring the Sourcing Module

Chapter 14: Configuring the Compensation Module

Chapter 15: Configuring the Workforce Management Operations and Workforce Management Retention Module

Chapter 16: Performing Cross-Module Configuration

Index

1 What's New in This Release

What's New in Siebel Analytics Enterprise Data Warehouse Implementation Guide, Version 7.7.2, Rev. A

Table 1 lists changes described in this version of the documentation to support release 7.7.2 of the software.

Table 1. What's New in Siebel Analytics Enterprise Data Warehouse Implementation Guide, Version 7.7.2, Rev. A

Topic	Description
Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26	Added a process that describes how to install PowerCenter and Siebel Analytics Enterprise Data Warehouse.
Process of Creating the Development Repository on page 28	Added a process that guides you through creating a Siebel Analytics Enterprise development repository.
Process of Configuring the General Ledger Module for SAP R/3 on page 79	Added a process that explains how to configure the General Ledger module for SAP R/3.
Configuring Payables Module for SAP R/3 on page 97	Added information that explains how to configure the Payables module for SAP R/3.
Configuring Receivables for SAP R/3 on page 105	Added information that explains how to configure the Receivables module for SAP R/3.
Process of Configuring Sales for Post-Load Processing	Added information that explains how to configure the Sales module for Universal Source.
Process of Configuring Sourcing for a Universal Source on page 176 and Configuring Sourcing for Post-Load Processing on page 182	Added information that explains how to configure the Sourcing module for Universal Source and post-load processing.
Chapter 15, "Configuring the Workforce Management Operations and Workforce Management Retention Module"	Added a chapter that gives an overview of the Workforce Management Operations and Workforce Management Retention module and explains how to configure the module for Oracle 11i.
Chapter 14, "Configuring the Compensation Module"	Added a chapter that gives an overview of the Compensation module and explains how to configure the module for PeopleSoft 7.5 and Oracle 11i.

Table 1. What's New in Siebel Analytics Enterprise Data Warehouse Implementation Guide, Version 7.7.2, Rev. A

Topic	Description
Chapter 17, "Storing, Extracting, and Loading Additional Data"	Added a chapter that discusses the methodology for storing additional data in the data warehouse. This chapter also provides general procedures for extracting and loading new data.
Chapter 18, "Integrating Additional Data"	Added a chapter that provides procedural information for creating and modifying the Extract, Transform and Load (ETL), and the Post-Load Processing (PLP) components.

2 Overview of Implementing the Siebel Analytics Enterprise Data Warehouse

To implement the Siebel Analytics Enterprise Data Warehouse follow the process of installing, populating, and configuring as discussed in the subsequent chapters.

This chapter contains the following topics:

Installing the Siebel Analytics Enterprise Data Warehouse

To install the Siebel Analytics Enterprise Data Warehouse, perform the following tasks:

- Determine your Analytics system configuration, including the expected rate of growth for your data warehouse. For more information, see the following section in Chapter 3, "Installing the Siebel Analytics Enterprise Data Warehouse":

  Process of Defining Your Installation Configuration on page 19

- Set up system infrastructure, including servers, databases, users, and so on, in your development environment. This step includes installing PowerCenter and all patches. For more information, see the following sections in Chapter 3, "Installing the Siebel Analytics Enterprise Data Warehouse":

  - Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26
  - Process of Creating the Development Repository on page 28

- Populate the Siebel Analytics Enterprise Data Warehouse repository with the ETL objects required for your data warehouse. For more information, see Initializing and Populating the Siebel Analytics Enterprise Data Warehouse on page 14.
- Set up the Siebel Analytics Enterprise Data Warehouse tables. For more information, see Configuring the Siebel Analytics Enterprise Data Warehouse on page 15.

Figure 1 illustrates the tasks involved in installing the Siebel Analytics Enterprise Data Warehouse, that are discussed above.

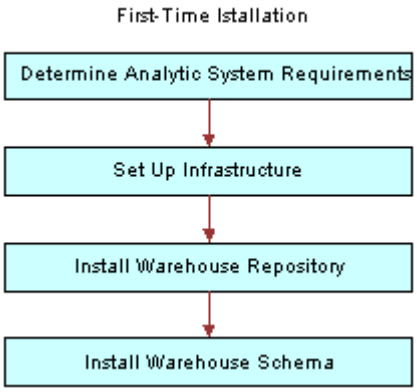


Figure 1. Implementing the Siebel Analytics Enterprise Data Warehouse: Install Warehouse Phase

For more information on installing the Siebel Analytics Enterprise Data Warehouse, see Chapter 3, "Installing the Siebel Analytics Enterprise Data Warehouse." This chapter covers the tasks to be completed before and while installing the Siebel Analytics Enterprise Data Warehouse.

Initializing and Populating the Siebel Analytics Enterprise Data Warehouse

After you have installed the Siebel Analytics Enterprise Data Warehouse, the next step is to load the data and see if it loads properly.

To populate the Siebel Analytics Enterprise Data Warehouse, perform the following tasks:

- **Initialize the Siebel Analytics Enterprise Data Warehouse.** This task includes loading prepackaged data provided by Siebel Analytics Enterprise Applications.
- **Load data.** This task includes running all workflows to populate the data warehouse with source data.

See Figure 2 for an illustration on the Initialization of the Siebel Analytics Enterprise Data Warehouse followed by the loading of the data.

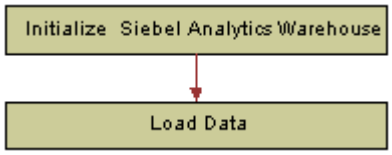


Figure 2. Implementing the Siebel Analytics Enterprise Data Warehouse: Populating the Warehouse

For more information on initializing and populating the Siebel Analytics Enterprise Data Warehouse, see Chapter 4, "Initializing and Populating the Siebel Analytics Enterprise Data Warehouse." This chapter outlines tasks needed to prepare your data warehouse to perform your initial data capture.

Configuring the Siebel Analytics Enterprise Data Warehouse

When the data warehouse is loaded, you must then verify that the data is loaded according to your expectations. For example, Siebel Analytics Enterprise Data Warehouse only loads booked orders for the Sales Order Lines table IA_SALES_ORDLNS; it filters out all other information, such as nonbooked orders. However, you may want nonbooked orders in the Sales Order Lines table as well. Therefore, you would need to configure a mapping to reset the loading logic for this table.

To configure the Siebel Analytics Enterprise Data Warehouse, perform the following tasks:

- **Perform Gap Analysis.** This task includes comparing what is actually stored in the data warehouse against what you expect to be stored.
- **Configure Repository Objects.** This task includes changing the way in which data is loaded to better meet your business requirements. It also includes resetting the run-time for any of the sessions, worklets, or workflows.
- **Reload and Validate Data.** After all of your objects are configured, you must perform a check on the source data to verify that no data is missing. If data is missing, your reports can be inaccurate. Therefore, you must run the extract mappings and then check the staging tables to verify that all columns are populated.

See Figure 3 for an illustration of this entire phase of performing gap analysis, configuring the repository object, and reloading and validating the data. The configuration continues until the data is loaded as expected.

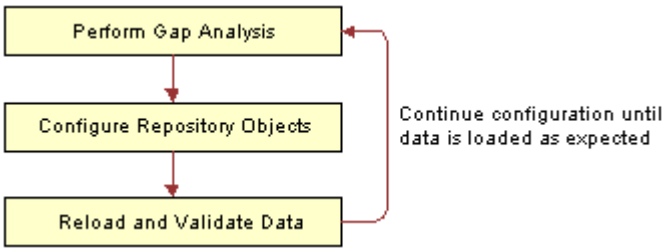


Figure 3. Implementing the Siebel Analytics Enterprise Data Warehouse: Configuring Phase

Table 2 describes the chapters in this guide that provide more detail on he configuration phase of the Siebel Analytics Enterprise Data Warehouse implementation.

Table 2. Chapters Discussing the Configuration Phase

Chapter	Description
Chapter 5, "Planning Your Warehouse Configuration Project"	This chapter provides the methodology for comparing prepackaged the Siebel Analytics Enterprise Data Warehouse repository objects with your business organization's needs.
Chapter 6, "Configuring the General Ledger Module"	This chapter provides configuration information about the General Ledger module.
Chapter 7, "Configuring the Inventory Module"	This chapter provides configuration information information about the Inventory module.
Chapter 8, "Configuring the Payables Module"	This chapter provides configuration information about the Payables module.
Chapter 9, "Configuring the Receivables Module"	This chapter provides configuration information about the Receivables module.
Chapter 10, "Configuring the Profitability Module"	This chapter provides configuration information about the Profitability module.
Chapter 11, "Configuring the Sales Module"	This chapter provides configuration information about the Sales module.
Chapter 12, "Configuring the Service Module"	This chapter provides configuration information about the Service module.
Chapter 13, "Configuring the Sourcing Module"	This chapter provides configuration information about the Sourcing module.
Chapter 15, "Configuring the Workforce Management Operations and Workforce Management Retention Module"	This chapter provides configuration information about the Workforce Management Operations and Workforce Management Retention module.
Chapter 14, "Configuring the Compensation Module"	This chapter provides configuration information about the Compensation module.

Additional Resources

The following documentation contains information that may be relevant to your use of Siebel Analytics Enterprise Applications.

- For more information about the system requirements and supported platforms, see the *System Requirements and Supported Platforms* on Siebel SupportWeb.
- For more information about the individual modules, see *Siebel Analytics Enterprise Applications User Guide*.

- For more information about the installation and configuration tasks related to Siebel Analytics, see *Siebel Analytics Installation and Configuration Guide*.
- For more information about PowerCenter and PowerMart installation and requirements, see the PowerCenter/PowerMart *Installation QuickStart Guide* and the PowerCenter/PowerMart *Installation and Configuration Guide*.
- For a list of all domain values, see *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

3 Installing the Siebel Analytics Enterprise Data Warehouse

This chapter describes the tasks you must complete before and while installing the Siebel Analytics Enterprise Data Warehouse.

NOTE: Siebel Analytics must be installed before the Siebel Analytics Enterprise Data Warehouse. For information on installing Siebel Analytics, see the *Siebel Analytics Installation and Configuration Guide*.

This chapter contains the following topics:

Process of Defining Your Installation Configuration

Before installing any software, you must first define your installation configuration and related infrastructure requirements. To define your installation, perform the following tasks:

Determining Configuration Requirements

To determine configuration requirements, you must identify the Siebel Applications components to be installed and plan their configuration in your environment. Some factors that determine your installation configuration are:

- Hardware costs
- Operating system used to run Siebel Applications
- Number of rows being extracted from the source

- Number of PowerCenter users

The configuration must specify:

- **Connection information for source machines.** Specify information about the network addresses of the source system from which data needs to be extracted, any associated constraints, ODBC connect strings, and so on.
- **Physical location of the server.** Identify the specific machines and their IP addresses, disk sizes, CPUs, memory, operating system, and so on.
- **Physical location of the databases.** Specify the database to be configured on the servers. The databases are used to house the data warehouse, staging tables, control tables, and so on.

You must also make sure the following in relation to the physical location of the databases:

- It is recommended that you locate the staging area and the data warehouse in a single database with one user ID. This configuration makes it easier to create outer joins between the staging area and data warehouse tables, without needing multiple synonyms.
- If you do create a separate user for the staging tables and the data warehouse, your staging area user must have select privileges on all the data warehouse objects. Create the appropriate grant and synonym creation scripts necessary for this purpose. To minimize input or output contention and improve performance, it is recommended that you create indexes in a separate tablespace.

Gathering PowerCenter Information

This task is a step in the Process of Defining Your Installation Configuration on page 19.

While you are preparing to install the Siebel Analytics Enterprise Data Warehouse, you must gather information for your databases and repositories, such as passwords, IDs, and so on. Siebel applications provide variables with predefined designations to act as placeholders for the information needed throughout this guide. The variables begin with the dollar symbol ($), followed by a name that represents the placeholder role. For example, the PowerCenter Server requires a password for the UNIX user ID: `$PC_SVR_UNIX_PASSWORD`.

Table 3 lists the variable names for installing the PowerCenter. You can use this table as a worksheet, and record the values applicable to your installation.

Table 3. Variable Names with Explanations for Installing PowerCenter

Variable	Explanation	Your Value
PowerCenter Server Parameters for Windows NT		
$PC_SVR_NT_USERID	The NT user ID with administrative access to PowerCenter and, if you are extracting from SAP, to PowerConnect.	
$PC_SVR_NT_PASSWORD	The password for the user ID to use to install PowerCenter.	

Table 3. Variable Names with Explanations for Installing PowerCenter

Variable	Explanation	Your Value
$PC_SVR_NT_DOMAIN	The NT domain on which PowerCenter is to be installed.	
PowerCenter Server Parameters for UNIX		
$PC_SVR_UNIX_USERID	The user ID that has administrative access to PowerCenter and, if you are extracting from SAP, administrative access to PowerConnect.	
$PC_SVR_UNIX_PASSWORD	The password to the UNIX user ID.	
PowerCenter Client Parameters		
$PC_CLIENT_USERID	The user ID that has administrative access to PowerCenter and, if you are extracting from SAP, to PowerConnect.	
$PC_CLIENT_PASSWORD	The password for the user ID (to be used to install PowerCenter).	
$PC_CLIENT_DOMAIN	The NT domain on which PowerCenter is to be installed.	
PowerCenter Repository Server Parameters		
$PC_REP_SVR_DOMAIN	The domain for the user account that runs the Repository Server service (must have the Logon as a service right).	
$PC_REP_SVR_USERNAME	The username for the user account that runs the Repository Server service.	
$PC_REP_SVR_PASSWORD	The password for the user account that runs the Repository Server service.	
$PC_REP_SVR_PORT_NUMBER	The port number the Repository Server uses to connect to the repository client applications. The default port number is 5001.	
$PC_REP_SVR_PORT_MAX	The maximum port number the Repository Server can assign to the Repository Agent process. The default value is 65535.	
$PC_REP_SVR_PORT_MIN	The minimum port number the Repository Server can assign to the Repository Agent process. The default value is 5002.	
PowerCenter Repository Parameters		
$PC_TEMP_REPO_NAME	The name you want to assign to your first, or temporary PowerCenter Repository (IA_TEMP).	

Table 3. Variable Names with Explanations for Installing PowerCenter

Variable	Explanation	Your Value
$PC_TEMP_REPO_USERID	The database user ID to connect to the database housing the first, or temporary PowerCenter Repository (IA_TEMP).	
$PC_TEMP_REPO_PASSWORD	The database password to connect to the database housing the first, or temporary PowerCenter Repository (IA_TEMP).	
$PC_TEMP_REPO_SERVER	The server housing the first, or temporary PowerCenter Repository (IA_TEMP).	
$PC_TEMP_REPO_DATABASE	The database where your first, or temporary PowerCenter Repository (for IA_TEMP) resides.	
$PC_TEMP_CODE_PAGE	The code page specific to the characters and language your system is using.	
$PC_CONTENT_INSTALL_DIR	The install directory you choose when you first install the PowerCenter content from the installation CD (when you run SETUP.EXE). By default, the content files are installed on C:\Program Files\Informatica.	
$PC_DEV_REPO_NAME	The name you want to assign to your development repository (IA_DEV).	
$PC_DEV_REPO_USERID	The user ID to connect to the database housing the PowerCenter development repository for IA_DEV.	
$PC_DEV_REPO_PASSWORD	The password to connect to the database housing the PowerCenter development repository for IA_DEV.	
$PC_DEV_REPO_SERVER	The server housing the PowerCenter development repository for IA_DEV.	
$PC_DEV_REPO_DATABASE	The database where your development repository (for IA_TEMP) resides.	
$PC_DEV_CODE_PAGE	The code page specific to the characters and language your system is using.	
$PC_QA_REPO_NAME	The name that you want to assign to your QA repository (IA_QA).	
$PC_QA_REPO_USERID	The user ID to connect to the database housing the PowerCenter QA repository (IA_QA).	

Table 3. Variable Names with Explanations for Installing PowerCenter

Variable	Explanation	Your Value
$PC_QA_REPO_PASSWORD	The password to connect to the database housing the PowerCenter QA repository (IA_QA).	
$PC_QA_REPO_SERVER	The server hosting the PowerCenter QA repository (IA_QA).	
$PC_QA_REPO_DATABASE	The database where your QA repository (for IA_QA) resides.	
$PC_QA_CODE_PAGE	The code page specific to the characters and language your system is using.	
$PC_PROD_REPO_NAME	The name you want to assign to your production repository (IA_PROD).	
$PC_PROD_REPO_USERID	The user ID to connect to the database housing the PowerCenter production repository (for IA_PROD).	
$PC_PROD_REPO_PASSWORD	The password to connect to the database housing the PowerCenter production repository (for IA_PROD).	
$PC_PROD_REPO_SERVER	The server housing the PowerCenter production repository (for IA_PROD).	
$PC_PROD_REPO_DATABASE	The database where your production repository for IA resides (IA_PROD).	
$PC_PROD_CODE_PAGE	The code page specific to the characters and language your system is using.	
PowerCenter Server License Information		
Platform Key	You are asked to enter the Platform Key during your installation of PowerCenter. This information is provided with your installation CDs.	
Database Key	This key varies with each RDBMS, so the Oracle key is different from the key you enter if you are running Sybase. This information is provided with your installation CDs.	
Siebel Applications and Siebel Analytics Enterprise Data Warehouse License Information		
Siebel Applications and Siebel Analytics Enterprise Data Warehouse Key	You are asked to enter the serial number (key) during your installation of Siebel Applications or Siebel Analytics Enterprise Data Warehouse. This information is provided with your installation CDs.	

Gathering Information About Your Data Warehouse and Staging Tables

This task is a step in the Process of Defining Your Installation Configuration on page 19.

You must gather information for the data warehouse and the staging tables.

Table 4 lists information about the data warehouse and staging tables. You can use this table as a worksheet and record the values applicable to your installation.

Table 4. Information About Your Data Warehouse and Staging Tables

Variable	Explanation	Your Value
STAGING AREA		
$IA_SA_USERID	The database user ID to connect to the database housing the staging tables. It is recommended that this ID be the same user as for the data warehouse, $IA_DW_USERID. If this ID is different from the data warehouse user, this user must have select privileges on all the data warehouse objects. Create the appropriate grant and synonym creation scripts necessary for this purpose.	
$IA_SA_PASSWORD	The password to connect to the database housing the staging tables. This password must be the same as the $IA_DW_PASSWORD password.	
$IA_SA_DW_SERVER	The server housing the data warehouse and staging area databases (preferably the same for both).	
$IA_SA_DATABASE	The database where the staging area and data warehouse tables reside (must be the same as $IA_DW_DATABASE).	
DATA WAREHOUSE		
$IA_DW_USERID	The database user ID to connect to the database housing the data warehouse tables. It is recommended that this user ID is the same as the user ID for the data warehouse, $IA_SA_USERID.	
$IA_DW_PASSWORD	The password to connect to the database housing the data warehouse tables. It is recommended that this password is the same as $IA_SA_PASSWORD.	

Table 4. Information About Your Data Warehouse and Staging Tables

Variable	Explanation	Your Value
$IA_SA_DW_SERVER	The server housing the data warehouse and staging area databases (preferably the same for both).	
$IA_DW_DATABASE	The database where the data warehouse tables reside (must be the same as $IA_SA_DATABASE).	

Determining Source Requirements

Some source systems and database platforms require additional steps, as detailed in this section. To install PowerConnect for specific source systems, see the relevant PowerConnect *User and Administrator Guide*.

NOTE: This section applies to customers using universal business adapters or SAP R/3.

Using the Universal Business Adapter

For source systems without prepackaged business adapters, you must provide flat file extracts. The flat files must conform to a template (located in the directory `$Siebel\ETL\Siebel Warehouse Support Files\SrcFiless`). Siebel Analytics Enterprise Applications use the Universal Interface document to process and load the data into the data warehouse.

Using the SAP R/3 Source System

This section describes how to customize your Siebel Analytics Enterprise Data Warehouse environment for a SAP R/3 source system.

Creating a Profile

The SAP R/3 administrator must create a profile in the R/3 system that allows you to use the integration features. This profile must include authorization for the objects and related activities listed in Table 5.

Table 5. SAP R/3 Integration Features

Integration Feature	Authorization Object	Activity
Import definitions, install programs	[S_DEVELOP]	All activities Also must set Development Object ID to PROG
Extract data	[S_TABU_DIS]	READ
Run file mode sessions	[S_DATASET]	WRITE
Submit background job	[S_PROGRAM]	BTCSUBMIT, SUBMIT

Table 5. SAP R/3 Integration Features

Integration Feature	Authorization Object	Activity
Release background job	[S_BTCH_JOB]	DELE, LIST, PLAN, SHOW Also must set Job Operation to RELE
Run stream mode sessions	[S_CPIC]	All activities

Generating ABAP Code

If you are extracting from SAP R/3, you must generate the ABAP code from the extract mappings. The mappings used to extract data from SAP R/3 all follow this naming convention, M_S_*_EXTRACT. For the mappings whose names fall into this category, you generate the ABAP codes. For more information on generating the ABAP codes, see the *PowerConnect for SAP R/3 User and Administrator Guide*.

Creating the Database Tables

Contact your system administrator to create your database requirements. You need six table spaces on your database for the following uses:

- Siebel Analytics Enterprise Data Warehouse
- Staging Area
- Siebel Analytics Enterprise Data Warehouse Repository
- Indexes
- Temporary Tables
- Schema Control Tables

Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse

This section describes how to install PowerCenter and the Siebel Analytics Enterprise Data Warehouse. PowerCenter is included on its own CD, whereas the Siebel Analytics Enterprise Data Warehouse is included on the Siebel Analytics Enterprise Applications CD. To install the required products, perform the following tasks:

- Installing PowerCenter on page 27
- Installing the Siebel Analytics on page 27
- Installing the Siebel Analytics Enterprise Data Warehouse on page 27
- Moving the Siebel Analytics Enterprise Data Warehouse Repository on page 27

Installing PowerCenter

This task is a step in the Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26.

Before installing any other product, you must install your basic PowerCenter platform. For more information about PowerCenter and the PowerMart installation and requirements, see the PowerCenter/PowerMart *Installation QuickStart Guide* and the PowerCenter/PowerMart *Installation and Configuration Guide*.

Installing the Siebel Analytics

This task is a step in the Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26.

You must install Siebel Analytics before you install the Siebel Analytics Enterprise Data Warehouse. For the supported version of Siebel Analytics for Siebel Analytics Enterprise Applications, see the *System Requirements and Supported Platforms* on Siebel SupportWeb. For more information about installing Siebel Analytics, see the *Siebel Analytics Installation and Configuration Guide*.

Installing the Siebel Analytics Enterprise Data Warehouse

This task is a step in the Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26.

This task copies the Repository, Web Catalog, and ETL folders and files to your machine.

To install the Siebel Analytics Enterprise Data Warehouse

1. Navigate to the Siebel Analytics Enterprise Data Warehouse ZIP file.
2. Extract the SiebelBusinessAnalytics.zip file to your hard drive.

   A SiebelBusinessAnalytics folder is created along with the ETL, Repository, and WebCatalog subfolders.

Moving the Siebel Analytics Enterprise Data Warehouse Repository

This task is a step in the Process of Installing PowerCenter and the Siebel Analytics Enterprise Data Warehouse on page 26.

You must move the Shell.rep file to your PowerCenter Repository Server backup directory before you can complete the next procedure. The shell repository is a placeholder for the PowerCenter Repository Server. This shell repository is an empty repository that contains the folder structure, and the relational and application connections. When you have installed your required Siebel Applications components, place this file in the backup directory to restore the latest Siebel Analytics Enterprise Data Warehouse repository. For information on how to restore the latest Siebel Analytics Enterprise Data Warehouse repository, see Creating and Configuring an Empty Siebel Analytics Enterprise Data Warehouse Repository on page 29.

Process of Creating the Development Repository

The following section guides you through restoring the prepackaged Siebel Analytics Enterprise Data Warehouse repository and creating a development repository with just the contents applicable to your data warehouse.

To create a development repository, perform the following tasks:

- Creating and Configuring an Empty Siebel Analytics Enterprise Data Warehouse Repository on page 29
- Registering Packages for the SAP R/3 Repository on page 30
- Modifying the Repository and Repository Server on page 31
- Importing Module Workflows into the Repository on page 31
- Backing Up the Development Repository on page 32
- Copying the Required Data Files on page 32
- Creating the Data Warehouse Schema Control Tables on page 33
- Deploying Stored Procedures on page 34
- Creating the Seed Data on page 34

Related Topic

- Modifying the Rollback Segments on page 28

Modifying the Rollback Segments

The PowerCenter repository population process performs a single, logged transaction. Because this single transaction is extremely large, you must configure your rollback segments to return your database to its original state if the install transaction fails for any reason.

For information on modifying rollback segments, contact your database administrator, or read the documentation for your specific database.

Creating and Configuring an Empty Siebel Analytics Enterprise Data Warehouse Repository

This task is a step in the Process of Creating the Development Repository on page 28.

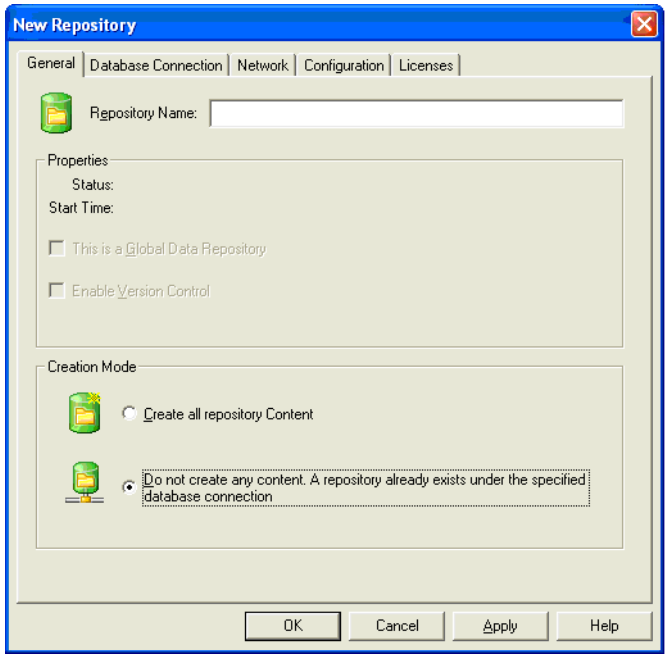
This task creates a repository containing configuration folders for the different source systems that Siebel Analytics Enterprise Applications supports. The Siebel Analytics Enterprise Applications has source-independent and application-independent folders, as well as the Post-Load Process folder. After you create the repository, you may want to delete the folders and the connections that are not applicable to you.

You can also configure your server information in the following procedure.

To create and configure an empty Siebel Analytics Enterprise repository

1. Launch the Repository Server Administration Console.
2. Right-click on Repositories, and select Add a Repository.
3. In the General tab, enter a name for your new repository, and click on the radio button, Do not create any content.

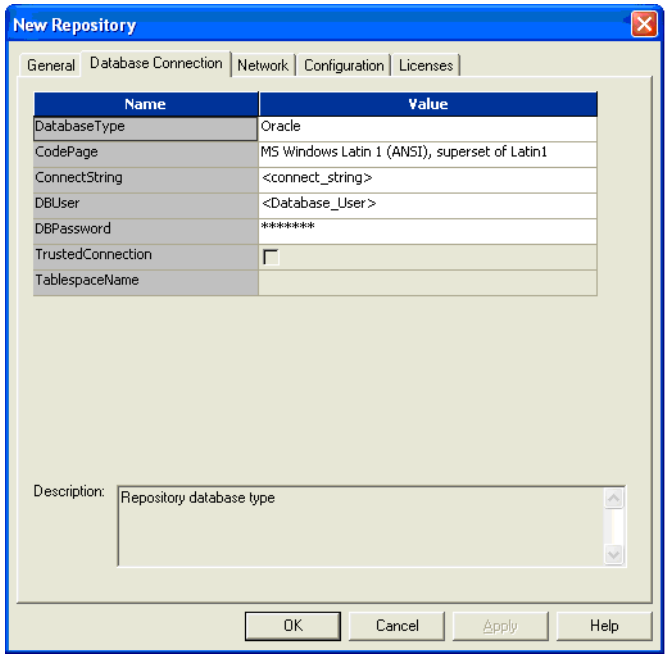
   An example is shown in the following figure.



4. Click the Database Connection tab and enter the following:
   - Database type
   - Code page
   - Connect string

- Database user
- Database password

An example is shown in the following figure.



5 Click the Licenses tab, and add your license key or keys.

6 Click Update and click OK.

7 Right-click on Repositories, and select All Tasks and Restore.

8 Browse to the Shell.rep file in the backup directory in the PowerCenter Repository Server, and click OK.

Your repository is created.

Registering Packages for the SAP R/3 Repository

This task is a step in the Process of Creating the Development Repository on page 28.

If you are sourcing from SAP R/3, then you must register the pmsapplg.xml plug-in on the repository.

To register a package for the SAP R/3 repository

1 Launch the Repository Server Administration Console.

2 Click on Available Packages.

3 Right-click on pmsapplg.xml plug-in and select Register.

4 In the Register Plugin dialog box, select your SAP R/3 Repository from the Repository drop-down list.

5 Select Update existing plugin registration check box.

6 Enter your user name and password.

7 Click OK.

Modifying the Repository and Repository Server

This task is a step in the Process of Creating the Development Repository on page 28.

To modify your server information log on to the Workflow manager.

To modify the repository and repository server

1 In the Workflow Manager client, right-click on Server, and then click Edit.

2 In the Server Editor, edit the Host Name or IP Address.

3 Click Resolve Server.

4 Verify that the Resolved IP Address box is unavailable, and displays the correct IP address of the server machine.

5 Click Advanced.

6 Enter the directory path to $PMRootDir (your PowerCenter root directory), and click OK.

Importing Module Workflows into the Repository

This task is a step in the Process of Creating the Development Repository on page 28.

You must import individual workflows for every module purchased within the Siebel Analytics Enterprise. Each module involves importing the module specific XML file using the PowerCenter Repository Manager.

To import the object metadata into the repository

1 In PowerCenter Repository Manager, open the Configuration folder for your source.

2 Connect to the repository.

3 Navigate to Repository File > Import Objects.

The Import Wizard is launched.

4 Click Browse, navigate to the XML file that contains the objects you want to import, and then click Next.

5 Click Add All, and then click Next.

NOTE: Click the Add All and not the Add option.

6 In the Match Folders screen, match the folders in your XML file to the folders in your destination repository, and click Next.

7 Specify the rules to resolve object conflicts, click Next, and then click OK.

8 Repeat the procedure for each module and XML file.

Backing Up the Development Repository

This task is a step in the Process of Creating the Development Repository on page 28.

The final step in creating your development repository is to back up the repository.

To back up the development repository

1 In the Administration Console, expand the Repositories node.

2 Select your repository.

3 Navigate to Action > All Tasks > Backup.

The Backup Repository dialog box appears.

4 Enter the repository user name, password, and file name for the repository backup file.

5 (Optional) If you want to overwrite an existing repository backup file, choose to replace the existing file.

6 Click OK after completing all changes.

Copying the Required Data Files

This task is a step in the Process of Creating the Development Repository on page 28.

The data files you must copy are currently located on the machine where you installed Siebel Analytics Enterprise Applications.

For Windows systems, copy the folders as described in the following procedure. For UNIX systems, use FTP to access these files, and be sure you use binary mode.

To copy the required data files to the PowerCenter Server

1 On the server where you installed Siebel Analytics Enterprise Applications, navigate to

`$Siebel\ETL\Siebel Warehouse Support Files\SrcFiles`

where

`$Siebel` is the path of the Siebel Analytics Enterprise Applications installation directory.

2 Copy all the files to $pmserver\SrcFiles

where

$pmserver is the path of the PowerCenter Server installation directory.

3 Return to the $Siebel\ETL\Siebel Warehouse Support Files\SrcFiles directory on your Siebel Analytics Enterprise Applications machine.

4 Repeat all the steps in this procedure for the Post_Load_Processing folder.

Creating the Data Warehouse Schema Control Tables

This task is a step in the Process of Creating the Development Repository on page 28.

This section describes how to create schema control tables so that the Siebel Analytics Enterprise Data Warehouse can operate successfully. If you already have a Siebel Relationship Management Warehouse, the following procedure modifies the existing date tables (W_DAY_D) and adds columns required for the Siebel Analytics Enterprise Data Warehouse. If you do not have a Siebel Relationship Management Warehouse, this step creates the necessary date tables.

To create the data warehouse schema control tables

1 Copy the ddlimp utility from the $Siebel\ETL folder.

2 Copy the schema control files from the $Siebel\ETL folder:

- ddlsme_warehouse.ctl
- ddlsme_staging.ctl
- ddlsme_control.ctl
- ddlsme_temp.ctl

3 Create a role in the database called SSE_ROLE with privileges to create objects.

NOTE: In Oracle this equates to CONNECT and RESOURCE privileges.

4 Create an ODBC DSN entry in the database.

5 Open the command prompt window, and change the directory to the folder where the preceding files are copied.

6 Create the data model, including all the required tables for the Siebel Analytics Enterprise Data Warehouse, by using the following command:

```
ddlimp /s N /u <schema_user>  /p <schema_password> /c <ODBC connection string> /G
"SSE_ROLE" /f ddlsme_warehouse.ctl /b "" /K "" /X "" /W N
```

NOTE: For the Oracle ODBC connection using the Siebel Merant ODBC driver is the preferred option.

7 Repeat Step 6 for ddlsme_staging.ctl, ddlsme_control.ctl, and ddlsme_temp.ctl.

To populate the fields, you must run the Common Initialization Workflow. For more information on common initialization workflows, see Running the Common Initialization Workflow on page 42.

Deploying Stored Procedures

This task is a step in the Process of Creating the Development Repository on page 28.

This section applies only to the Inventory and Sourcing modules for Oracle 11i, and the Compensation and Workforce Management Operations and Workforce Management Retention modules for PeopleSoft.

Stored procedures are a group of SQL statements that perform particular tasks on the database. For example, stored procedures can help to improve the performance of the database.

You can deploy stored procedures by copying the stored procedure files from your Siebel Analytics Enterprise installation and deploying them to the target data warehouse.

NOTE: Some sessions may fail if these procedures are not compiled in the database before running the workflows.

To deploy stored procedures

1. On the server where you installed the Siebel Analytics Enterprise Applications, navigate to the appropriate files, containing the stored procedures:
   - For Oracle 11i, navigate to `$siebel\ETL\Siebel Warehouse Support Files\Stored_Procedure_Scripts\Oracle_Apps_v11i`.
   - For PeopleSoft, navigate to `$siebel\ETL\Siebel Warehouse Support Files\Stored_Procedure_Scripts\Psft_75`.
2. Copy the stored procedures, and move the source codes into the data warehouse schema.
3. Compile the stored procedures in the target data warehouse database.

NOTE: If you have problems deploying the stored procedures, see your database reference guide, or contact your database administrator.

Creating the Seed Data

This task is a step in the Process of Creating the Development Repository on page 28.

To create the seed data you must run some SQL statements on your data warehouse. These SQL statements insert a row with zero (0) as the primary key in the Dimension and Fact tables.

To create the seed data

1. On the server where you installed the Siebel Analytics Enterprise Applications, navigate to insertSQL.txt.
2. Run the SQL statements found in insertSQL.txt on your data warehouse.
3. Commit the changes to your data warehouse.

Starting the PowerCenter Server

For Windows, start the service. For UNIX, carry out the following procedure.

To start the UNIX PowerCenter Server

1. Make two copies of the pmserver.cfg file and name them as follows:
   - `$pmserver_temp.cfg`
   - `$pmserver_dev.cfg`
2. Type pmconfig `pmserver_temp.cfg` in the UNIX shell.

   The configuration file appears. Some entries require their default values. You must configure some other entries to your environment.
3. Enter the required parameters for your PowerCenter repository.

Parameter Name	Parameter Value
ServerName	$PC_PROD_PMSERVER
RepositoryName	$PC_PROD_REPO_NAME
PMUser	Administrator
PMPassword	xxx
RepServerHostName	$REPO_SERVER_HOST_NAME
RepServerPortNumber	XXX
LogFileName	pmserver.log
DataMovementMode	ASCII
MaxSessions	10
LMSharedMem	2000000
PlatformKey	XXX
OracleKey	000

4. Repeat Step 2 and Step 3 to configure the $pmserver_dev.cfg file.
5. Verify that PMServer is running on its own port.
6. Start PMServer.

Setting Up the Siebel Analytics Enterprise Repository

Siebel Analytics Server Home directory is referred to as $SAHome.

To set up the Siebel Analytics Enterprise repository

1. Copy the Siebel Analytics Enterprise repository file (SiebelBusinessAnalytics.rpd) to your Siebel Analytics Server, <$SAHome\SiebelAnalytics\Repository\>.
2. Open the SiebelBusinessAnalytics.rpd with Admin Tool, and log in as the administrator, using the SADMIN ID and password, and then do the following:
   a. Edit the connection pool named Siebel Enterprise Connection Pool.
   b. Edit the DSN name to point to the Data Warehouse database.
   c. Set the User Id and Password and the database alias to connect to the database.
3. Repeat Step a through Step c for the connection pool named, Siebel Enterprise DBAuth Connection Pool.
4. Navigate to Manage > Variables > Static, and edit the OLAP_DSN, OLAP_USER and OLAPTBO variables. Set them to the OLAP DSN, User Id, and table owner, respectively.
5. Navigate to Manage > Security > Users, and then:
   a. Double-click on Administrator user, and enter a new password.
   b. Double-click on SADMIN, and enter a new password.
   c. Save and exit.
6. Save the repository, and, when prompted, click Yes to Check Global Consistency.
7. Click OK when WARNINGS are displayed, and exit from the application.

For more information about setting up the Siebel Analytics repository, see the *Siebel Analytics Installation and Configuration Guide*.

Setting Up the Siebel Analytics Web

To set up the Siebel Analytics Web you must modify the SiebelBusinessAnalytics.webcat file.

To set up the Siebel Analytics Web

1. Copy the following file:

   $Siebel\Web Catalog\SiebelBusinessAnalytics.webcat

   to:

   <$SAHome\SiebelAnalyticsData\Web\Catalog\>
2. Modify the registry to point to SiebelBusinessAnalytics.webcat.
3. Restart the Siebel Analytics Web service.

For more information about setting up the Siebel Analytics Web, see the *Siebel Analytics Installation and Configuration Guide*.

4 Initializing and Populating the Siebel Analytics Enterprise Data Warehouse

This chapter outlines the tasks you must do to prepare your data warehouse before performing your initial data capture.

This chapter contains the following topics:

Overview of Initializing and Populating the Siebel Analytics Enterprise Data Warehouse

To initialize and populate the Siebel Analytics Enterprise Applications, you must set up database connections, load your data warehouse with certain required default information (such as copyrights and dates), and configure data captures from your source. Finally, you must run your first data capture to populate your data warehouse. These tasks are required so that you may begin to analyze how prepackaged Siebel Applications components perform in comparison to your organization's business needs.

Process of Setting Up Database Access

This section contains information about database objects and access in relation to your system and the Siebel Applications installation.

To set up database access, you must perform the following tasks:

Related Topic

Database Types for Data Warehouse Objects

PowerCenter supports database independence. It is the database type that you specify in the Workflow Manager connection that determines the database type that PowerCenter Server uses. A target table that is displayed as the default, DB2, can actually be in SQL Server, DB2, or Oracle, depending on the Workflow Manager connection information.

Sources and targets found in the Siebel Analytics Enterprise Applications folder are usually defined with the database type DB2. The source-specific folders in the Designer navigator window also contain sources and targets defined for databases other than what you actually run. However, you must specify your database type in Workflow Manager. The only limitation is that a single Source Qualifier needs to read from all the tables (with the same database type) that are defined in a mapping.

Modifying the Repository Connections

This task is a step in the Process of Setting Up Database Access on page 37.

Your server configuration requires that the following database connections be configured to reflect the actual database type that your system is running:

- **Source.** `IA_[SOURCE ABBREVIATION]_SOURCE`
- **Staging Area.** `IA_[SOURCE ABBREVIATION]_STAGE`
- **Data Warehouse.** `IA_[SOURCE ABBREVIATION]_WAREHOUSE`

  NOTE: If you have more than one source, you must modify the connection for each flat file or database.

Configuring these database connections also helps to make sure that Siebel Analytics Enterprise Applications are installed correctly. Configure all three repository connections listed in the preceding paragraph for each of your source types.

Each source type requires a relational database connection, or an application database connection, or both. Relational and application connections have been defined for the following:

- Oracle applications with an Oracle database connection
- PeopleSoft applications with an Oracle database connection
- SAP R/3 with a source connection
- Warehouse with an Oracle database connection
- Staging with an Oracle database connection

If you want to use these connections, you can edit their properties. Otherwise, you must create a new connection. For a list of source types and their connection configuration, see Table 6.

Table 6. Source Types and the Connection Configuration

Source Type	Database Connection	Connection Type
Common	IA_WAREHOUSE	Relational
PeopleSoft	IA_PSFT8_SOURCE IA_PSFT8_STAGE IA_PSFT8_WAREHOUSE	Relational and application
Oracle 11i	IA_ORA11i_SOURCE IA_ORA11i_STAGE IA_ORA11i_WAREHOUSE	Relational
SAP R/3	IA_SAP_SOURCE IA_SAP_STAGE IA_SAP_WAREHOUSE	Application

To configure the relational database connections

1. In Workflow Manager, select Connections > Relational....
2. Select your target database in the Relational Connection Browser.

   NOTE: If you want to change the database type, you must:

   a. Select Edit....

   b. Select your database type, and click OK.
3. Click New....
4. In the Connection Object Definition box, type the appropriate Database Name, User Name, Password, Connect String, and Code Page, and click OK.
5. Repeat this procedure for each connection you want to configure:
   - Source
   - Staging area
   - Data warehouse
6. Click Close.

To configure the application database connections

1. In Workflow Manager, select Connections > Application....
2. Select your target database in the Application Connection Browser.
3. Click New....

4 In the Connection Object Definition box, type the appropriate Database Name, User Name, Password, Connect String, and Code Page for your source connection.

5 Click OK, and then click Close.

To replace relational database connections

1 Connect to the repository.

2 In Workflow Manager, select Connections > Replace Connections.

3 Click Add a New Connection Pair.

4 Enter your source database connection in the From field, and your new source database connection in the To field.

5 Click Replace.

NOTE: You cannot replace or copy an application database connection. You have to delete the original application connection, and create a new connection with the same name. For example, if you want to extract from a PeopleSoft DB2 database, you have to delete the existing PeopleSoft Oracle application connection called, IA_PSFT8_SOURCE. Create a new application connection using PeopleSoft DB2, with the same name, IA_PST8_SOURCE.

Optimizing Database Performance

This task is a step in the Process of Setting Up Database Access on page 37.

After setting up your database connections, but before loading any data, you may want to optimize your repository's database performance. To perform optimization, request your Database Administrator to run the appropriate, update-statistics operation for your RDBMS on all tables and indexes.

The optimization procedure updates information about the distribution of key values, and improves the performance of operations (for example, starting the workflow or opening mappings). It is also recommended that you run an update, statistics operation on both your source and target databases.

When the repository has been restored, and you are satisfied with your setup, you are then ready to run the Siebel Applications initialization workflows. For more information about initializing workflows, see About Working with Initialization Workflows on page 41.

About Working with Initialization Workflows

Not all the data required by the Siebel Analytics Enterprise Applications can be extracted from your source. To set up the Siebel Analytics Enterprise Data Warehouse correctly, you may want to supplement your source data with other information. For such additional data, the Siebel Analytics Enterprise Applications provides `*.csv` and `*.txt` files that are used to populate dimension tables. These files are placed in your `$Siebel\ETL\Siebel Warehouse Support File\SrcFiles` directory as an automatic part of the installation. The load workflows that you see listed in the Workflow Manager directory use these prepackaged files.

Some of these prepackaged files require no interaction from you. They are already packaged with the data that has to be loaded. Several prepackaged files are required by the Siebel Analytics Enterprise Applications to properly manage your data, and are initialized when you run the common initialization workflows. However, some of the prepackaged files do require further information that is specific to your data warehouse, so you can edit these files as described in this chapter.

Table 7 lists the Siebel Analytics Enterprise Data Warehouse Workflow Names.

Table 7. Siebel Analytics Enterprise Data Warehouse Workflow Names

Functional Workflow Name	Logical Workflow Name	Folder Name
Initialization Workflow	INITIALIZE	Siebel
Source-Specific Initialize Workflow	[SOURCE]_[SUBJECT]_Application_Initialize	Source Application
Source-Specific Main Workflow	[SOURCE]_[SUBJECT]_Application	Source Application
Post-Load Processing Workflow	[PLP]_[SUBJECT]_Application	Post Load Processing
Post-Load Processing Initial Workflow	[PLP]_[SUBJECT]_Application_INIT	Post Load Processing
Post-Load Processing Incremental Workflow	[PLP]_[SUBJECT]_Application_INCR	Post Load Processing

When you load the data warehouse for the first time run the following:

1. The initialization workflow
2. The source-specific initialize workflow, if applicable to your module
3. The source-specific main workflow
4. The post-load processing initial workflow or the post-load processing workflow, whichever is in your configuration folder.

When refreshing the data warehouse, you must run the following:

1. The source-specific main workflow
2. The post-load processing incremental workflow or the post-load processing workflow, whichever is in your configuration folder.

Process of Working with Initialization Workflows

To work with initialization workflows, perform the following tasks:

- Running the Common Initialization Workflow on page 42
- Loading Your Fiscal Calendar on page 45

Related Topics

- About Working with Initialization Workflows on page 41
- Common Initialization Workflow Files on page 44
- Working with Source-Specific Main Workflows on page 47

Running the Common Initialization Workflow

This task is a step in the Process of Working with Initialization Workflows on page 42.

Running the common initialization workflow, INITIALIZE, verifies that both the PowerCenter and the Siebel Analytics Enterprise Data Warehouse are installed successfully. Running the INITIALIZE workflow also initializes the dimension tables, and verifies that your date tables are configured correctly.

The common initialization workflow must be run only once, but it is run before any of the other workflows. The INITIALIZE workflow contains prepackaged files that are required by your system, regardless of your source type, to use the Siebel Analytics Enterprise Applications installation.

The common initialization workflow contains four worklets, Z_INITIAL, Z_LOAD_DATES_TIME, Z_LOAD_DATES_AGGR, and Z_LOAD_DATES INIT, in addition to one stand-alone workflow, S_M_Z_PARAMETERS_FILE_LOAD. These worklets, nested within the initialization workflows, write results to the target tables specified in Table 8. A *worklet* is an object that represents a set of tasks. It allows you to reuse a set of workflow instructions in several workflows. Table 8 lists the common initialization worklets and the target tables.

Table 8. Common Initialization Worklets and Target Tables

Workflow or Worklet Name	Task Name	Target Table Name
Z_INITIAL	S_M_Z_INITIAL_LOAD_1	IA_DATES

Table 8. Common Initialization Worklets and Target Tables

Workflow or Worklet Name	Task Name	Target Table Name
		IA_CAL_MONTHS
		IA_CAL_WEEKS
		IA_CAL_QTRS
		IA_CAL_YEARS
		IA_FSC_MONTHS
		IA_FSC_WEEKS
		IA_FSC_QTRS
		IA_FSC_YEARS
Z_LOAD_DATES_TIME	S_M_Z_TIME_OF_DAY_LOAD	IA_TIME_OF_DAY
	S_M_Z_TIME_OF_DAY_AGGR	IA_HOUR_OF_DAY
	S_M_Z_DATES_LOAD	IA_DATES
Z_LOAD_DATES_AGGR	S_M_Z_DATES_AGGR	IA_CAL_MONTHS
		IA_CAL_WEEKS
		IA_CAL_QTRS
		IA_CAL_YEARS
	S_M_Z_DATES_FSC_AGGR	IA_FSC_MONTHS
		IA_FSC_WEEKS
		IA_FSC_QTRS
		IA_FSC_YEARS
	S_M_Z_DATES_FSC_AGGR	IA_DATES
	S_M_Z_SYSDATE_CREATION	TZ_DATES_GENERIC
Z_LOAD_DATES	SIL_DayDimension	W_DAY_D

Table 8. Common Initialization Worklets and Target Tables

Workflow or Worklet Name	Task Name	Target Table Name
	SIL_DayDimension_GenerateRow1	W_DUAL_G
	SIL_DayDimension_GenerateRow2	W_DUAL_G
	SIL_DayDimension_GenerateRow3	W_DUAL_G
	SIL_DayDimension_GenerateRow4	W_DUAL_G
	SIL_DayDimension_GenerateRow5	W_DUAL_G
	SIL_DayDimension_GenerateRow6	W_DUAL_G
	SIL_DayDimension_GenerateRow7	W_DUAL_G
	SIL_DayDimension_GenerateSeed	W_DUAL_G
	SIL_DayDimension_Unspecified	W_DAY_D
	SIL_FiscalMonthDimension	W_FSC_MONTH_D
	SIL_FiscalWeekDimension	W_FSC_WEEK_D
	SIL_MonthDimension	W_MONTH_D
	SIL_QuarterDimension	W_QTR_D
	SIL_WeekDimension	W_WEEK_D

Common Initialization Workflow Files

The following files are found in the common initialization workflow, INITIALIZE:

- timespan.txt
- dates.txt

You modify the timespan.txt and the dates.txt files. These files reflect your data warehouse time span. Do not modify the other files in the INITIALIZE workflow, because the system requires those default values during installation.

Timespan.txt File

Required by the INITIALIZE workflow. Open the timespan.txt, edit the $$START_DATE, and the $$END_DATE.

Time.txt File

Required by the Z_LOAD_DATES_TIME worklet. The time.txt file contains references to the time_am.csv, and time_pm.csv files. Do not modify the file.

Time_am and Time_pm.csv Files

Required by the Z_LOAD_DATES_TIME worklet. The time.csv files contain either the A.M. or the P.M. hours of the 24-hour day. Do not modify these files. Each of the two comma-separated files contains a 12-hour segment of the day, broken down by minute and second, and mapped to a description such as night, dawn, and morning for the A.M. hours, or afternoon, evening, and night for the P.M. hours.

Dates.txt File

Required by the INITIALIZE workflow, the dates.txt file contains pointers to each of the Dates_xxxx_xxxx.csv files. You must configure this file to match your data warehouse requirement for a time horizon.

Example of Dates.txt File Use

For example, imagine that today is March 3, 2004 and you want to create a data mart that contains your data from January 2004 through December 2012. In your dates.txt file, you reference the dates_xxxx_xxxx.csv files (see the next paragraph) that define your desired time horizon by removing the entries for the files that are not needed. In this example, you must reference two files (dates_2000_2009.csv and dates_2010_2019.csv), so you remove the entries for the other Dates_xxxx_xxxx.csv files that do not match your time horizon.

Dates_XXXX_XXXX.csv Files

Required by the Z_LOAD_DATES_TIME worklet. The Dates.csv files are used as described under the heading, Dates.txt. Do not modify the Dates_xxxx_xxxx.csv files.

Example of Dates_XXXX_XXXX.csv Files Use

In this example, the Xs are a placeholder for the decade start and end dates (such as 1970–1979). These files are divided by decades. For example, the 1970s decade is 1970-1979. The files are optional, because you choose the appropriate decade files, depending on the time horizon you want to represent in your data warehouse. (For example, if you want to represent 1990 through 2009, you keep the Dates_1990_1999.csv and the Dates_2000_2009.csv files in your Dates.txt file, thus defining your time horizon.) Each comma-separated file contains every day of the decade, mapping it to weeks, months, quarters, and years. Each file also contains other descriptive information for each year, such as day of week, end or beginning of month flags, and so on.

Loading Your Fiscal Calendar

This task is a step in the Process of Working with Initialization Workflows on page 42.

The INITIALIZE Workflow contains worklets to load information about your company's fiscal calendar, holidays, and other date-related concepts. If you want to load your Fiscal Calendar, you must configure the INIALIZE workflow before you can run the Fiscal Calendar.

The dates dimension (IA_DATES) contains fields that you can configure to store fiscal calendar information. By default, the fields that contain the fiscal information are populated with calendar information. Therefore, if your fiscal calendar differs from the standard calendar, you must configure one of two prepackaged .csv files (fiscal_months.csv, and fiscal_week.csv). Your fiscal calendar must be set up such that fiscal weeks roll up to fiscal months, which roll up to fiscal quarters, which roll up to fiscal dates. By default, the Z_UPDATE_FISCAL_DATES_BY_WEEKS workflow is disabled, and you load the fiscal calendar by running the Z_UPDATE_FISCAL_DATES_BY_MONTHS.

The explanations that follow help you to determine your system needs, and to choose an appropriate option, based on how much information you want to specify.

Configuring Your Fiscal Month

If you want to specify the beginning date of each fiscal month within a fiscal year, you may do so in the fiscal_months.csv file. The Z_UPDATE_FISCAL_DATES_BY_MONTHS worklet uses the fiscal_months.csv file as source information to calculate the appropriate fiscal week information, within each month that is specified. Run the worklet to update each record in the IA_DATES table for the period that you specify.

Configuring Your Fiscal Week

If your fiscal week information cannot be described using the methods in the preceding sections, you may use the fiscal_week.csv file. You can specify the start date for each week, and indicate which fiscal week, month and year each week is in, with the fiscal_week.csv file. After the file has been modified, enable the Z_UPDATE_FISCAL_DATES_BY_WEEKS worklet, and disable the Z_UPDATE_FISCAL_DATES_BY_MONTHS worklet in the INITIALE workflow. The Z_UPDATE_FISCAL_DATES_BY_WEEKS worklet calculates only information within the parameters that you specified, such as the day in the fiscal week.

Using the Fiscal_months.csv and the Fiscal_weeks.csv Files

There are two fiscal calendar files, one of which you have to modify. The two fiscal calendar files are used by the Z_UPDATE_FISCAL_DATES_BY_MONTHS or Z_UPDATE_FISCAL_DATES_BY_WEEKS worklets. The contents of these files must represent your company fiscal month or week. You can edit the files by entering the correct start and end dates for your company's fiscal month, week, or year.

To modify the Fiscal_weeks input file

1. Open the fiscal_weeks.csv file in the $Siebel\ETL\Siebel Warehouse Support Files\Sample_Data\Universal_Source\Common folder.
2. Following the format specified in the spreadsheet, start with row six and enter the number of your fiscal year in column A, month in column B, and week in column C.

   Enter the start date of the fiscal week for a given year in column D.
3. Repeat the numbering process for each week in the fiscal year.

   Be sure to identify the correct month for each week.
4. Select File > Save, and close the file.

Follow the preceding procedure to edit the fiscal_months.csv file used by the Z_UPDATE_FISCAL_DATES_BY_MONTHS worklet to configure your fiscal calendar.

Working with Source-Specific Main Workflows

This section allows you to prepare your source-specific main workflow so that it meets your implementation requirements. In this source-specific main workflow, the Siebel Analytics Enterprise Applications provides you with the logic to extract all the data from a particular source system. Depending on your source system, the workflow contains worklets for each Siebel Analytics Enterprise Application product.

You may want to remove some worklets that are not part of your implementation, depending on the module or product family you want to use. After configuring your main workflow, run it to capture your first, complete extraction of all the rows in your source database. The result provides you with a populated target schema that you can use for further configuration.

Most of the source-specific main workflows that you see in the Navigation window of the Workflow Manager contain nested worklets. These usually include the prepare worklet, fact extract worklet, dimension extract worklet, dimensions load worklet, facts load worklet, and the execution finished worklet.

The purpose of the primary extract sessions is to track any physical deletions that occur in the source system. Primary extract sessions do not perform a full data extract; they extract only the key_id and source_id for facts. The delete sessions delete the rows that are in the data warehouse.

If rows are physically removed from your source system, you must make a choice about retaining the rows in your data warehouse:

- If you want to retain the rows in the data warehouse even though the rows are removed from the source system, then keep the default primary extract workflows, and the corresponding delete mappings disabled.
- If you do not want to retain the rows in the data warehouse after they are removed from the source system, then enable the default primary extract workflows, and the corresponding delete mappings.
- If your source system archives rows, you may want to set a parameter to search for archive dates, and then execute delete mappings only on rows that have been archived and are no longer needed in the warehouse. See the discussion on deletion configuration for source-archived records, in Working with Primary Extract and Delete Mappings on page 234.

About Modifying Session Parameters for Initial and Incremental Loads

Parameters are designed to regulate the behavior of workflows and tasks, and are set using a source file. Siebel Analytics Enterprise supplies this file. However, you must modify the appropriate .csv parameters source file to set up your ETL process frequency. For a list of parameter files for every source application see Table 9.

Table 9. Source Folders and Parameter Files

Source Folder	Parameter File
Configuration for Oracle Applications v11i	file_parameters_ora11i.csv
Configuration for PeopleSoft 7.5	file_parameters_psft75.csv
Configuration for PeopleSoft 8	file_parameters_psft8.csv
Configuration for SAP R/3	file_parameters_sapr3.csv

Table 10 is an example of the file_parameters_psft8.csv parameter file. It shows a number of important columns and example sessions for the file_parameters_psft8.csv parameter file.

Table 10. Example of the file_parameters_psft8.csv Parameter File

PARM_TYPE	PARM_CODE	PARM_NVALUE_1	PARM_SVALUE_1	PARM_SVALUE_3	PARM_DVALUE_1	Source_ID
PSFT80	S_M_P_AP_XACTS_VCHR_ACCTG_LINE_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	0	D	MPLT_BCP_AP_XACTS_VCHR_ACCTG_LINE_GL_REFERENCE_EXTRACT	20040010100000	PSFT80
PSFT80	S_M_P_AR_XACTS_BI_ACCT_ENTRY_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	0	D	MPLT_BCP_AR_XACTS_BI_ACCT_ENTRY_GL_REFERENCE_EXTRACT	20040101000000	PSFT80
PSFT80	S_M_P_AR_XACTS_ITEM_DST_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	0	D	MPLT_BCP_AR_XACTS_ITEM_DST_GL_REFERENCE_EXTRACT	20040101000000	PSFT80
PSFT80	S_M_P_GL_COGS_CM_ACCTG_LINE_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	0	D	MPLT_BCP_GL_COGS_CM_ACCTG_LINE_GL_REFERENCE_EXTRACT	20040101000000	PSFT80

The following are the important columns in the parameter file:

- **PARM_TYPE.** The PARM_TYPE column represents the source system you are working on. For example, PeopleSoft 8 has a value of PSFT80.
- **PARM_CODE.** The PARM_CODE column contains the name of the session concatenated with the session parameter name. You have to configure this value. For example, in both the initial and incremental extracts mappings in Table 10, you use the LAST_EXTRACT_DATE parameter.
- **PARM_NVALUE_1.** The PARM_NVALUE_1 column represents the number of days you want to use as your extraction window. For your initial loads, the value is 0. For incremental loads with a window of 4 days from the current system date, the value is 4. For more information on configuring these dates, see Configuring the Date Parameters for Parameter Files on page 49.
- **PARM_SVALUE_1.** The PARM_SVALUE_1 column represents the type of parameterization being done. For example, when configuring dates the value is D.
- **PARM_SVALUE_3.** The PARM_SVALUE_3 column represents the mapplet name. For Siebel Analytics Enterprise Applications, most of the extraction logic is hidden in the business component mapplet. The mapplet name corresponds to the PARM_CODE column.
- **PARM_DVALUE_1.** The PARM_DVALUE_1 column represents the date that is used for initial runs. The value from this column is used only when PARM_NVALUE_1 has a non-zero value.
- **SOURCE_ID.** The SOURCE_ID always represents the source system you are working on. For example, PeopleSoft 8 has a value of PSFT80.

Process of Modifying Session Parameters for Initial and Incremental Loads

To modify the session parameters for the initial and incremental loads, you must perform the following tasks:

- Configuring the Date Parameters for Parameter Files on page 49
- Configuring the Database Parameter for the Source System on page 51

Related Topic

- About Modifying Session Parameters for Initial and Incremental Loads on page 48

Configuring the Date Parameters for Parameter Files

This task is a step in the Process of Modifying Session Parameters for Initial and Incremental Loads on page 49.

If you use Table 10 for the initial run, the PARM_DVALUE_1 parameter has a value of January 01, 2004. If you extract changed data from 2004, then you are extracting the entire data set. If you have older data, change the value of PARM_DVALUE_1 in the format yyyymmddhh24miss.

If you want incremental runs with a two-day window for the first two sessions, you change only the PARM_NVALUE_1 column for the corresponding sessions. If you want incremental runs with a three-day and four-day window for the third and the fourth sessions, you also change the PARM_NVALUE_1 column for the corresponding sessions. When PARM_NVALUE_1 has a nonzero value, PARM_DVALUE_1 is not used. For an example of how your parameter file (file_parameters_psft8.csv) would appear for incremental loads, see Table 11.

Table 11. Example of the file_parameters_psft8.csv Parameter File for an Incremental load

PARAM_TYPE	PARAM_CODE	PARAM_NVALUE_1	PARAM_SVALUE_1	PARAM_SVALUE_3	PARAM_DVALUE_1	SOURCE_ID
PSFT80	S_M_P_AP_XACTS_VCHR_ACCTG_LINE_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	2	D	MPLT_BCP_AP_XACTS_VCHR_ACCTG_LINE_GL_REFERENCE_EXTRACT	20040101000000	PSFT80
PSFT80	S_M_P_AR_XACTS_BI_ACCT_ENTRY_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	2	D	MPLT_BCP_AR_XACTS_BI_ACCT_ENTRY_GL_REFERENCE_EXTRACT	20040101000000	PSFT80
PSFT80	S_M_P_AR_XACTS_ITEM_DST_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	3	D	MPLT_BCP_AR_XACTS_ITEM_DST_GL_REFERENCE_EXTRACT	20040101000000	PSFT80
PSFT80	S_M_P_GL_COGS_CM_ACCTG_LINE_GL_REFERENCE_EXTRACT:LAST_EXTRACT_DATE	4	D	MPLT_BCP_GL_COGS_CM_ACCTG_LINE_GL_REFERENCE_EXTRACT	20040101000000	PSFT80

To modify the .csv parameter file

1. Navigate to the appropriate .csv parameter file.

   For example, for Oracle 11i navigate to `$pmserver\SrcFiles\file_parameters_ora11i.csv`. For a list of parameter files, see Table 9.

2. Copy your .csv file to the SRCFiles folder under the PowerCenter Server folder.
3. Open the file using Microsoft WordPad or Notepad.
4. Search for the column labeled PARM_DVALUE_1.
5. Change the default date (197001010000) to the date you require.
6. Change your new date in every session you want to run incrementally.
7. Save your file.

Configuring the Database Parameter for the Source System

This task is a step in the Process of Modifying Session Parameters for Initial and Incremental Loads on page 49.

You must configure your database parameter for your source system in the M_Z_SESS_PAR_FILE_GENERATE mapplet in the Siebel Applications folder.

NOTE: If you are using the Configuration for Oracle 11i and the Configuration for SAP R/3 folders, then you do not have to configure the database parameter. For SAP, the extracts are done using ABAP codes, and the initial and incremental filters do not depend on any specific back-end database. For both versions of Oracle Applications, you use the ORACLE RDBMS.

To configure the database parameter for the source system

1. In PowerCenter Designer, open the Siebel Applications folder.
2. Open the M_Z_SESS_PAR_FILE_GENERATE mapplet with Mapping Designer.
3. Double-click the EXP_PARAMETERS Expression transformation to open the Edit Transformation box.
4. Click the Ports tab.
5. Set the expression for the port DATABASE_NAME_VAR with the database value you are using for your source system.
   - If the source system database is Oracle 11i, set the value to ORACLE.
   - If DB2, set the value to DB2.
   - If MS SQL SERVER, set the value to MSSQL.
6. Validate the mapping and save the repository.

About Working with Post-Load Processing Workflows

Regardless of which workflow you use to execute your data captures, there is one feature, post-load processing, which requires execution for your initial data load to operate successfully. This section outlines steps for working with the post-load processing feature.

The post-load processing workflows search for an indicator file before they start running. This indicator file is created by the Execution Finished Worklet, and is called, file_plp_<suite name>.ind.

TIP: If you want the post-load processing workflow to start as soon as your fact loads complete, schedule these two workflows at the same time. The post-load processing workflow starts as soon as the indicator file appears in its directory.

NOTE: The General Ledger, Payables, Profitability, and the Receivables modules have a post-load processing initial workflow, and a post-load processing incremental workflow. Run the initialization workflow the first time you load your data warehouse, and the incremental workflow to refresh your data warehouse.

5 Planning Your Warehouse Configuration Project

This chapter provides guidelines on how to assess which objects need to be configured. After you complete your configuration project, do not forget that every time a business rule changes, or changes occur in your source system, you may need to configure this project again.

This chapter discusses important topics that are applicable to the planning of your configuration project. It contains the following topics:

Data Warehouse Configuration Stages

There are many stages where you can configure the handling of data to meet your needs. Configuration can happen in any of the following stages:

- **Extract Mapping.** For more information, see Configurable Objects in the Extract Mapping on page 54.
- **Load Mapping.** For more information, see Configurable Objects in the Load Mapping on page 55.
- **Post-Load Mapping.** For more information, see Configurable Objects in the Post-Load Mapping on page 55.
- **Reporting.** For more information, see Configurable Objects in the Reporting Area on page 56.

Where to configure an object depends on what you are trying to accomplish. Each stage performs different tasks when sourcing, transforming, loading, and reporting data. Generally, data goes through an extract mapping, load mapping, and front-end calculation process. Sometimes data also goes through the post-load stage, where the data is transformed before populating a key figure or aggregate table. Figure 4 illustrates these stages.

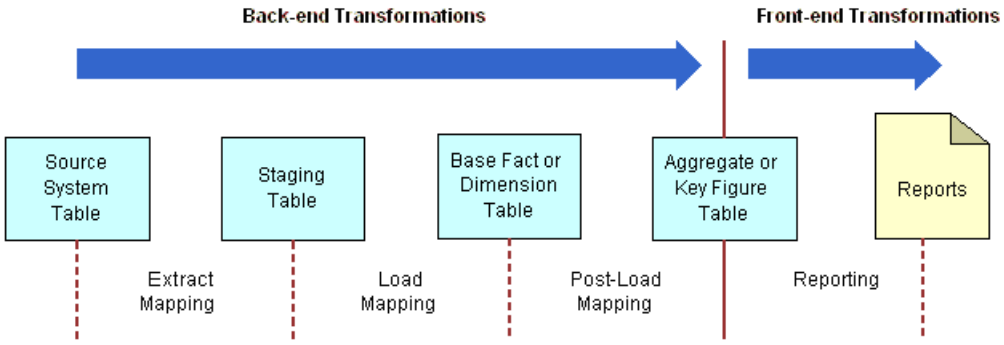


Figure 4. Stages for Configuration

Each of these stages contains various objects that are configurable. The sections that follow describe some of the common, configurable objects in each stage.

Configurable Objects in the Extract Mapping

Extract mappings are useful in extracting data from a source system and staging the data in a staging table. Within each extract mapping (with the exception of Universal Source extract mappings), you find the following objects—Business Component mapplet, Expression transformation, and staging table. A *mapplet* is a reusable object that contains a set of transformations, and it allows you to reuse the transformation instructions in multiple mappings.

The Business Component mapplet can be configured to do any of the following tasks:

- Extract additional data from new or existing source tables.
- Set up an incremental extract.
- Filter the type of records extracted.

NOTE: Universal Source extract mappings do not contain Business Components. These mappings have a flat file source which requires that most transformations be performed prior to the extract mapping.

The Expression transformation in the extract mapping can be configured to perform the following tasks:

- Set the Key ID for all records.
- Set the Source ID for all records.

The staging table is the target table for the extract mapping, and stores the extracted information.

Configurable Objects in the Load Mapping

Load mappings are useful in transforming and porting data from the staging table to a base fact or dimension table. Within each load mapping, you find the following objects—Source Qualifier transformation, Source Adapter mapplet, Analytic Data Interface (ADI) mapplet, and target table or tables.

The Source Qualifier transformation can be configured to filter records being loaded into the Siebel Analytics Enterprise Data Warehouse target table.

The Source Adapter mapplet in the load mapping is where you usually find all source-dependent transformations. This is the last, and preferred, stage at which you can perform source-dependent transformations. The Source Adapter mapplet can be configured to do any of the following tasks:

- Transform source data so that it becomes source-independent. For example, it maps some source values to the Siebel Applications domain values.
- Provide values for dimension class table types, such as COMPANY as the business location type for the MPLT_SAS_BUSN_LOCS_COMPANY data.
- Load exchange rates and currency codes, instead of performing lookups to retrieve them.
- Set the dimension IDs in fact loads so that it can resolve the dimension keys.
- Set the Type 2 flags.
- Convert positive values to negative values.

The ADI mapplet in the load mapping must not be configured at the mapping level. Instead, changes must be made at the session level, using an SQL override. The types of SQL overrides include—redirecting lookups to dimension, code, exchange rate, and custom-built dimension tables. The target tables of a load mapping can vary. If the table is a dimension load mapping, and Type 2 functionality has been enabled, you usually have two instances of the IA_* target table:

- One instance for inserting new records
- One instance for updating records

You also have an OD_* load control table. If the table is a fact load mapping, you usually have one IA_* target table and an OD_* load control table.

Configurable Objects in the Post-Load Mapping

Post-load mappings are useful in transforming data from a base fact table, and loading it into an aggregate or key figure table. For information about aggregate and key figure tables, see the Siebel Applications Overview. Because post-load mappings vary widely, the only common items that are configurable are:

- Time period for which data is stored or aggregated
- Calculations for metrics contained within the aggregate and key figure tables

Configurable Objects in the Reporting Area

If you have purchased Siebel Applications, which contains the Siebel Analytics Enterprise Data Warehouse and the Siebel Analytics Server schema and metadata, then you can configure metrics, attributes, reports, hierarchies, and schema definitions in the Siebel Analytics Server repository. This is the last stage at which you can perform configuration tasks.

About Scoping Your Configuration Project

After you have completed the initialization process as described in the installation chapters, the Siebel Analytics Enterprise Data Warehouse is ready to be populated using the default configuration. Although the default configuration produces data in the data warehouse, which can be used to populate your front-end analytic environment, it may not accurately reflect your business models. Your company may have unique business rules and requirements that depart from the Siebel Analytics Enterprise Data Warehouse's prepackaged assumptions.

For example, your definition of a sales order may not be the same as the definition in the Siebel Analytics Enterprise Data Warehouse. In addition to storing sales orders as defined by the Siebel Analytics Enterprise Data Warehouse, the Sales Order table in your source system may also store sales inquiries, sales estimates, or other customized data to suit your specific needs. If your definition of sales orders is either more general, or more specific than the Siebel Analytics Enterprise Data Warehouse definition, you must configure the Siebel Analytics Enterprise Data Warehouse to store the data you require. The research required to determine which areas you need to configure is called *gap analysis*. For more information, see Gap Analysis on page 56.

Gap Analysis

Gap analysis identifies the difference, or gaps, between what a product does by default and what your company needs the analytic solution to do. Gap analysis must be performed both at the time of installation, and as you prepare to do the configuration. This chapter focuses on gap analysis in terms of configuration.

TIP: It is recommended that you perform gap analysis, beginning with the front-end, and then work your way down to the Siebel Analytics Enterprise Data Warehouse. This approach saves you time and effort. This chapter provides gap analysis only for the Siebel Analytics Enterprise Data Warehouse.

Preparing for Gap Analysis

To best prepare for gap analysis, you must understand the following components of your analytic solution:

- Your source system. You can do this by understanding:
  - How you use the default features of your source system. Use your source system's standard documentation.

- How you have customized your source system. Use any documentation that describes modifications to the source system's default configuration. These documents may provide information on the effect that customizations to your source system have on how data is stored in the Siebel Analytics Enterprise Data Warehouse.

- The custom columns you need that fall outside of the Siebel Analytics Enterprise Data Warehouse data model.
- The Siebel Analytics Enterprise Data Warehouse. You can do this by understanding:
  - The architecture of the product
  - Each data warehouse table column's origin in your transaction source system
  - Each column's business meaning and expected values
- Your business rules, and how they differ from the ones assumed by the Siebel Analytics Enterprise Data Warehouse.

Guidelines for Populating the Siebel Analytics Enterprise Data Warehouse with Data

You are ready to embark on gap analysis for the back-end as soon as you have populated the Siebel Analytics Enterprise Data Warehouse with the data. The following list provides recommended steps for populating the Siebel Analytics Enterprise Data Warehouse:

1. Complete the installation and initialization processes described in the previous installation chapters of this guide. If you are upgrading, complete the upgrade process before conducting gap analysis on the new features.
2. Use the default configuration to complete one entire extract and load cycle to populate the Siebel Analytics Enterprise Data Warehouse with your data.

After the data warehouse is populated, you are ready to begin gap analysis.

Beginning Gap Analysis

The most efficient way to approach gap analysis is to start with the front-end reports. If you do not have easy access to the front-end reports, you can perform the gap analysis strictly from the back end. To understand where gap analysis ends in the front-end, and begins in the back-end, you must understand the entire gap-analysis process, as shown in Figure 5, and outlined as follows:

1. **Begin with your front-end reports.** Do they meet all your needs? If your requirements do not exceed what is available in the reports, you do not have any gaps to analyze. Otherwise, you need to identify information that is not available in the default reports. To do so, proceed to the next step.
2. **Continue with your front-end solution.** Is the information you are seeking already available in, or can it be derived from, the front-end metadata?

3 **Conclude with the data warehouse data model.** Is the information you need already being extracted from the source system, or do the extracts need configuration?

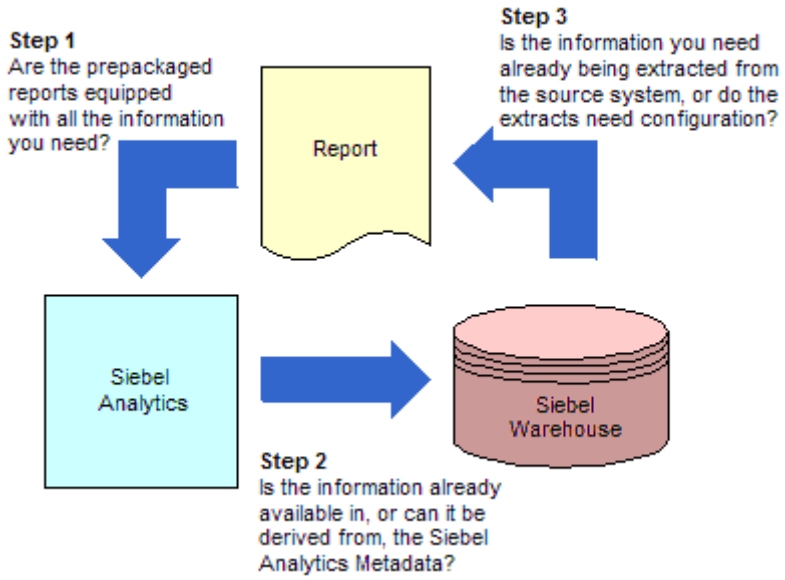


Figure 5. Simple Approach to Gap Analysis

Resolving Gaps in the Data Warehouse Data Model

You can redefine, add, or delete functionality in the data warehouse model to best meet your business requirements. The following sections provide a brief overview of general back-end configuration information that spans all modules. They also provide configuration information specific to each module.

There are two high-level possibilities for configuration at the back end:

- The information you need is already extracted, but is not transformed and loaded as you require. To reconfigure, modify the appropriate mappings to transform and load additional data into the extension columns provided in the target table.
- The information you need is not being extracted from your source system. There are many ways to address this issue—the method you choose depends on how much information is required. You can modify an existing extract mapping to extract the data, or create a new extract mapping.

Roles Involved in the Gap-Analysis Process

Gap Analysis requires the expertise of a variety of analysts, experts, and specialists. The following is a list of roles commonly involved in gap analysis, which may be filled by more than one individual, according to your company's needs:

- **Source specialist.** The source specialist is often a consultant who has helped you implement the source system or who works for the manufacturer. A source specialist knows your company's requirements of the source system, as well as customizations that must be included in the Siebel Analytics Enterprise Data Warehouse.
- **Business analyst.** The business analyst understands the intricacies of the business process and your reporting requirements. For example, the Customer Relationship Management business analyst understands how a sales order and an invoice are defined, and the information that is sourced from them to produce the necessary reports.
- **Front-end specialist.** The front-end specialist participates in designing reports and customizing the user interface. The business analysts use these reports created by the front-end specialist.
- **Professional Services Consultants (PSCs).** Siebel PSCs, or other third-party consultants, are responsible for making sure that your implementation of the Siebel Analytics Enterprise Data Warehouse runs smoothly. They provide expertise in the Siebel Analytics Enterprise Data Warehouse, data warehousing concepts, and various source systems.
- **Data warehouse expert.** The data warehouse expert is familiar with your storage requirements, performance levels, and database access.

Configuration Guidelines

Although the applications are configurable, it is not recommended that you modify the data model itself. Before performing any customization work, review the following best practices.

Choosing Where to Transform an Object

When you choose an object to edit, such as a mapping, it is recommended that you change the transformation as close to the source as possible. For example, if you are editing a delete mapping to handle archived records, it is better to edit the Source Qualifier's SQL rather than add a filter transformation after the Source Qualifier. Following this guideline may minimize efforts when upgrading, as well as the impact on prepackaged logic you want to retain.

Documenting All Changes

It is critical to document your work as you add and change objects in the data warehouse, so that the modifications are available when upgrading to new versions of the Siebel Analytics Enterprise Data Warehouse. Create a change log to record the object added or modified, the objects it impacts, and a brief description of your task. (See Table 12 for a sample change log.) For example, if you add source tables to a Business Component, record the additional tables in the log as well as their purpose and impact. When upgrading, you can make the decision to add the tables again, or you can identify if the upgrade itself has made the additional tables unnecessary.

Table 12. Sample Change Log

Entry Number	Date Entered	Entered By	Object Type	Description of Change	Status
1	01/03/03	Sara Chin	Session	Modifying the S_M_S_SALES_ORDLNS _LOAD session so that non-booked orders are loaded into the Sales Order Lines table (IA_SALES_ORDLNS). Do this by removing the booking flag filter condition in the following two filter transformations: FIL_SALES_BKGLNS and FIL_SALES_BKGLNS1.	In progress. Assigned to Oscar Tapa.
2	01/04/03	Mike Thomas	Expression transformation	Reset the AP Balance ID for the EXP_SAI_AP_XACTS Expression transformation from GL_ACCOUNT_ID ||'~'|| VENDOR_SITE_ID ||'~'|| ORGANIZATION_ID to GL_ACCOUNT_ID ||'~'|| VENDOR_SITE_ID.	Completed by Mike Thomas.

6 Configuring the General Ledger Module

After the General Ledger module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

Overview of the General Ledger Module

The Siebel General Ledger module provides information to support your enterprise's balance sheet and provides a clearer understanding of the chart of accounts.

For more information about the General Ledger module, see the *Siebel Analytics Enterprise Applications User Guide*.

The default configuration for the General Ledger module is based on what has been identified as the most-common level of detail or granularity. However, you can configure and modify the extracts to best meet your business requirements. The following subsection provides information about the extract, transform and load process for Oracle 11i that applies to all Financial Analytics-related modules (General Ledger, Receivables, Payables, and Profitability).

Fact Table Extract, Transform, and Load Process for Oracle 11i

The extract, transform and load (ETL) method used with Oracle applications assumes that posting from your journal to your general ledger is done at the summary level, and that references are maintained in General Ledger for Accounts Payable (AP) and Accounts Receivable (AR) subledgers. The following section provides the strategy for the ETL process for the General Ledger transaction journal.

The driving table for extracting transactions from Oracle Applications is the GL_JE_LINES table. The mapping M_I_GL_XACTS_JOURNALS_EXTRACT for Oracle 11i selects:

- Actual transactions from GL_JE_HEADERS
- Posted transactions from GL_JE_LINES_STATUS
- Periods from GL_PERIOD_STATUSES

The extract loads data into the staging tables TI_GL_JOURNALS and TI_STAGE_GLRF_DERV. The derv staging table is a configurable filter that you can redefine if you are posting data to the General Ledger at the detail level. For additional information about configuring the detail-level data, see Extracting Data Posted at the Detail-Level for Oracle 11i on page 62.

In the transformation stage, the M_I_STAGE_GL_XACTS_DERIVE mapping moves the information from TI_GL_JOURNALS and merges that with the definitions from IA_GL_ACCOUNTS. The information is then apportioned into a series of staging tables based on FIN_STMT_ITEM_CODE, which covers the basic accounting areas of AR, AP, Revenue, Tax, and Other.

Load mappings move information from staging tables into a fact table (IA), a load control table (OD), and an aggregate table (NU). The load control table mimics the structure of the IA table, and stores the data in the format in which it was extracted. The system compares the data with the current extract's data to determine if:

- Each record in the current load already exists in the data warehouse
- Each record is an update to an existing record in the data warehouse
- Each record is an entirely new record

The system compares extracted source data in the staging tables against the load control tables for updates to dimension data. Although the load control tables store source-specific reference data, the structure of the load control tables is source-independent. The aggregate table contains prestored summaries in the data warehouse. This feature supports commonly used metrics, and can improve query performance. The aggregation process uses an aggregate table that contains both facts and dimensions, representing a summary of base-level table records.

Extracting Data Posted at the Detail-Level for Oracle 11i

By default, the Siebel Analytics Enterprise Data Warehouse assumes that the posting from your journal to your General Ledger is done at the summary level, and that references are maintained in General Ledger for AP and AR subledgers. If import references are not maintained in General Ledger and the posting from AP and AR is at the detail-level, then modify the filter condition in M_I_GL_XACTS_JOURNALS_EXTRACT and disable the S_M_I_XACTS_IMP_GLRF_EXTRACT session so that only the session S_M_I_GL_XACTS_JOURNALS_EXTRACT loads into the common table TI_STAGE_GLRF_DERV.

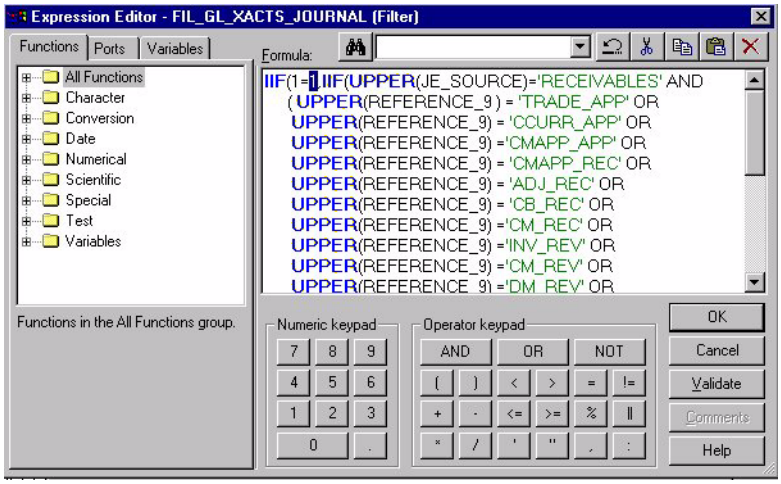
To modify the filter condition for posting at the detail level

1 In the PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 Open the M_I_GL_XACTS_JOURNALS_EXTRACT mapping.

3 Select the FIL_GL_XACTS_JOURNAL Expression transformation, and click the Properties tab to edit the filter condition.

To load postings at the detail level, replace the '1=2' condition with '1=1', as shown in the following figure.



4 Validate and save your changes to the repository.

To disable the session for General Ledger extract

1 In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i folder.

2 Expand the Oracle 11i Strategic Sourcing Application workflow.

3 Expand the O_FACTS_EXTRACT worklet and the O_Facts_Extract_Finance_Common_AP_AR_GL worklet

4 Double-click the S_M_I_XACTS_IMP_GLRF_EXTRACT session to open the Edit Tasks window.

5 Click Disable this task, and click OK.

6 Repeat Step 2 to Step 5 for the Oracle 11i Enterprise Sales Application and the Oracle 11i Finance Application workflows.

NOTE: As a best practice, you must move unused sessions to another folder to avoid error messages and preserve it for future use.

Fact Table Extract, Transform, and Load for SAP R/3 for Sales

The extract, transform and load (ETL) method used for SAP R/3 data depends on the granularity in which SAP R/3 presents the data, as well as whether or not you want to integrate the Sales module. Usually, SAP R/3 updates sale and purchase subledger data in General Ledger at the header/date level.

There are different sets of workflows to extract, apportion, and load the data into the various General Ledger tables in the data warehouse, depending on whether your data is posted at the header level or the detail level.

Fact Table ETL Process for Header-Level Sales Data for SAP R/3

The General Ledger module extracts invoice and sales order header-level data from the SAP R/3 tables BKPF (Accounting Document Header table) and BSEG (Accounting Document Segment table) and loads them into six staging tables. The entire ETL data flow is comprised of four areas:

- Extracting Sales Order and Invoice Data
- Loading Invoice and Sales Order Header Tables
- Apportioning Amounts and Quantities
- Loading General Ledger Staging Table

Each of these areas are discussed in the following sections.

Extracting Sales Order and Invoice Data

Before loading data at the header level for sales orders and invoice data, General Ledger extracts transactional data from the SAP R/3 tables BKPF and BSEG and loads it into three staging tables—General Ledger staging table (TS_STAGE_GL), Sales Order Header staging table (TS_STAGE_SO_HDR) and Invoice Header staging table (TS_STAGE_IV_HDR).

By default, only those transactions from BKPF and BSEG that have a status of NULL or 'S' are extracted. Null status implies that the transactions are already posted. The status 'S' marks the record as a noted item.

Loading Invoice and Sales Order Header Tables

In the load process, the Invoice Header and Sales Order Header tables are populated by different sources. The TS_STAGE_IV_HDR, IA_SALES_IVCLNS, and OD_SALES_IVCLNS tables load the Invoice Header table (TS_STAGE_GL_IHD). The TS_STAGE_IV_HDR supplies header records for invoice data. Using these headers, you can extract sales invoice line item data from IA_SALES_IVCLNS table and aggregate it to provide total amount (NET_DOC_AMT), total quantity (INVOICED_QTY), and total number of items (TOTAL_ITEMS) for each invoice. These aggregated amounts are stored in the TS_STAGE_GL_IHD header table.

The method in which General Ledger loads the Invoice Header table is similar to the loading of the Sales Order Header table. The TS_STAGE_SO_HDR, IA_SALES_ORDLNS, and IA_SALES_HIST tables load the Sales Order Header table (TS_STAGE_GL_SHD). The TS_STAGE_SO_HDR table supplies header records for sales order data. Using these headers, you can extract sales order line item data from IA_SALES_ORDLNS table and aggregate it to provide total amount (NET_DOC_AMT), total quantity (SALES_ORDER), and total number of items (TOTAL_ITEMS) for each sales order. These aggregated amounts are stored in the TS_STAGE_GL_SHD header table.

Apportioning Amounts and Quantities

The Invoice and Sales Order Header staging tables record total amounts and quantities for each document number (invoice number or sales order number). However, you must break down these amounts into different segments, so that you can load the appropriate amounts into the applicable staging table. There are six finance-related staging tables involved with apportioning amounts and quantities:

- Accounts Payable Transaction staging table (TS_AP_XACTS)
- Accounts Receivable Transaction staging table (TS_AR_XACTS)
- Tax Transaction staging table (TS_TAX_XACTS)
- General Ledger Cost of Goods Sold staging table (TS_GL_COGS)
- General Ledger Revenue staging table (TS_GL_REVENUE)
- General Ledger Other staging table (TS_GL_OTHER)

The General Ledger module prepackages logic to apportion header-level data to load the appropriate amounts into the corresponding staging table. The following examples illustrate the concept of apportioning data, by looking at how General Ledger apportions header-level invoice data to derive the detail-level amounts.

As previously stated, the General Ledger Header staging table (TS_STAGE_GL_IHD) stores header-level invoice data, while the IA_SALES_IVCLNS table stores line item invoice data, including the line item amounts for each invoice document. These line item amounts are aggregated by invoice number and loaded into the appropriate header record in the TS_STAGE_GL_IHD staging table.

Segment ratios in the General Ledger module apportion the header-level amounts into separate amounts based on each of the following segments—Revenue, Tax, and Freight. This allows the total amount of each invoice to be separated into those same segments of Revenue, Tax and Freight.

The segment ratios in General Ledger are created using two tables—TS_STAGE_GL and TS_STAGE_GL_IHD. The TS_STAGE_GL table stores total amounts for each segment for each order, and the TS_STAGE_GL_IHD stores the total amount for each Invoice. Revenue ratio equals the revenue amount divided by the total invoiced amount. Tax ratio equals the tax amount divided by the total invoiced amount. Freight ratio equals the freight amount divided by total invoiced amount.

The segmentation can vary in many ways depending on your business requirements. There can be a different number of segments, different segment types, or segments can be used to apportion quantities instead of amounts. In addition to the invoice data, the same concept also applies to sales order data that is posted at the header level.

Loading General Ledger Staging Tables

After the sales order and invoice data is apportioned, the data is loaded into the applicable staging table. Using a lookup performed in the TS_STAGE_FIN_STMT staging table on Company Code and General Ledger Account number, the General Ledger module determines the financial statement that each segment amount belongs to. The possible financial statements are Accounts Payable, Accounts Receivable, Tax, General Ledger Revenue, General Ledger Cost of Goods Sold, and General Ledger Others. Others are defined as any financial statements that do not belong to one of the other five defined categories.

When the lookup determines the type of financial statement item, it loads the data into the appropriate staging table.

NOTE: The General Ledger Balance ID in the fact tables must be the same as the Key ID in the General Ledger Balance table. Keeping these two column values the same verifies that the same granularity, or incremental level, is maintained. If there is a disparity, the resulting balances may be skewed and misinterpreted. Also, note that Accounts Receivable and Accounts Payable have different Balance IDs due to their distinct granularity. For information on configuring any of the three Balance IDs, see Configuring the Receivables Module on page 99.

Fact Table ETL Process for Detail-Level Information for SAP R/3

The General Ledger module uses the Group ID when data is posted at a detail level. The Group ID identifies one set of offsetting debits and credits. Each time the debits and credits are offset, the Group ID is reset.

In relation to General Ledger, the Group ID changes each time the debits and credits offset each other. Although the sales order number is the same, the Group ID changes. In this way, the Group ID relates a single record on the sales order side to multiple records in General Ledger. From the sales order perspective, the Group ID changes when the line item changes.

Similar to loading header-level data, the General Ledger module also loads detail-level data from the SAP R/3 tables BKPF and BSEG into six staging tables, then into their IA tables. However, the General Ledger module uses staging tables for detail-level data to determine three data streams—invoice, sales order, and others.

When loading data at the detail level, General Ledger first extracts and loads transactional data from the SAP R/3 tables BKPF and BSEG into the same three staging tables that it does for loading header level information—TS_STAGE_GL, TS_STAGE_SO_HDR, and TS_STAGE_IV_HDR. Before loading data into TS_STAGE_GL, the Group ID is generated by checking the debit and credit amounts.

In this next phase of loading detail-level information, the General Ledger module selects all invoice lines from IA_SALES_IVCLNS that exist in TS_STAGE_IV_HDR, and generates Group IDs for each combination of invoice and line items. A unique Group ID is created when the invoice line changes. The data is then loaded into the staging table TS_STAGE_GL_ILN.

For invoice lines data, sales invoice lines data is extracted from IA_SALES_INVLNS that exists in the TS_STAGE_IV_HDR table. The Group ID is selected from IA_SALES_HIST, and the data is then loaded into the staging table TS_STAGE_GL_SLN.

When creating header tables, the tables TS_STAGE_GL, TS_STAGE_IV_HDR, and TS_STAGE_GL_ILN join to get the keys, split the transactions, and set the Balance ID. In addition, a lookup is performed on the TS_STAGE_FIN_STMT staging table. The data then loads into the six staging areas, which are AP Transactions, AR Transactions, Tax Transactions, General Ledger Revenue, General Ledger Cost of Goods Sold, and General Ledger Others.

Similar to the Invoice Lines data, when creating header tables, the subitems of the sales order lines are created. The tables (TS_STAGE_GL, TS_STAGE_SO_HDR, and TS_STAGE_GL_SLN) join to retrieve the keys, split the transactions, and set the Balance ID. In addition, a lookup is performed on the TS_STAGE_FIN_STMT staging table. The data then loads into the six staging areas.

General Ledger Information for Nonsales Transactions for SAP R/3

Loading transaction information other than sales is handled in the same fashion as sales order lines and invoice lines data. All the transactions from TS_STAGE_GL that do not exist in TS_STAGE_IV_HDR are selected. Also the tables TS_STAGE_IV_HDR, TS_STAGE_SO_HDR, and TS_STAGE_GL join to get the keys, split the transactions and set the Balance ID. In addition, a lookup is performed on the TS_STAGE_FIN_STMT staging table. Data then loads into the six staging areas.

Extracting Data Posted at the Header Level for SAP R/3

The General Ledger module extracts sales information posted to the General Ledger at the detail level by default. However, this can be configured if your installation of SAP R/3 is configured to store data at the header level. For information on how the General Ledger loads data posted at the header level, see Fact Table ETL Process for Header-Level Sales Data for SAP R/3 on page 64.

To configure the fact extract to extract data posted at the header level

1. In PowerCenter Workflow Manager, open the Configuration for SAP R/3 folder.
2. Expand the SAP_EnterpriseSales_Application workflow and the Extract_Facts worklet.
3. Right -click on the worklet S_FACTS_EXTRACT_FINANCE_DETAIL_SALES_COMBINATION_ONLY, and select Edit.
4. In the Edit Tasks window, select the Disable this task check box.
5. Right -click on the worklet S_FACTS_EXTRACT_FINANCE_HEADER_SALES_COMBINATION_ONLY, and select Edit.
6. In the Edit Tasks window, clear the Disable this task check box.
7. Expand the SAP_EnterpriseSales_Application workflow and the Load_Facts worklet.
8. Right -click on the worklet S_FACTS_LOADS_FINANCE_DETAIL_SALES_COMBINATION_ONLY, and select Edit.
9. In the Edit Tasks window, select the Disable this task check box.

10 Right -click on the worklet S_FACTS_LOADS_FINANCE_HEADER_SALES_COMBINATION_ONLY, and select Edit.

11 In the Edit Tasks window, clear the Disable this task check box.

Process of Configuring the General Ledger for Oracle 11i

The following list shows the procedures that you use to configure the General Ledger module for Oracle 11i:

- Configuring the Financial Statement Item Categorization on page 68
- Filtering Extracts Based on Set of Books ID on page 70
- Configuring General Ledger Transaction Extracts on page 71
- Configuring General Ledger COGS Extract on page 72
- Configuring the General Ledger Account Hierarchies on page 73

Configuring the Financial Statement Item Categorization

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

The Financial Statement Item code categorizes each General Ledger account record in the IA_GL_ACCOUNTS dimension table. Each General Ledger account is assigned to one of the possible financial statement items—General Ledger Revenue, General Ledger COGS, Tax Transactions, AR Transactions, AP Transactions, or General Ledger Others. Any account that is not assigned to a financial statement item is automatically assigned to General Ledger Others. The logic for assigning the accounts is located in the FIN_STMT_CODES_ORA.CSV file. For example, this file might have the layout shown in Table 13.

Table 13. Sample Layout of FIN_STMT_CODES_ORA.CSV File

Set of Books ID	From Account	To Account	Financial Statement Item
1	1100	1190	REVENUE
1	1200	1250	AR
1	1260	1270	COGS
1	1280	1290	COGS
1	1300	1300	AR
1	2000	2200	AP
1	2300	2400	TAX

Table 13. Sample Layout of FIN_STMT_CODES_ORA.CSV File

Set of Books ID	From Account	To Account	Financial Statement Item
1	3000	3100	TAX
21	1200	1250	AR
21	1260	1300	COGS
21	1400	1400	COGS
42	2400	2500	AP
62	2200	2300	AP
62	2500	2600	REVENUE
102	31000	32000	AR
102	22000	23000	AP
102	41000	41000	REVENUE

In Table 13, in the first row, all accounts within the account number range from 1100 to 1190 containing a Set of Books (SOB) ID equal to 1 are assigned to Revenue. Similarly, looking at the second row in the table, all accounts with the Set of Book ID equal to 1 and within the account number range from 1200 to 1250 are assigned to Accounts Receivable. Each row categorizes all accounts within the specified account number range and with the given Set of Books ID.

NOTE: Because different sets of books can contain the same account numbers, the Set of Books ID is also contained in this file. By keeping note of the account number as well as the Set of Books ID, you can uniquely identify each transaction in the data warehouse.

If your account numbers are categorized differently from what is prepackaged in the FIN_STMT_CODES_ORA.CSV file, then you must modify the file to accurately categorize your accounts.

NOTE: When you specify the FIN STMT ITEM, you must capitalize the letters and only use the following values—REVENUE, COGS, AR, AP, TAX, and OTHERS.

To assign accounts to different financial statement items

1. Navigate to `$pmserver\srcfiles\FIN_STMT_CODES_ORA.CSV`.
2. Open the FIN_STMT_CODES_ORA.CSV file.
3. Modify the `TO ACCT` and `FROM ACCT` ranges as necessary.
4. Save and close the FIN_STMT_CODES_ORA.CSV file.

   For information on uploading this information, see Chapter 4, "Initializing and Populating the Siebel Analytics Enterprise Data Warehouse."

Filtering Extracts Based on Set of Books ID

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

If you have multiple sets of books and want to use only some of them as sources for the extract, then you have to modify the Source Qualifier. For example, assume that you have four sets of books for your enterprise: a set of books for your U.S. organization (SOB_ID = 1), a set of books for your Japan organization (SOB_ID = 2), a set of books for your German organization (SOB_ID = 3), and a set of books for your enterprise as a whole (SOB_ID = 4). If you want to extract only the enterprise level information, you extract only transactions where the SOB_ID = 4. Therefore, in the Source Qualifier's SQL Query and User Defined Join fields, you must add the following filter statement:

```
'AND SET_OF_BOOKS_ID IN (4)'
```

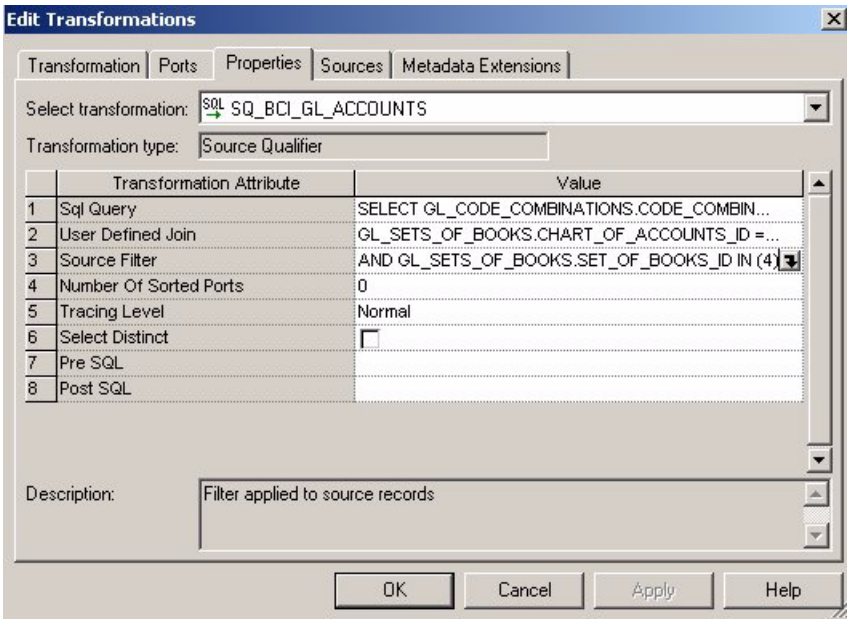
To filter extracts based on Set of Books ID

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the `MPLT_BCI_GL_ACCOUNTS`.
3. Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

   Insert the filter condition in the SQL Query field and in the User Defined Join field. For example, if you want to use only the Set of Books whose ID is '4', then you insert the following filter:

   ```
   AND GL_SETS_OF_BOOKS.SET_OF_BOOKS_ID IN (4)
   ```

   in the WHERE clause in the SQL Query field, and at the end of the statement in the User Defined Join field, as shown in the following figure.



4 Validate and save changes to the repository.

Configuring General Ledger Transaction Extracts

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

There are two separate transaction extracts for General Ledger—General Ledger Revenue and General Ledger COGS. By default, the General Ledger module extracts only Completed revenue and COGS that have been posted to the general ledger. Completed revenue transactions are those where the RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG = 'Y'. If you want to extract incomplete revenue transactions, you can remove the filter in the Business Component.

You must modify both the regular mapplet (MPLT_BCI_GL_REVENUE) as well as the primary extract mapplet (MPLT_BCI_GL_REVENUE_PRIMARY).

To modify the extract filter for General Ledger Revenue

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

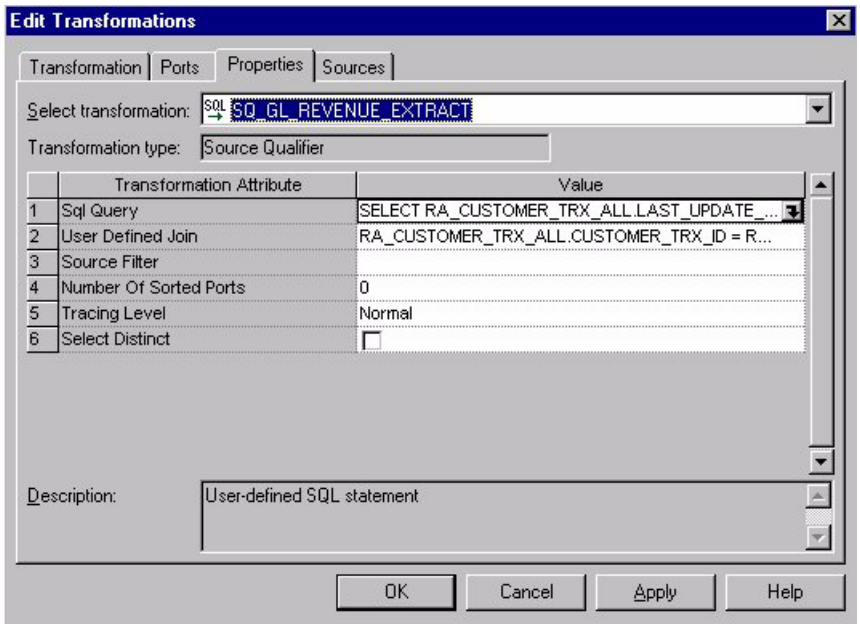
2 In Mapplet Designer, open the MPLT_BCI_GL_REVENUE mapplet.

3 Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

In the User Defined Join field and in the SQL Query field, remove the condition:

```
AND RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG = 'Y'
```

This step is shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_BCI_GL_REVENUE_PRIMARY mapplet.

Configuring General Ledger COGS Extract

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

By default, the General Ledger module extracts only COGS transactions that have been posted to the general ledger. All COGS transactions that have been transferred satisfy the following condition—`MTL_TRANSACTION_ACCOUNTS.GL_BATCH_ID <> -1`. If you want to extract all transactions, you can remove the filter in the Business Component mapplet.

Because General Ledger never deletes posted transactions, there are no prebuilt primary extract and delete mappings for COGS data. Therefore, if you decide to remove this filter and begin extracting unposted transactions, you must also create primary extract and delete mappings similar to those used for General Ledger Revenue. You can use the primary extract mapping (`M_I_GL_REVENUE_PRIMARY_EXTRACT`), and the delete mapping (`M_I_GL_REVENUE_DELETE`) as models for the extract and delete mappings you are creating in Oracle 11i.

To modify the extract filter for General Ledger COGS

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In Mapplet Designer, open MPLT_BCI_GL_COGS.

3 Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

In the User Defined Join field and in the SQL Query field, remove the condition:

```
AND MTL_TRANSACTION_ACCOUNTS.GL_BATCH_ID <> -1
```

4 Validate and save your changes to the repository.

Configuring the General Ledger Account Hierarchies

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

The Siebel Analytics Enterprise Data Warehouse prepackages a hierarchy table (IA_HIERARCHIES) to store hierarchy information. The General Ledger module uses this table to store General Ledger account hierarchy information. To make sure the hierarchical information is correct, you must accomplish three tasks:

1 Load the General Ledger Accounts data into the IA_GL_ACCOUNTS table.

2 Load all hierarchies that are configured in your source system into the IA_HIERARCHIES table.

3 Configure mappings and sessions to update the following hierarchy columns in the IA_GL_ACCOUNTS table—HIER1_KEY, HIER2_KEY, HIER3_KEY, HIER4_KEY, HIER5_KEY, and HIER6_KEY.

To load the General Ledger Accounts data, run the S_M_I_GL_ACCOUNTS_LOAD session for Oracle 11i. This session is part of the prepackaged workflow. Therefore, if you run the prepackaged workflow successfully, this information should already be populated in the General Ledger Account table.

Each of these steps is discussed in detail in the following sections.

To load the General Ledger Accounts data

1 In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i folder.

2 Expand B_I_DIM_LOAD, then expand B_I_DIM_LOAD_FIRST.

3 Select S_M_I_GL_ACCOUNTS_LOAD.

4 Select Server Requests > Start, or click the Start button.

Loading Hierarchies into the IA_HIERARCHIES Table

This task is a step in the Process of Configuring the General Ledger for Oracle 11i on page 68.

The General Ledger module prepackages M_I_GL_HIER_DERIVE mapping to extract hierarchy definitions from Oracle 11i. To better assist you in loading your hierarchy information into the IA_HIERARCHIES table, Table 14 gives you descriptions of the columns presented in the table.

Table 14. Columns in the IA_HIERARCHIES Table

Column Name	Description
HIER_KEY	This surrogate key is generated for each hierarchy. This key must be linked to the HIERARCHY_KEY column in the IA_GL_ACCOUNTS table.
HIERARCHY_ID	This column uniquely identifies the hierarchy within a given category. For general ledger account hierarchies, the ID format is as follows: 'GL_HIER' || <top node of hierarchy> || HIER1_CODE... HIER10_CODE
HIER_CODE	This code represents a hierarchy. The hierarchy code defines the name of the hierarchy (for example, Balance Sheet, Profit and Loss, and so on).
HIER_NAME	This column provides the name of the entire hierarchy.
HIER_CAT_CODE	This code represents the category of the hierarchy. For general ledger account hierarchies, the category code is GL_HIER.
HIER_CAT_DESC	This is a description of the category to which the hierarchy belongs. For general ledger account hierarchies, the category description is General Ledger Hierarchy.
HIER_SUBCAT_CODE	This is the code of the subcategory of the hierarchy.
HIER_SUBCAT_DESC	This is a description of the subcategory of the hierarchy.
LEVEL_MIN_NUM	This is the starting number, which is set to '0'. For general ledger accounts, this is the beginning account number that is associated with this hierarchy. All accounts including and between the LEVEL_MIN_NUM and LEVEL_MAX_NUM are included in this category.
LEVEL_MAX_NUM	This is the last number, which is set to 99999999. For general ledger accounts, this is the ending account number that is associated with this hierarchy. All accounts including and between the LEVEL_MIN_NUM and LEVEL_MAX_NUM are included in this category.
HIER[X]_CODE	Each of these code columns represents a level in the hierarchy, where [X] denotes the level. HIER1 is the highest level of the hierarchy, and HIER20 is the lowest. Each code name column (HIER[X]_NAME) corresponds to a code column (HIER[X]_CODE). The code of the highest level of the hierarchy is catenated with 'GL_ACCT' to form the hierarchy of the IA_HIERARCHIES table.

Table 14. Columns in the IA_HIERARCHIES Table

Column Name	Description
HIER[X]_NAME	Each of these code name columns represents a level in the hierarchy, where X denotes the level. Hier1 is the highest level of the hierarchy, and HIER20 is the lowest. Each code name column (HIERX_NAME) corresponds to a code column (HIERX_CODE).
HIER_ATTR[X]_CODE and HIER_ATTR[X]_NAME	There are five sets of extension columns for code name pairs. The X represents the level, where the same level is shared by each code name pair, such as HIER_ATTR1_CODE and HIER_ATTR1_NAME
HIER_ATTR[X]_Text	These are extension columns to store additional text. There are three available for your use.

The Hierarchy ID is set for every unique hierarchy structure. The format is:

'GL_HIER' ~ HIER_CODE ~ HIER1_CODE... HIER20_CODE

where:

- GL_HIER specifies that the hierarchy applies to general ledger accounts
- HIER_CODE specifies the ID for a hierarchy structure
- HIER1_CODE ~ HIER2_CODE... HIER20_CODE specifies each of the unique hierarchy levels in the given hierarchy structure

For example, if one of the General Ledger account hierarchies has the following structure:

Account (A)=> Current Asset (CA) => Fixed Asset (FA) => Balance Sheet (BS)

where the Balance Sheet hierarchy level is the highest level of the hierarchy, and BS denotes the hierarchy code, then set the Hierarchy ID to:

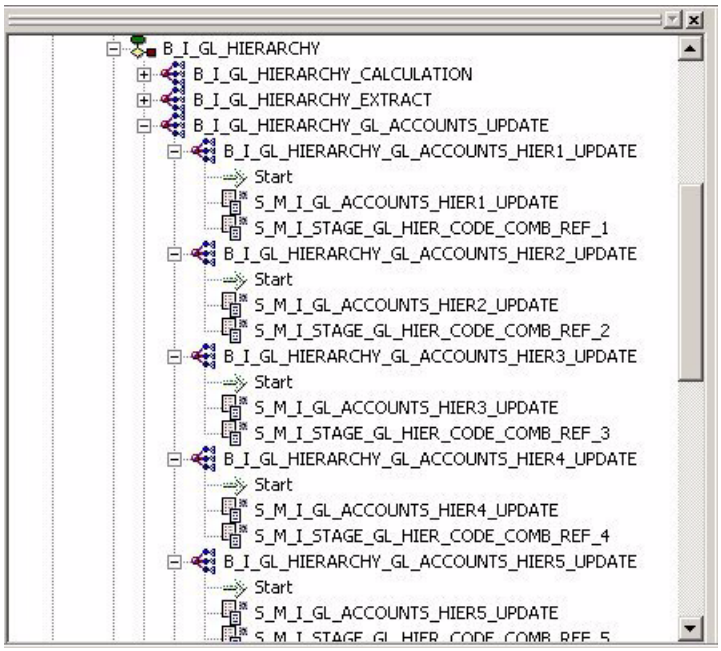
'GL_HIER~BS~FA~CA~A'

After you load the IA_HIERARCHIES table, you then must update the IA_GL_ACCOUNT table with the hierarchy information. Updating the IA_GL_ACCOUNT table requires a two-step process.

To update hierarchies in the IA_GL_ACCOUNTS table

1 You must first determine which hierarchies apply to each General Ledger Account.

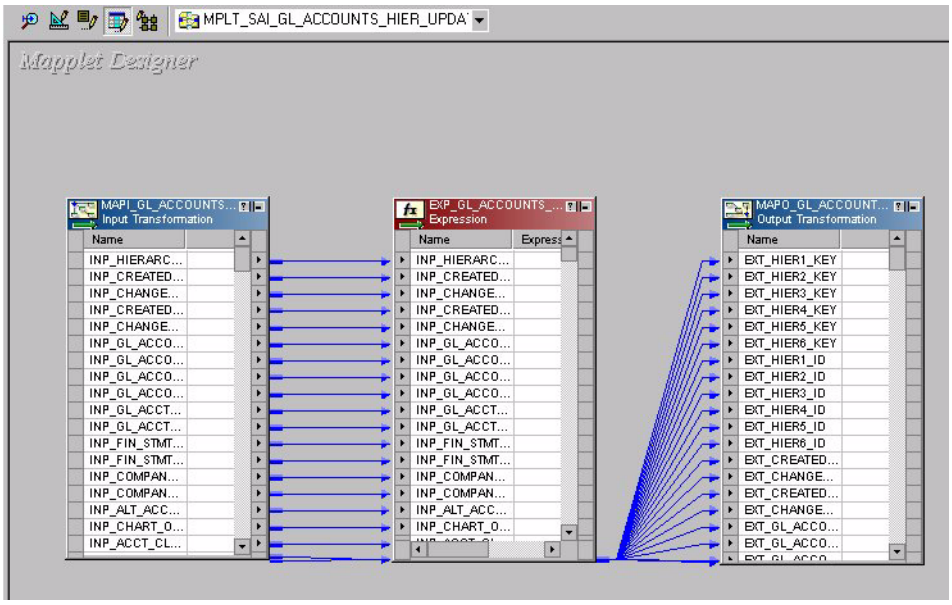
To make this determination, you must modify the S_M_I_STAGE_GL_HIER_CODE_COMB_REF_[X] sessions, ([X] denotes the hierarchy number in the IA_GL_ACCOUNTS table). Because there are a maximum of six possible hierarchy structures in the IA_GL_ACCOUNT table, there are six sessions, where each session locates information for exactly one hierarchy structure, as shown in the following figure.



Each of these sessions derives the General Ledger Account references for a particular hierarchy, which is later used to update the HIER[X]_KEY column in the IA_GL_ACCOUNTS table. The 'HIER[X]_KEY' refers to the HIER_KEY surrogate key in the IA_HIERARCHIES table.

2 Load the appropriate hierarchy structure in the HIER[X]_KEY columns in the IA_GL_ACCOUNT table.

To load this hierarchy structure, you must modify the MPLT_SAI_GL_ACCOUNTS_HIER_UPDATE Source Adapter mapplet in the M_I_GL_ACCOUNTS_HIER_UPDATE mapping in Oracle 11i, as shown in the following figure.



To configure hierarchy code in the session

1 In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i folder.

2 Expand Oracle11i_Finance_Application_GL_ACCOUNT_Hierarchy/ W_O_GL_HIERARCHY_GL_ACCOUNTS_UPDATE/W_O_GL_HIERARCHY_GL_ACCOUNTS_HIER[X]_UPDATE.

3 In Worklet Designer, right-click the W_O_GL_HIERARCHY_GL_ACCOUNTS_HIER[X]_UPDATE and click Open.

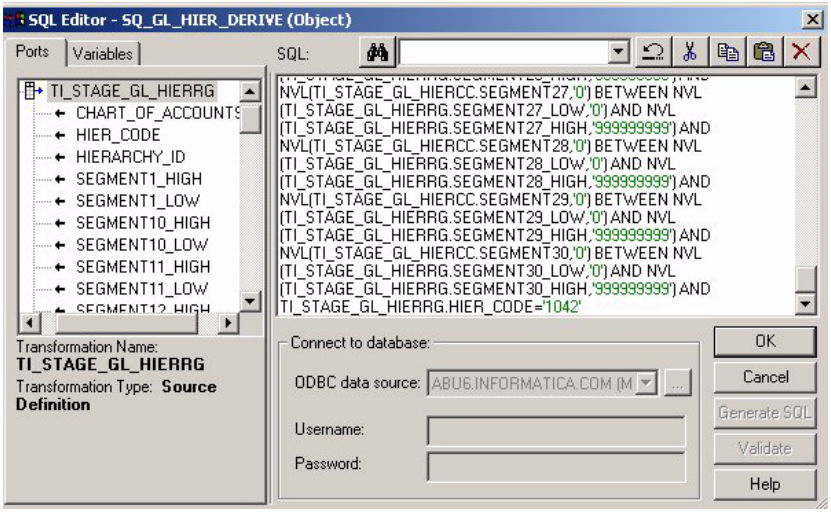
4 Double-click the appropriate session (S_M_I_STAGE_GL_HIER_CODE_COMB_REF_[X]) to open the Edit Tasks window.

5 In the Transformations tab, edit the SQL Query field by replacing the default hierarchy code.

The last override condition in the SQL contains the hierarchy code. The hierarchy code determines the hierarchy for which the mapping session calculates the General Ledger references. For example, in the S_M_I_STAGE_GL_HIER_CODE_COMB_REF_1 session, the condition:

```
TI_STAGE_GL_HIERRG.HIER_CODE='1042'
```

calculates all the General Ledger account references for the hierarchy code '1042'. This code is the AXIS_SET_ID defined in the Oracle Applications source table, RG_REPORT_AXIS_SETS. Depending on which hierarchy you want to store in the HIER1_KEY column in the IA_GL_ACCOUNTS table, the corresponding HIER_CODE must be set in the SQL override condition, as shown in the following figure.



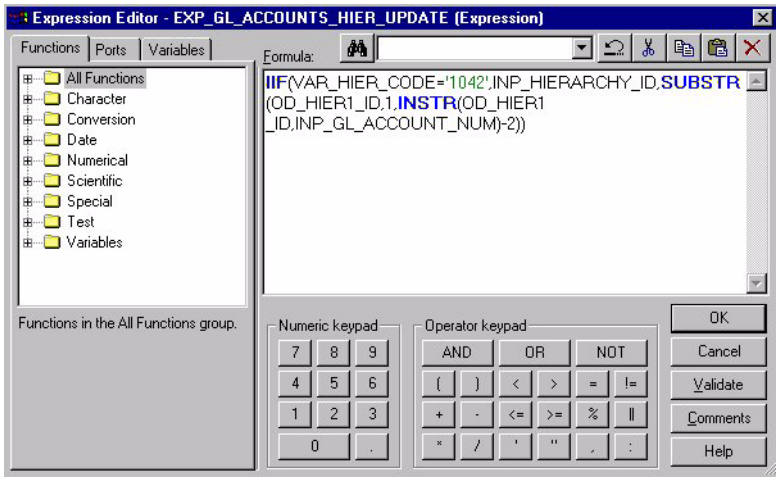
To configure the hierarchy code in the update mapping

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_SAI_GL_ACCOUNTS_HIER_UPDATE Source Adapter mapplet.
3. Double-click the Expression transformation to open the Edit Transformations window, and click the Ports tab.
4. Edit the EXT_HIER[X]_ID port.

   The value of each field must be the same as the set in the corresponding mapping—M_I_STAGE_GL_HIER_CODE_COMB_REF_X. For example, if you want to change the EXT_HIER1_ID port, then change the expression:

```
IIF(VAR_HIER_CODE='1042',INP_HIERARCHY_ID,SUBSTR(OD_HIER1_ID,1,INSTR(OD_HIER1_ID,I
NP_GL_ACCOUNT_NUM)-2))
```

In this case, you replace the '1042' with the applicable code used in the HIER_CODE column of the IA_HIERARCHIES table, as shown in the following figure.



5 Validate and save your changes to the repository.

Process of Configuring the General Ledger Module for SAP R/3

The following section provides configuration information for the General Ledger module for SAP R/3. The Siebel Analytics Enterprise Data Warehouse prepackages configuration that focuses on the detail level. For more information based on header level detail, see Extracting Data Posted at the Header Level for SAP R/3 on page 67.

To configure General Ledger for SAP R/3, perform the following tasks:

- Configuring the General Ledger Account Hierarchies on page 79
- Configuring Hierarchy ID in Source Adapter on page 81
- Configuring the General Ledger Balance Extract on page 82

Configuring the General Ledger Account Hierarchies

This task is a step in the Process of Configuring the General Ledger Module for SAP R/3 on page 79.

The General Ledger module prepackages a hierarchy table (IA_HIERARCHIES) to store all hierarchy information. The General Ledger module uses this table to store General Ledger account hierarchy information. For the hierarchies to work, you first have to load the General Ledger account hierarchy information into IA_HIERARCHIES. Second, you have to configure the Source Adapter for the GL_ACCOUNTS_LOAD mapping so that the hierarchy key in the IA_GL_ACCOUNTS table links to the IA_HIERARCHIES table. After these two tasks are accomplished, you are ready to load General Ledger Account data into the IA_GL_ACCOUNTS table.

You can relate up to six possible hierarchy structures to each general ledger account, using the IA_HIERARCHIES table:

- Load the primary hierarchy in the IA_GL_ACCOUNTS table.
- If you want additional hierarchies, you begin by adding hierarchies to the HIERX_KEY ports.
- Load the first additional hierarchy in the HIER1_KEY port, and then load the following hierarchy structure in the HIER2_KEY port, and the other hierarchies in sequence.

Loading the Hierarchy Definitions Into the IA_HIERARCHIES Table, The General Ledger module prepackages the M_S_GL_HIERARCHY_EXTRACT mapping to extract hierarchy definitions from SAP R/3. To better assist you in loading your hierarchy information into the IA_HIERARCHIES table, Table 15 provides descriptions of the columns presented in the table.

Table 15. Columns in the IA_HIERARCHIES Table

Column Name	Description
HIER_KEY	This surrogate key is generated for each hierarchy. This key must be linked to the HIERARCHY_KEY column in the IA_GL_ACCOUNTS table.
HIERARCHY_ID	This column uniquely identifies the hierarchy within a given category. For general ledger account hierarchies, the ID format is as follows: 'GL_HIER' || <top node of hierarchy> || HIER1_CODE... HIER10_CODE
HIER_CODE	This code represents a hierarchy. The hierarchy code defines the name of the hierarchy (for example, Balance Sheet, Profit and Loss, and so on).
HIER_NAME	This column provides the name of the entire hierarchy.
HIER_CAT_CODE	This code represents the category of the hierarchy. For general ledger account hierarchies, the category code is GL_HIER.
HIER_CAT_DESC	This is a description of the category to which the hierarchy belongs. For general ledger account hierarchies, the category description is General Ledger Hierarchy.
HIER_SUBCAT_CODE	This is the code of the subcategory of the hierarchy.
HIER_SUBCAT_DESC	This is a description of the subcategory of the hierarchy.
LEVEL_MIN_NUM	This is the first number, which is set to '0'. For general ledger accounts, this is the first account number that is associated with this hierarchy. All accounts including and between the LEVEL_MIN_NUM and LEVEL_MAX_NUM are included in this category.

Table 15. Columns in the IA_HIERARCHIES Table

Column Name	Description
LEVEL_MAX_NUM	This is the last number, which is set to 99999999. For general ledger accounts, this is the last account number that is associated with this hierarchy. All accounts including and between the LEVEL_MIN_NUM and LEVEL_MAX_NUM are included in this category.
HIERX_CODE	Each of these code columns represents a level in the hierarchy, where [X] denotes the level. HIER1 is the highest level of the hierarchy, and HIER20 is the lowest. Each code name column (HIER[X]_NAME) corresponds to a code column (HIER[X]_CODE). The code of the highest level of the hierarchy is catenated with GL_ACCT to form the hierarchy of the IA_HIERARCHIES table.
HIER_ATTRX_CODE and HIER_ATTRX_NAME	There are five sets of extension columns for code name pairs. The X represents the level, where the same level is shared by each code name pair, such as HIER_ATTR1_CODE and HIER_ATTR1_NAME.
HIER_ATTRX_TEXT	These are extension columns to store additional text. There are three available for your use.

You must set the Hierarchy ID for every hierarchy structure. The following format is recommended:

```
'GL_ACCT' || <Hierarchy Code for top node of hierarchy structure>
```

where:

- GL_ACCT specifies that the hierarchy applies to general ledger accounts
- <Hierarchy Code for top node of hierarchy> is the hierarchy code that specifies the highest level of the hierarchy

For example, if one of the general ledger account hierarchies has the following structure:

```
Account (A)=> Current Asset (CA) => Fixed Asset (FA) => Balance Sheet (BS)
```

where the Balance Sheet hierarchy level is the highest level of the hierarchy, and BS denotes the hierarchy code, then set the Hierarchy ID to:

'GL_HIER~BS~FA~CA~A'

Configuring Hierarchy ID in Source Adapter

This task is a step in the Process of Configuring the General Ledger Module for SAP R/3 on page 79.

To configure the General Ledger module to link to the General Ledger Account hierarchies from the IA_HIERARCHIES table, you must configure the Source Adapter in the GL_ACCOUNTS_LOAD mapping. By default, the HIERARCHY_KEY port is set to NULL. To populate the General Ledger account hierarchies when you load the IA_GL_ACCOUNTS table, use the same hierarchal structure that you used in IA_HIERARCHIES. For more information on the hierarchal structure used in IA_HIERARCHIES, see Configuring the General Ledger Account Hierarchies on page 79.

To configure the Hierarchy ID

1. In PowerCenter Designer, open the Configuration for SAP R/3 folder.
2. In Mapping Designer, open the mapping M_S_GL_ACCOUNTS_LOAD.
3. Double-click the Expression transformation to open the Edit Transformations window.
4. Edit the HIERX_KEY port (X denotes a distinct hierarchy structure).
5. Change NULL to reflect the Hierarchy ID specified in the HIERARCHY_KEY column in the IA_HIERARCHIES table.
6. Validate and save your changes to the repository.

Configuring the General Ledger Balance Extract

This task is a step in the Process of Configuring the General Ledger Module for SAP R/3 on page 79.

The Balance and Key IDs set the grain at which you want to maintain the balances. There are three different IDs—General Ledger Balance ID, Accounts Payable Key ID, and Accounts Receivable Key ID. The default configurations are set to the most representative grains for maintaining the three different balances.

By default, the General Ledger Balance ID (BALANCE_ID) is configured as follows:

```
Account_number|| '~'||Company_code||'~'||Business_area||'~'||Client
```

Therefore, the General Ledger Account, Company Code, Business Area, and Client Code maintain the General Ledger Balances. You use the Client Code to distinguish between different instances of SAP R/3. For example, if you are running one instance of SAP R/3 for your U.S. business, and another instance of SAP R/3 for your Japan business, you may have the same General Ledger Account numbers in each system referring to different accounts. To distinguish the same General Ledger Account numbers in different instances, the grain of the balance includes the Client Code.

To change the grain at which you accumulate the General Ledger balance, modify the Balance ID or Key ID definition in the Expression transformation in the applicable mapping. Note that there are two sets of mappings—the balance extract mapping and the initial fact load mapping. The extract moves the balance from the source to staging tables, and the initial fact load mappings move the data from staging tables to the data warehouse. The following procedure provides instructions on how to configure the balance extract.

To configure the Balance Extract for the General Ledger Balance

1. In PowerCenter Designer, open the Configuration for SAP R/3 folder.
2. In Mapping Designer, open the M_S_GL_BALANCE_EXTRACT mapping.
3. Double-click the Expression transformation to open the Edit Transformations window.
4. Edit the BALANCE_ID.

   Make sure the General Ledger Balance ID in the extract mapping has the same precision as the General Ledger Balance ID in the fact load mapping.

5 Validate and save your changes to the repository.

7 Configuring the Inventory Module

After the Inventory module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of the Inventory Module

The Inventory module allows you to analyze your organization's inventory to maintain a tight operating and audit control. Inventory is a considerable cost to most organizations, but it is important to adequately serve customer sales. Managing and minimizing your organization's inventory investment can reduce costs and ultimately contribute to the competitiveness of your organization. Because operating at the lowest cost while maintaining customer service allows for competitive advantage, minimizing your inventory costs can help your organization find use in the marketplace.

The Inventory module has four functional areas:

- Bill of Materials (BOM)
- Activity
- Balances
- Customer and Supplier Returns

The Bill of Materials (BOM) functional area allows you to determine the profit margin of the components that comprise the finished goods. BOM allows you to keep up with the most viable vendors in terms of cost and profit, and to keep your sales organization aware of product delivery status, including shortages.

Activity functional area allows you to analyze the various types of events and tasks that occur. Examples of these activities include tracking inventory by type of movement. For example, transfer, issues, receipts, returns, sales, and so on. It allows the user to understand the impact of these activities on business operations, and allows the identification of problematic trends early. For example, large quantities of product in-transit.

The Balances functional area allows you to analyze the inventory held by an organization in relation to a number of different dimensions. For example, Product type, Product number, Storage location, Plant, Consigned Inventory, Restricted, and so on. It allows the user the ability to understand and determine the optimal distribution of assets as well as identify potential issues such as unnecessary build up of inventories.

The Customer and Supplier Returns functional area allows the user to specifically monitor the return of product by both Customers and Suppliers. At a Product level, it allows the user to identify early, potential, Customer-satisfaction issues relating to problematic Suppliers and Product.

For detailed information on the Inventory module, see the *Siebel Analytics Enterprise Applications User Guide*.

Process of Configuring the Inventory Module for Oracle 11i

This section contains configuration information that is specific to the Inventory module for Oracle 11i.

To configure the Inventory module for Oracle 11i, you can perform the following tasks:

- Configuring Quantity Types for Product Transactions for Oracle 11i on page 87:
- Configuring the Region Name on page 88
- Configuring the State Name on page 90
- Configuring the Country Name on page 91
- Configuring the Make-Buy Indicator on page 92

Related Topic

About Configuring the Handling of Currency Types on page 88

Configuring Quantity Types for Product Transactions for Oracle 11i

This task is a step in the Process of Configuring the Inventory Module for Oracle 11i on page 87.

Oracle 11i categorize quantities into different types—Goods Received quantities, Delivery quantities, and Base quantities:

- *Goods Received quantity* refers to the number of goods received.
- *Delivery quantity* refers to the number of goods delivered.
- *Base quantity* can be any transaction quantity.

The Siebel Analytics Enterprise Data Warehouse extracts the transaction type and loads this value into the XACT_SRC_TYPE column. In this column, the value 1 denotes a Goods Received quantity, and 2 denotes a Delivery quantity.

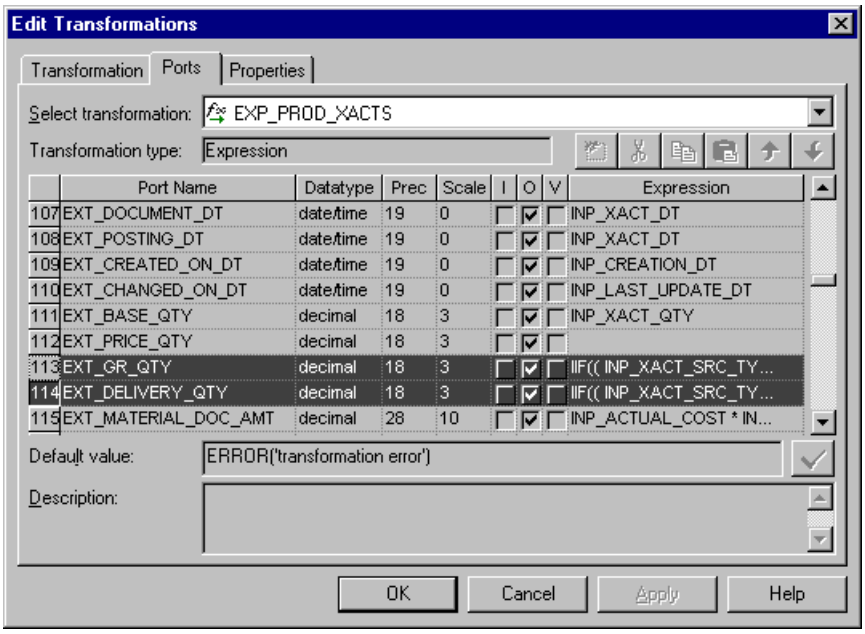
All quantities extracted from the source system are always loaded into the Base quantity column (EXT_BASE_QTY). However, only the receipt quantity is loaded into the Goods Received quantity column (EXT_GR_QTY), and only delivered quantities are loaded into the Delivery quantity column (EXT_DELIVERY_QTY).

If your definition of goods received or delivery quantity is different from the prepackaged condition, then you can edit the condition to suit your business needs.

To configure the Quantity type

1. In PowerCenter Designer, open the Configuration for Oracle Applications 11i folder.
2. Open the MPLT_SAI_PROD_XACTS mapplet.

3 Double-click the Expression transformation to open the Edit Transformations dialog box, and click the Port tab to display the EXT_GR_QTY and EXT_DELIVERY_QTY port, as shown in the following figure.



4 Edit the quantity types by substituting your desired condition for the prepackaged expression.

5 Click Apply.

6 Validate the mapplet, and save your changes to the repository.

About Configuring the Handling of Currency Types

The Inventory module, like all other modules, uses the same method for handling currency conversions from document currency to local and group currencies. The only distinction is how currency is handled for product transactions. In a product transaction, if the document currency code is not supplied, the ADI uses the same code that it uses for the local currency code, and the local currency is derived using the Set of Books ID.

This guide provides a functional overview of how local and group currencies are derived depending on what data is supplied to the Siebel Analytics Enterprise Data Warehouse. For more information on how to configure various components that relate to local, document, and group currencies, see Working with Document, Local, and Group Currencies on page 246.

Configuring the Region Name

This task is a step in the Process of Configuring the Inventory Module for Oracle 11i on page 87.

For Oracle 11i, you can reconfigure the region, state, and country names. This configuration information applies only to plant, storage, and supplier locations. By default, the Region Name column (EXT_REGION_NAME) is populated using the same code value as the Region Code column (EXT_REGION_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied region name instead of the code. If you want to reconfigure the load in this manner, you can load the region code and region name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Codes Lookup on page 258.

When you have loaded the region code and region name into the IA_CODES table, you can remove the expression in the Source Adapter that defines the Region Name column. By making the Region Name's expression blank, the ADI looks up the Region Name in the IA_CODES table, using the supplied region code when the load occurs. The load mapping then inserts the region name and region code into the data warehouse table.

To configure the Region Name

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the mapplet you want to edit.

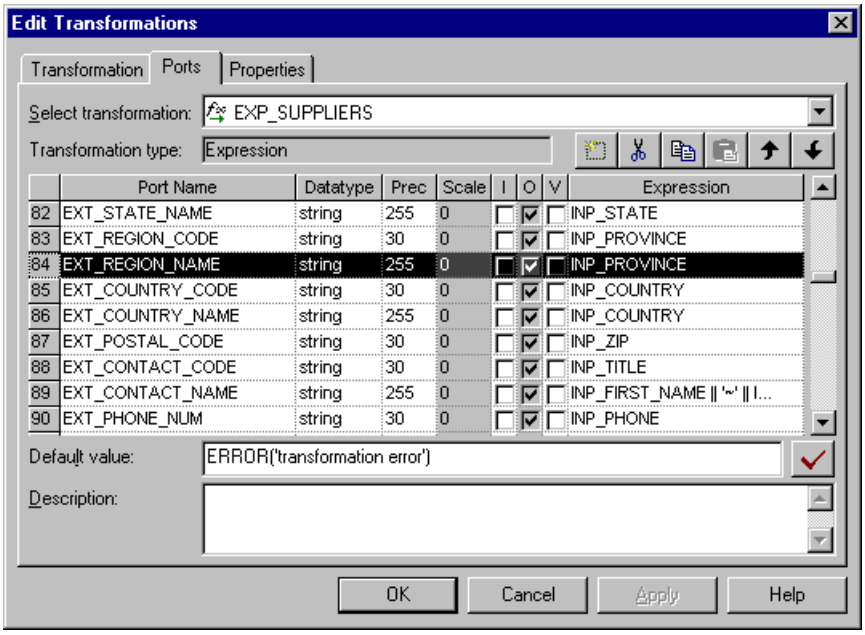
   The following is a list of all Source Adapter mapplets that use the EXT_REGION_NAME column:

   MPLT_SAI_SUPPLIERS

   MPLT_SAI_BUSN_LOCS_PLANT

   MPLT_SAI_BUSN_LOCS_STORAGE_LOC

3 Double-click the Expression transformation to open the Edit Transformations dialog box, and click the Port tab to display the EXT_REGION_NAME port, as shown in the following figure.



4 Edit the condition by removing the assigned value if you want the lookup to occur.

5 Click Apply.

6 Validate the mapplet, and save your changes to the repository.

Configuring the State Name

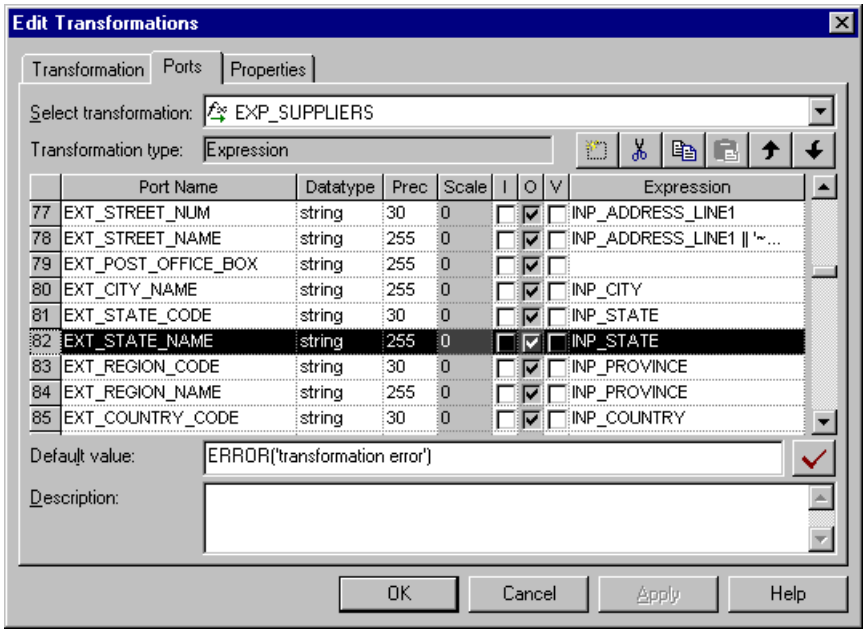
This task is a step in the Process of Configuring the Inventory Module for Oracle 11i on page 87.

For Oracle 11i, you can reconfigure the region, state, and country names that apply to the Supplier locations only. By default, the State Name column (EXT_STATE_NAME) is populated using the same code value as the State Code column (EXT_STATE_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied state name instead of the code. If you want to reconfigure the load in this manner, you can load the state code and state name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Codes Lookup on page 258.

When you have loaded the state code and state name into the IA_CODES table, you can remove the Expression in the Source Adapter that defines the State Name column. By setting the State Name's expression to null, the ADI looks up the state name in the IA_CODES table using the supplied state code, during the load process. The load mapping then inserts the state name and state code into the data warehouse table.

To configure the State Name

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SUPPLIERS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations dialog box, and click the Port tab to display the EXT_STATE_NAME port, as shown in the following figure.



4. Edit the condition by removing the assigned value if you want the lookup to occur.
5. Click Apply.
6. Validate the mapplet and save your changes to the repository.

Configuring the Country Name

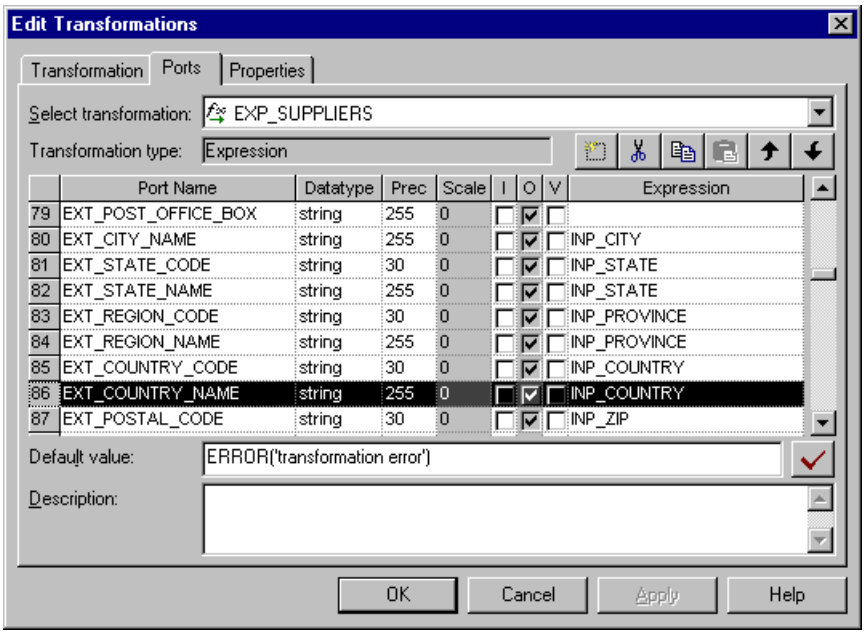
This task is a step in the Process of Configuring the Inventory Module for Oracle 11i on page 87.

For Oracle 11i, you can reconfigure the region, state, and country names that apply to supplier locations only. By default, the Country Name column (EXT_COUNTRY_NAME) is populated using the same code value as the Country Code column (EXT_COUNTRY_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied country name instead of the code. If you want to reconfigure the load in this manner, you can load the country code and country name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Codes Lookup on page 258.

When you have loaded the country code and country name into the IA_CODES table, you can remove the expression in the Source Adapter that defines the Country Name column. By setting the Country Name's expression to null, when the load occurs, the ADI looks up the country name in the IA_CODES table, using the supplied country code. The load mapping then inserts the country name and country code into the data warehouse table.

To configure the Country Name

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SUPPLIERS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations dialog box, and click the Port tab to display the EXT_COUNTRY_NAME port, as shown in the following figure.



4. Edit the condition by removing the assigned value if you want the lookup to occur.
5. Click Apply.
6. Validate the mapplet, and save your changes to the repository.

Configuring the Make-Buy Indicator

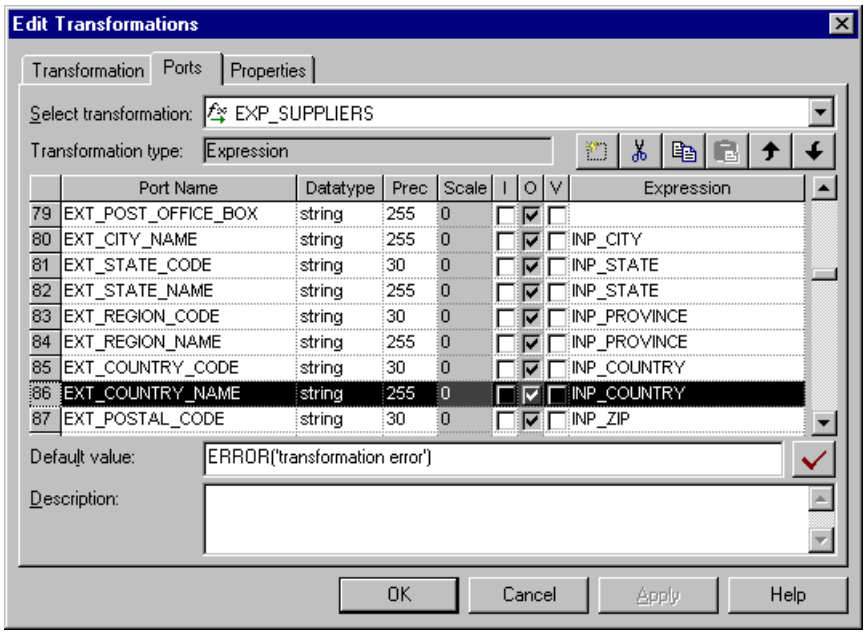
This task is a step in the Process of Configuring the Inventory Module for Oracle 11i on page 87.

The Make-Buy indicator specifies whether a material that was used to manufacture a product was made in-house or bought from an outside vendor. By default, the indicator is set using the INP_PLANNING_MAKE_BUY_CODE. If the code is set to 1, then the indicator is set to M (for make). However, if the code is set to 2, then the indicator is set to B (for buy). Otherwise, the indicator is set to null.

Your organization may require different indicator codes. If so, you can modify the indicator logic by reconfiguring the condition in the mapplet MPLT_SAO_PRODUCTS. For example, you may want your indicator code to be 0 for make, and 1 for buy.

To configure the Make-Buy Indicator

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_PRODUCTS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations dialog box, and click the Port tab to display the EXT_MAKE_BUY_IND port, as shown in the following figure.



4. Edit the condition by replacing the prepackaged condition with your desired logic.
5. Click Apply.
6. Validate the mapplet, and save your changes to the repository.

8 Configuring the Payables Module

After the Payables module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Payables

The Payables module provides information about your enterprise's accounts payable information and identifies the cash requirements to meet your obligations. There are three functional areas:

- Payables Due
- Payables Turnover
- Payments

The information found in the Payables module pertains to data found exclusively under Accounts Payable in your financial statements and chart of accounts. Analysis of your payables allows you to evaluate the efficiency of your cash outflows. The need for analysis has become increasingly important because suppliers have become strategic business partners with the focus on increased efficiency for just in time, and quality purchasing relationships. For more information about the Payables module, see the *Siebel Analytics Enterprise Applications User Guide*.

The default configuration for the Payables module is based on what has been identified as the most-common level of detail, or granularity. However, you can configure or modify the extracts to best meet your business requirements. This chapter describes the configuration information for Oracle11i. For more Information about the extract, transform and load process for Oracle 11i that applies to all of the Financial Analytics related modules (General Ledger, Receivables, Payables, and Profitability), see Configuring the General Ledger Module on page 61.

Configuring Payables Module for Oracle 11i

This section contains Payables configuration information that is specific to Oracle 11i.

Configuring the Accounts Payable Balance ID

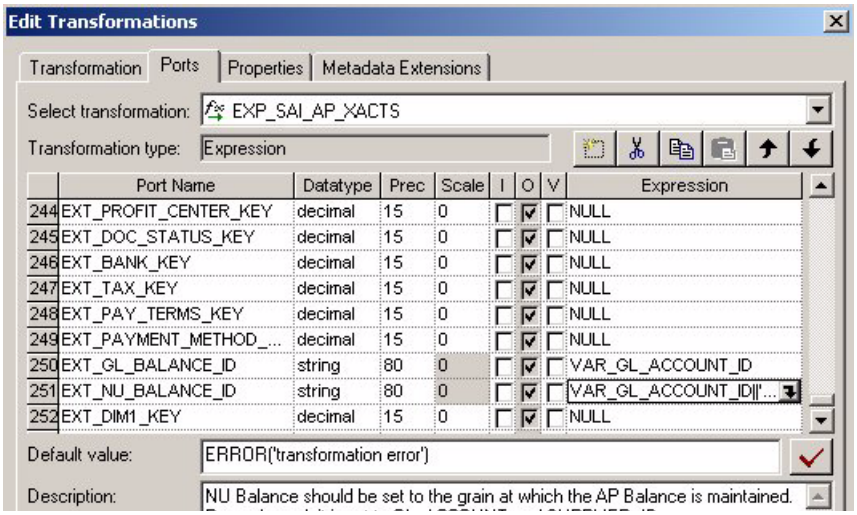
By default, the AP Balance ID is maintained at the following granularity:

```
GL_ACCOUNT_ID||'~'||VENDOR_SITE_ID||'~'||ORGANIZATION_ID
```

However, if you want to maintain your AP balance at a different grain, you can redefine the Balance ID value in the applicable mapplets. You have to modify both the regular mapplet (MPLT_SAI_AP_XACTS) as well as the update mapplet (MPLT_SAI_AP_XACTS_UPDATE) for Oracle 11i.

To modify the Accounts Payable Balance ID

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_SAI_AP_XACTS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations window, and click the Ports tab to edit the Balance ID definition in the EXT_NU_BALANCE_ID column, as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_SAI_AP_XACTS_UPDATE mapplet.

Configuring Payables Module for SAP R/3

This section contains Payables configuration information that is specific to SAP R/3.

Configuring the Balance Extract for Accounts Payable

The Balance and Key IDs set the grain at which you want to maintain a balance. There are three different IDs—General Ledger Balance ID, Accounts Payable Key ID, and Accounts Receivable Key ID. You set the default configurations to the most-representative grains for maintaining the three different balances.

By default, the Accounts Payable Key ID (KEY_ID) is configured as follows:

```
Vendor__creditor__account_number||'~'||Company_code||'~'||Client
```

Therefore, you use the vendor or creditor account number, company code, and client code to maintain the Accounts Payable Balance. You use the client code to distinguish between different instances of SAP R/3. For example, if you are running one instance of SAP R/3 for your U.S. business, and another instance of SAP R/3 in your Japan business, you may have the same AP account numbers in each system, which refer to different accounts. To distinguish the same GL account numbers in different instances, you set the grain of the balance to include the client code.

To configure the Key ID for the AP Balance Extract

1. In PowerCenter Designer, open the Configuration for SAP R/3 folder.
2. In Mapping Designer, open M_S_AP_BALANCE_EXTRACT.
3. Double-click the Expression transformation to open the Edit Transformations window.
4. Edit the KEY_ID. Make sure the AP Key ID in the extract mapping has the same precision as the AP Key ID in the fact load mapping.
5. Validate and save your changes to the repository.

9 Configuring the Receivables Module

After the Receivables module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional, configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Receivables

The Receivables module provides information to support your credit and collection activities, and to monitor and identify potential, receivables problems. There are the three functional areas:

- Credit
- Payments
- Turnover

The information found in the Receivables module pertains to data found exclusively in the Accounts Receivable account grouping of your financial statements and chart of accounts. Each day that your receivables are past the due date represents a significant, opportunity-cost to your company. Keeping a close eye on the trends, and clearing of AR is one way to assess the efficiency of your sales operations, the quality of your receivables, and the value of key customers. For more information about the Receivables module, see the *Siebel Analytics Enterprise Applications User Guide*.

The default configuration for the Receivables module is based on what has been identified as the most-common level of detail or granularity. However, you can configure and modify the extracts to best meet your business requirements. This chapter addresses the configuration information for Oracle 11i. For more Information about the extract transform and load process for Oracle 11i that applies to all of the Financial Analytics related modules (General Ledger, Receivables, Payables, and Profitability), see Configuring the General Ledger Module on page 61.

NOTE: Because of the overlap of information between the Receivables and Profitability module, the configuration information in this chapter is the same as in the chapter for configuring the Profitability module. If you have purchased the Profitability module and have already configured it, no additional configuration is required for the Receivables module.

Process of Configuring Receivables for Oracle 11i

This section contains Receivables configuration information that is specific to Oracle11i.

To configure Receivables for Oracle 11i, perform the following tasks:

- Configuring the AR Balance ID for Oracle 11i on page 100
- Configuring the AR Adjustments Extract on page 101
- Configuring the AR Schedules Extract on page 102
- Configuring the AR Cash Receipt Application Extract on page 103
- Configuring the AR Credit Memo Application Extract on page 104

Configuring the AR Balance ID for Oracle 11i

This task is a step in the Process of Configuring Receivables for Oracle 11i on page 100.

By default, the AR Balance ID is maintained at the following granularity:

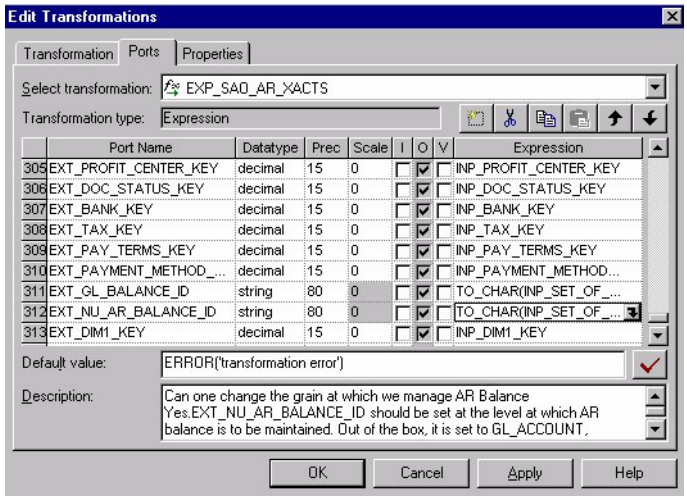
```
SET_OF_BOOKS_ID||'~'||CODE_COMBINATION_ID||'~'||CUSTOMER_ID||'~'||CUSTOMER_SITE_US
E_ID
```

However, if you want to maintain your AR balance at a different grain, you can redefine the Balance ID value in the applicable mapplets. You have to modify both the regular mapplet (MPLT_SAI_AR_XACTS) as well as the update mapplet (MPLT_SAI_AR_XACTS_UPDATE) for Oracle 11i.

To modify the AR Balance ID

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_SAI_AR_XACTS mapplet.

3 Double-click the Expression transformation to open the Edit Transformations window, and click the Ports tab to edit the Balance ID definition in the EXT_NU_AR_BALANCE_ID column, as shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_SAI_AR_XACTS_UPDATE mapplet.

Configuring the AR Adjustments Extract

This task is a step in the Process of Configuring Receivables for Oracle 11i on page 100.

By default, the Receivables module extracts only approved, adjustment entries against accounts receivable transactions. *Approved adjustments* are entries where the AR_ADJUSTMENTS_ALL.STATUS ='A'. If you want to extract additional types of AR adjustment entries, you can remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as the following:

- Those that require more research
- Those that are rejected
- Those that are not accrued charges

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_ADJ), as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_ADJ_PRIMARY).

To modify the extract filter for AR adjustments

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

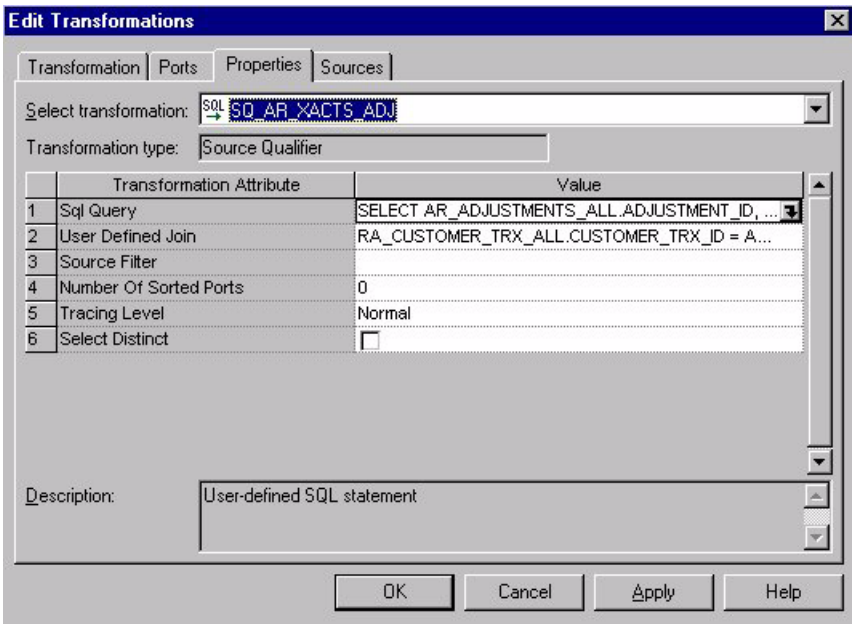
2 In Mapplet Designer, open the MPLT_BCI_AR_XACTS_ADJ mapplet.

3 Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

In the SQL Query field and in the User Defined Join field, modify the condition:

```
AND AR_ADJUSTMENTS_ALL.STATUS = 'A'
```

as shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_ADJ_PRIMARY mapplet.

Configuring the AR Schedules Extract

This task is a step in the Process of Configuring Receivables for Oracle 11i on page 100.

By default, Receivables extracts only *completed schedules*; that is, transactions where the RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG(+) = 'Y'. If you want to extract additional types of AR schedule entries, you must remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as those that were marked as incomplete.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_SCH), as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_SCH_PRIMARY).

To modify the extract filter for AR schedules

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

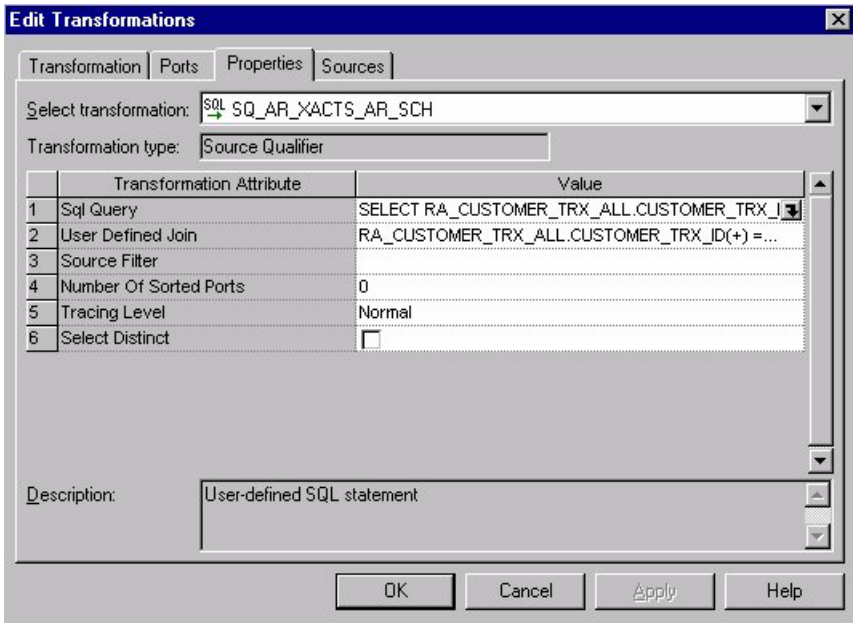
2 In Mapplet Designer, open the MPLT_BCI_AR_XACTS_SCH mapplet.

3 Double-click the Source Qualifier to open the Edit Transformations window.

In the User Defined Join field and in the SQL Query field, modify the condition:

```
AND RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG(+) = 'Y'
```

as shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_SCH_PRIMARY mapplet.

Configuring the AR Cash Receipt Application Extract

This task is a step in the Process of Configuring Receivables for Oracle 11i on page 100.

By default, the Receivables module extracts only confirmed, cash-receipt application entries against accounts receivable transactions. *Confirmed receipts* are entries where the AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG = 'Y' OR 'NULL'. If you want to extract additional types of cash-receipt application entries, you can remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as nonconfirmed applications.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_APPREC) as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_APPREC_PRIMARY).

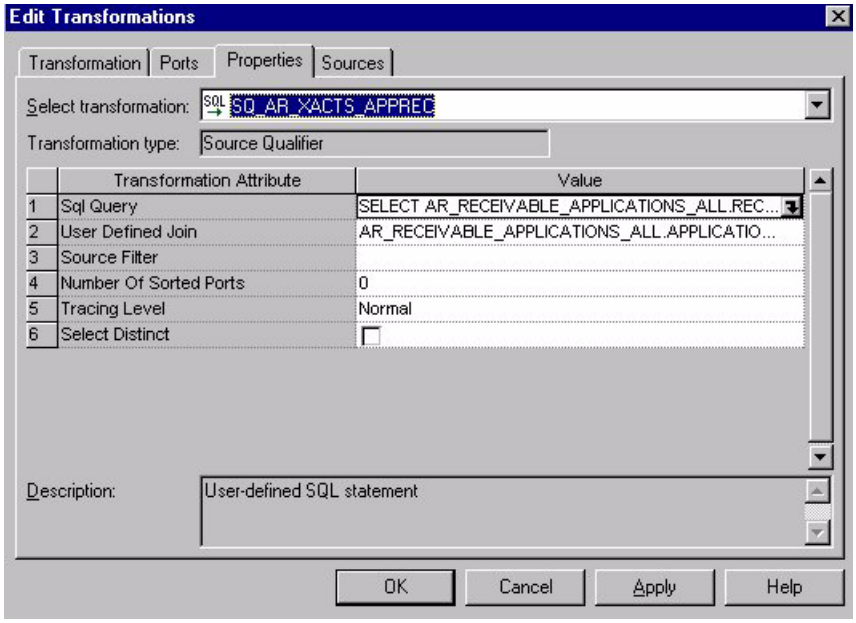
To modify the extract filter for AR cash receipt application

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_BCI_AR_XACTS_APPREC mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

   In the User Defined Join field and in the SQL Query field, modify the condition:

   ```
   AND NVL(AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG,'Y') = 'Y'
   ```

   as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_APPREC_PRIMARY mapplet.

Configuring the AR Credit Memo Application Extract

This task is a step in the Process of Configuring Receivables for Oracle 11i on page 100.

By default, the Receivables module extracts only confirmed, credit-memo application entries against accounts receivable transactions. *Confirmed credit memos* are entries where the AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG = 'Y' OR 'NULL'. If you want to extract additional types of AR credit memo application entries, you can remove the filter. By modifying or removing the filter, you can extract other entries such as nonconfirmed credit memos.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_APPCM) as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_APPCM_PRIMARY).

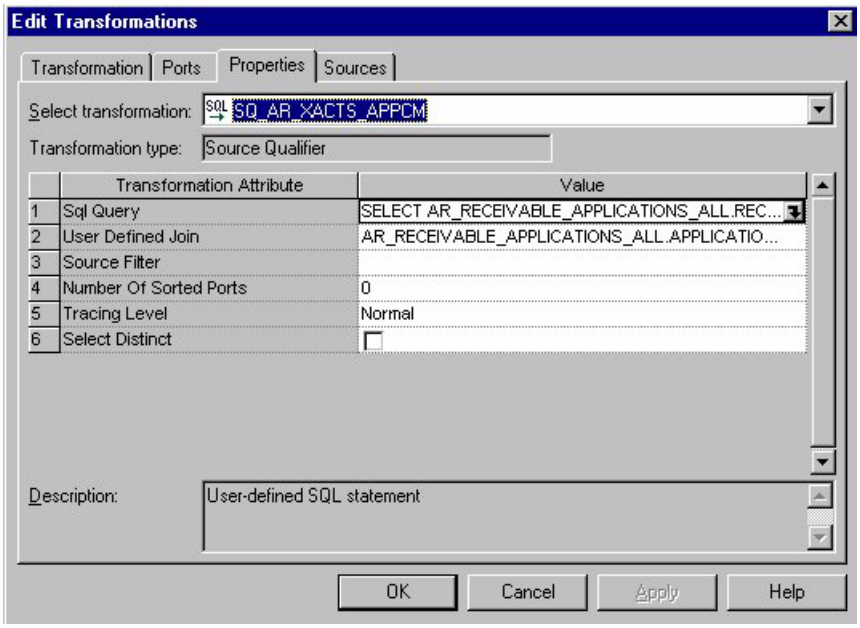
To modify the extract filter for AR Credit Memo Application receipts

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_BCI_AR_XACTS_APPCM mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

   In the User Defined Join field and in the SQL Query field, modify the condition:

   ```
   AND NVL(AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG,'Y') = 'Y'
   ```

   as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_APPCM_PRIMARY mapplet.

Configuring Receivables for SAP R/3

This section contains Receivables configuration information that is specific to SAP R/3.

Configuring the Receivables Extracts

By default, the Accounts Receivable Key ID (KEY_ID) is configured as follows:

```
Customer_number||'~'||Company_code||'~'||Client
```

Therefore, the Accounts Receivable balance is maintained by customer number, company code, and client code. The client code is used to distinguish between different instances of SAP R/3. For example, if you are running one instance of SAP R/3 for your U.S. business, and another instance of SAP R/3 in your Japan business, you may have the same AR account numbers that refer to different accounts in each system. To distinguish the same GL account numbers in different instances the grain of the balance is set to include the client code.

To change the grain at which you accumulate the balances, modify the Key ID definition in the Expression transformation in the applicable mapping. Note that there are two sets of mappings—the balance extract mappings and the initial fact load mappings. The extract provides the balance from the source to staging tables. The initial fact load mappings move the data from staging tables to the data warehouse.

To configure the Key ID for the AR Balance extract

1. In PowerCenter Designer, open the Configuration for SAP R/3 folder.
2. In Mapplet Designer, open M_S_AR_BALANCE_EXTRACT.
3. Double-click the Expression transformation to open the Edit Transformations window.
4. Edit the Key ID.

   Make sure the AR Key ID in the extract mapping has the same precision as the AR Balance ID in the fact load mapping.
5. Validate and save your changes to the repository.

10 Configuring the Profitability Module

After the Profitability module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Profitability

The Profitability module provides cost analysis, revenue trends, and sales performance to provide an accurate picture of profit and loss. There are the two functional areas:

- Profitability
- Revenue

The information found in the Profitability module pertains to data found in the accounts receivable and revenue account groupings of your financial statements and chart of accounts. This module is designed to provide insight into your enterprise's revenue and profitability information, which ties into your accounts receivable. For more information about the Profitability module, see the *Siebel Analytics Enterprise Applications User Guide*.

The default configuration for the Profitability module is based on what has been identified as the most-common level of detail, or granularity. However, the extracts are configurable and can be modified to best meet your business requirements. This chapter describes the configuration information for Oracle 11i sources. For information about the extract transform and load process for Oracle 11i that applies to all modules related to Financial Analytics (General Ledger, Receivables, Payables, and Profitability), see Configuring the General Ledger Module on page 61.

NOTE: Because of the overlap of information between the Receivables and Profitability module, the configuration information in this chapter is the same as in the chapter for configuring the Receivables module. If you have purchased the Receivables module and have already configured it, no additional configuration is required for the Profitability module.

Process of Configuring Profitability for Oracle 11i

This section contains Profitability configuration information that is specific to Oracle 11i.

To configure Profitability for Oracle 11i, perform the following tasks:

- Configuring the Accounts Receivable Balance ID for Oracle 11i on page 108
- Configuring the Accounts Receivable Adjustments Extract on page 109
- Configuring the Accounts Receivable Schedules Extract on page 110
- Configuring the Accounts Receivable Cash-Receipt Application Extract on page 111
- Configuring the Accounts Receivable Credit-Memo Application Extract on page 112

Configuring the Accounts Receivable Balance ID for Oracle 11i

This task is a step in the Process of Configuring Profitability for Oracle 11i on page 108.

By default, the AR Balance ID is maintained at the following granularity:

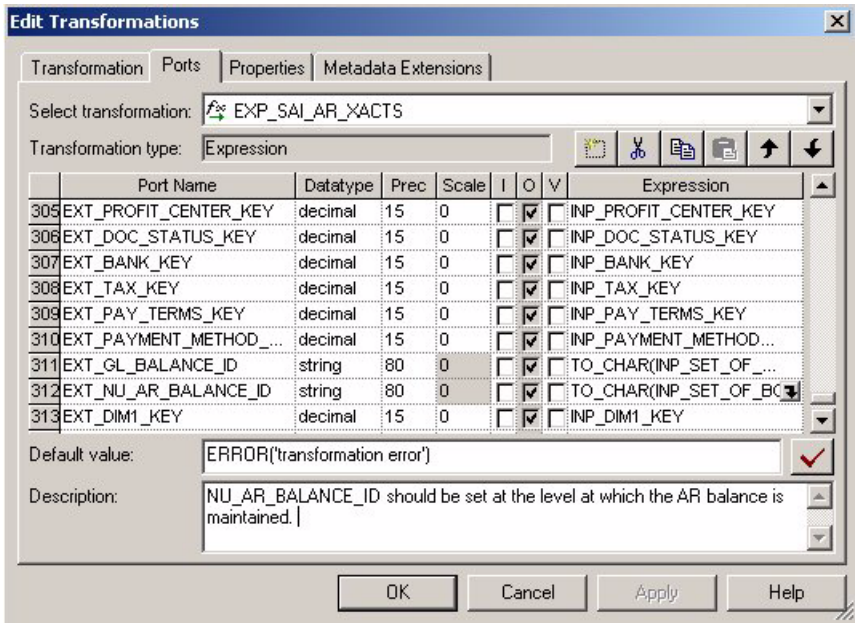
```
SET_OF_BOOKS_ID||'~'||CODE_COMBINATION_ID||'~'||CUSTOMER_ID||'~'||CUSTOMER_SITE_US
E_ID
```

However, if you want to maintain your AR balance at a different grain, you can redefine the Balance ID value in the applicable mapplets. You have to modify both the regular mapplet (MPLT_SAI_AR_XACTS) as well as the update mapplet (MPLT_SAI_AR_XACTS_UPDATE) for Oracle 11i.

To modify the Accounts Receivable Balance ID

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_SAI_AR_XACTS mapplet.

3 Double-click the Expression transformation to open the Edit Transformations window, and click the Ports tab to edit the Balance ID definition in the EXT_NU_AR_BALANCE_ID column, as shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_SAI_AR_XACTS_UPDATE mapplet.

Configuring the Accounts Receivable Adjustments Extract

This task is a step in the Process of Configuring Profitability for Oracle 11i on page 108.

By default, the Receivables module extracts only approved, adjustment entries against accounts receivable transactions. *Approved adjustments* are entries where the AR_ADJUSTMENTS_ALL.STATUS = 'A'. If you want to extract additional types of AR adjustment entries, you can remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as those that require more research, those that are rejected, and those that are not accrued charges.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_ADJ) as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_ADJ_PRIMARY). Repeat the following procedure for each mapplet.

To modify the extract filter for Accounts Receivable adjustments

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

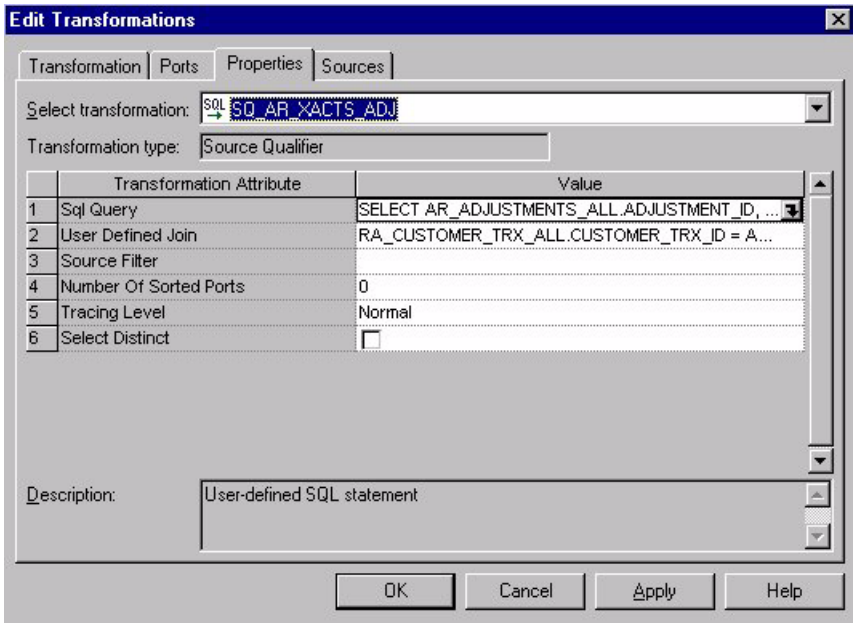
2 In Mapplet Designer, open the MPLT_BCI_AR_XACTS_ADJ mapplet.

3 Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

In the SQL Query field and in the User Defined Join field, modify the condition:

```
AND AR_ADJUSTMENTS_ALL.STATUS = 'A'
```

as shown in the following figure.



4 Validate and save your changes to the repository.

5 Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_ADJ_PRIMARY mapplet.

Configuring the Accounts Receivable Schedules Extract

This task is a step in the Process of Configuring Profitability for Oracle 11i on page 108.

By default, Profitability extracts only completed schedules; that is, transactions where the RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG(+) = 'Y'. If you want to extract additional types of AR schedule entries, you must remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as those that were marked as incomplete.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_SCH) as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_SCH_PRIMARY). Repeat the following procedure for each mapplet.

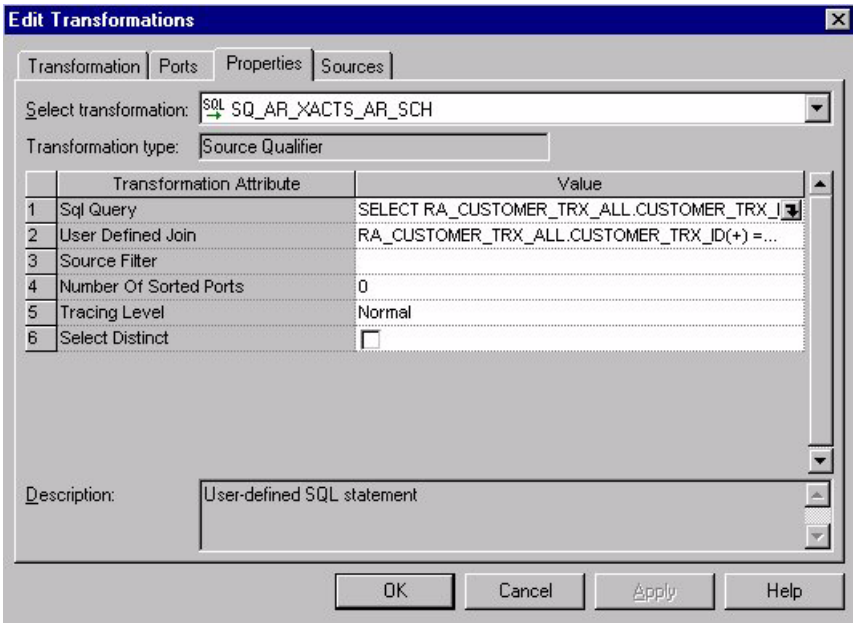
To modify the extract filter for Accounts Receivable schedules

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_BCI_AR_XACTS_SCH mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations window.

   In the User Defined Join field and in the SQL Query field, modify the condition:

   AND RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG(+) = 'Y'

   as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_SCH_PRIMARY mapplet.

Configuring the Accounts Receivable Cash-Receipt Application Extract

This task is a step in the Process of Configuring Profitability for Oracle 11i on page 108.

By default, the Receivables module extracts only confirmed, cash-receipt application entries against accounts receivable transactions. *Confirmed receipts* are entries where the AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG = 'Y' OR 'NULL'. If you want to extract additional types of cash-receipt application entries, you can remove the filter in the Business Component mapplet. By modifying or removing the filter, you can extract other entries, such as nonconfirmed applications.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_APPREC), as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_APPREC_PRIMARY). Repeat the following procedure for each mapplet.

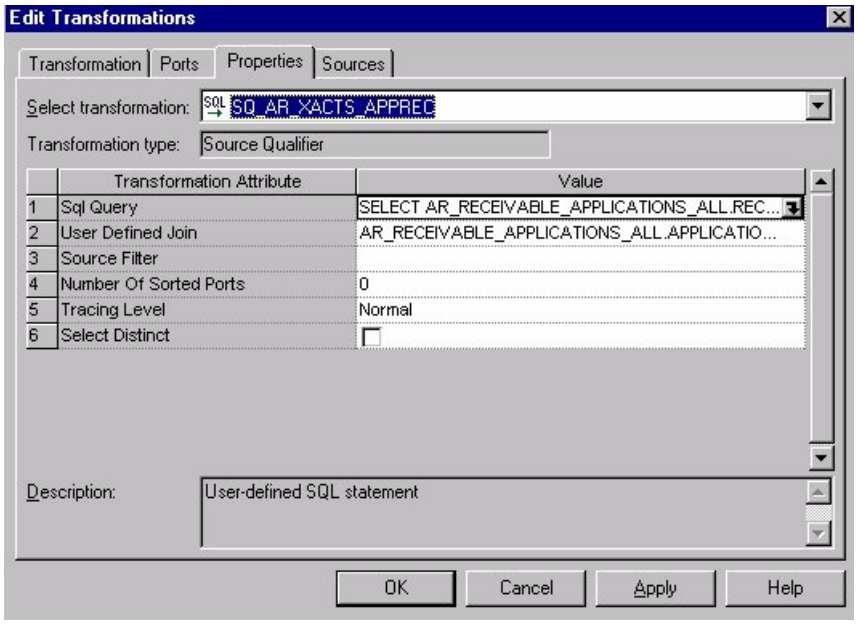
To modify the extract filter for Accounts Receivable cash-receipt application

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_BCI_AR_XACTS_APPREC mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

   In the User Defined Join field and in the SQL Query field, modify the condition:

   ```
   AND NVL(AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG,'Y') = 'Y'
   ```

   as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_APPREC_PRIMARY mapplet.

Configuring the Accounts Receivable Credit-Memo Application Extract

This task is a step in the Process of Configuring Profitability for Oracle 11i on page 108.

By default, the Receivables module extracts only confirmed, credit-memo application entries against accounts receivable transactions. *Confirmed credit memos* are entries where the AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG = 'Y' OR 'NULL'. If you want to extract additional types of AR credit-memo application entries, you can remove the filter. By modifying or removing the filter, you can extract other entries such as nonconfirmed, credit memos.

You must modify both the regular mapplet (MPLT_BCI_AR_XACTS_APPCM), as well as the primary extract mapplet (MPLT_BCI_AR_XACTS_APPCM_PRIMARY). Repeat the following procedure for each mapplet.

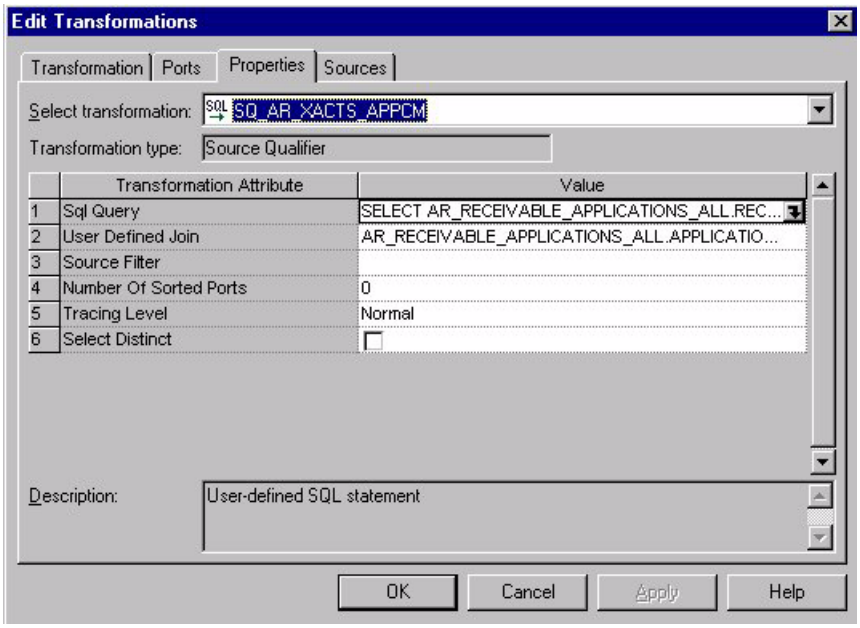
To modify the extract filter for Accounts Receivable Credit-Memo Application receipts

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In Mapplet Designer, open the MPLT_BCI_AR_XACTS_APPCM mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations window, and click the Properties tab.

   In the User Defined Join field and in the SQL Query field, modify the condition:

   ```
   AND NVL(AR_RECEIVABLE_APPLICATIONS_ALL.CONFIRMED_FLAG,'Y') = 'Y'
   ```

   as shown in the following figure.



4. Validate and save your changes to the repository.
5. Repeat Step 2 to Step 4 for the MPLT_BCI_AR_XACTS_APPCM_PRIMARY mapplet.

Configuring Profitability for SAP R/3

This section contains Profitability configuration information that is specific to SAP R/3.

Configuring the Receivables Extracts

By default, the Accounts Receivable Key ID (KEY_ID) is configured as follows:

```
Customer_number||'~'||Company_code||'~'||Client
```

Therefore, the Accounts Receivable balance is maintained by customer number, company code, and client code. The client code is used to distinguish between different instances of SAP R/3. For example, if you are running one instance of SAP R/3 for your U.S. business, and another instance of SAP R/3 for your Japan business, you may have the same AR account numbers that refer to different accounts in each system. To distinguish the same GL account numbers in different instances, the grain of the balance is set to include the client code.

To change the grain at which you accumulate the balances, modify the Key ID definition in the Expression transformation in the corresponding mapping. Note that there are two sets of mappings—the balance extract mappings and the initial fact load mappings. The extract provides the balance from the source to staging tables, and the initial fact load mappings move the data from staging tables to the data warehouse.

To configure the Key ID for the AR Balance extract

1. In PowerCenter Designer, open the Configuration for SAP R/3 folder.
2. In Mapping Designer, open M_S_AR_BALANCE_EXTRACT.
3. Double-click the Expression transformation to open the Edit Transformations window.
4. Edit the Key ID.

   Make sure the AR Key ID in the extract mapping has the same precision as the AR Balance ID in the fact load mapping.
5. Validate and save your changes to the repository.

11 Configuring the Sales Module

After the Sales module is installed, you may need to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Enterprise Sales Analytics

The Sales module allows you to analyze the movement of sales opportunities through different stages of the sales cycle. This analysis includes insight into which items are booked, backlogged, and invoiced. It also provides you with information that allows you to evaluate the sales performance of individual sales representatives or departments. The Sales module contains the functional area, Orders and Revenue.

The Orders and Revenue functional area consists of orders, invoices, and backlog. Sales orders are the entry point for the sales process. Invoices are the exit point from the fulfillment process. *Backlogs* are points of congestion in your fulfillment process.

In the Sales module, two main types of backlog exist:

- Operational
- Financial

Operational backlog can be further broken down into scheduled, unscheduled, delinquent, and blocked backlogs. Bookings and Revenue can be populated from three different sources:

- Oracle 11i
- Universal Source

Orders and Revenue also requires post-load processing mappings to populate its tables.

For more information on the Sales module or its functional areas, see the *Siebel Analytics Enterprise Applications User Guide*.

Process of Configuring Sales for Oracle 11i

This section contains Sales module configuration points that are specific to Oracle 11i. Of the three functional areas in the Sales module, Orders and Revenue is the only functional area that has prepackaged Oracle 11i business adapters to populate the warehouse tables; therefore, you can not find configuration points for the other two functional areas in this section. To configure Sales for Oracle 11i, perform the following tasks:

- Configuring Sales Order Lines Data Storage on page 116
- Tracking Attribute Changes in Bookings on page 118
- Configuring Sales Schedule Lines Data Storage on page 123
- Configuring Different Types of Backlog Calculations on page 128
- Accounting for Negative Values for Orders, Invoices, and Picks on page 135

Configuring Sales Order Lines Data Storage

This task is a step in the Process of Configuring Sales for Oracle 11i on page 116.

Sales order lines are the itemized lines that make up a sales order. This information is stored in the IA_SALES_ORDLNS table. This topic describes how to modify the type of information stored in this table.

Configuring the Handling of Booked and Nonbooked Orders in the Order Lines and Bookings Table

By default, only booked orders are extracted from the Oracle 11i source system as shown in Figure 6. Therefore, all orders loaded into the Sales Order Lines and Bookings tables are flagged as booked (EXT_BOOKING_FLAG = 'Y').

However, if you want to load nonbooked orders into the Sales Order Lines table, you have to configure the extract so that it does not filter out nonbooked orders. In Oracle 11i, the OE_LINES_ALL.BOOKED_FLAG = Y condition indicates that an order is booked; therefore, this statement is used to filter out nonbooked orders. To load all orders, including nonbooked orders, remove the filter condition from the WHERE clause in the S_M_I_SALES_ORDLNS_EXTRACT and S_M_I_SALES_ORDLNS_PRIMARY_EXTRACT sessions.

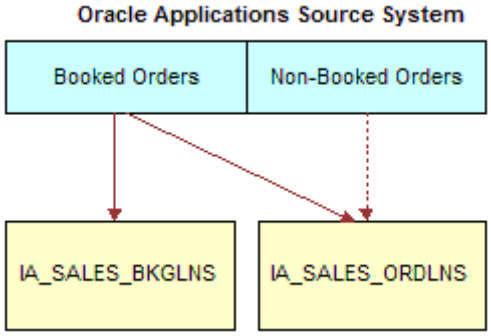


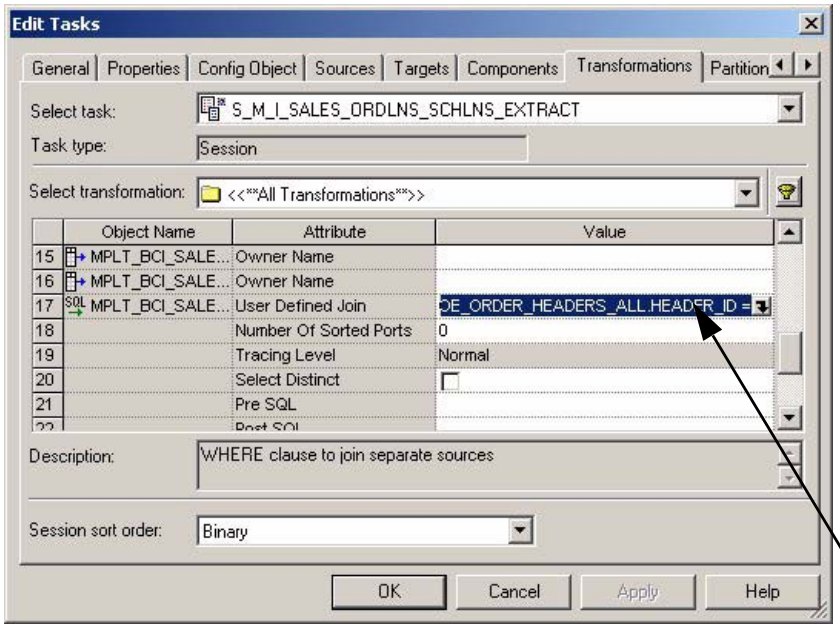
Figure 6. Oracle 11i: Default Configuration for Loading Booked Orders

To include nonbooked orders in the Sales Order Lines table

1. In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i folder.
2. Expand the Oracle 11i Enterprise Sales Application workflow.
3. Expand the O_FACTS_EXTRACT worklet and the O_FACT_EXTRACT_SALES worklet.
4. Double-click the S_M_I_SALES_ORDLNS_SCHLNS_EXTRACT session to open the Edit Tasks box.

5 In the Transformations tab, edit the MPLT_BCI_SALES_ORDLNS SQL statement.

Remove the OE_ORDER_LINES_ALL.BOOKED_FLAG='Y' filter in the SQL statement to include nonbooked orders in the Sales Order lines table.



OE_ORDER_LINES_ALL.BOOKED_FLAG='Y' is the filter condition for extracting only booked orders.

6 Validate and save your changes to the repository.

Tracking Attribute Changes in Bookings

Changes in booked orders are tracked in the Booking Lines table (IA_SALES_BKGLNS), not in the Sales Order Lines table (IA_SALES_ORDLNS). By default, the only changes tracked in the IA_SALES_BKGLNS table are changes in the ordered amount, ordered quantity, or Booking ID. By default, the Booking ID is defined as:

```
TO_CHAR(INP_LINE_ID)||'~'||TO_CHAR(INP_WAREHOUSE_ID).
```

Any changes in the sales order number, sales order item, or client code result in a change in the Booking ID, which results in another row in the IA_SALES_BLGLNS table. However, changes in any other fields does not result in a new row; instead, the existing information are overwritten with the changed information. No history is kept for changes to these other field values. If you want to track other changes you can do so. For example, you may want to track changes to the sales representative who is handling the order. The ETL processes are prepackaged to overwrite sales representative changes; however, if you want to retain them, you must add the attribute to the Booking ID definition in the Booking ID expression in the Source Adapter mapplet (MPLT_SAI_SALES_ORDLNS). The following section describes what happens if you modify the Booking ID to include the sales representative.

Viewing the Data Warehouse Changes by Salesperson ID

Assume you want to track changes to the sales representative for bookings and debookings. You decide to do this to better evaluate each representative's sales performance. To track changes by Salesperson ID, you have to modify the VAR_BOOKING_ID to use the value: TO_CHAR(INP_SALESREP_ID). The following describes what happens in the source system and the IA_SALES_BKGLNS table when you change sales representatives under this scenario.

Day 1: One order is placed with Salesperson 1001. The source system rows the information as shown in the Table 16.

Table 16. Oracle 11i: Source System Table Row After Day One Activity

Sales Order Number	Sales Order Line Number	Salesperson ID	Quantity	Selling Price	Date
1	1	1001	100	25	1-June-2000

The row in Table 16 would be entered into the IA Bookings table (IA_SALES_BKGLNS) as shown in Table 17.

Table 17. Oracle 11i: IA_SALES_BKGLNS Table Row After Day One Activity

SALES_ORDER_NUM	SALES_ORDER_ITEM	SALESREP_ID	SALES_QTY	NET_DOC_AMT	BOOKING_ID	BOOKED_ON_DT
1	1	1001	100	2500	1001	1-June-2000

Day 2: Salesperson 1002 takes over this order, replacing Salesperson 1001. Thus, the salesperson associated with the order is changed from 1001 to 1002 in the source system. The row in the source system would look like the row shown in Table 18.

Table 18. Oracle 11i: Source System Table Row After Day Two Activity

Sales Order Number	Sales Order Line Number	Salesperson ID	Quantity	Selling Price	Date
1	1	*1002*	100	25	2-June-2000

The Sales Order Lines ADI, which also writes to the booking table, now does a debooking for the old line and inserts a new row into the IA_SALES_BKGLNS booking table. On day two, the row in the IA_SALES_BKGLNS table would look like the row shown in the Table 19.

Table 19. Oracle 11i: IA_SALES_BKGLNS Table Row After Day Two Activity

SALES_ORDER_NUM	SALES_ORDER_ITEM	SALESREP_ID	SALES_QTY	NET_DOC_AMT	BOOKING_ID	BOOKED_ON_DT
1	1	1001	100	2500	1001	1-June-2000
1	1	1001	-100	-2500	1001	2-June-2000
1	1	1002	100	2500	1002	2-June-2000

Tracking Multiple Attribute Changes in Bookings

When you modify the default VAR_BOOKING_ID column, the SQL statement is configured as follows for Oracle 11i:

```
TO_CHAR(INP_LINE_ID)||'~'||to_char(INP_WAREHOUSE_ID)
```

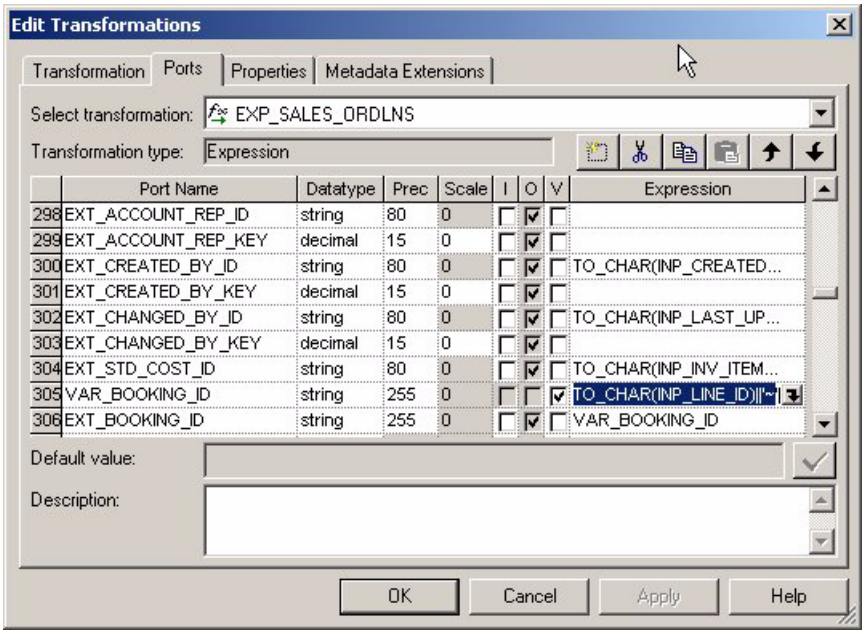
However, if you want to track changes based on more than one attribute, in the SQL statement you must concatenate the attribute column IDs in the VAR_BOOKING_ID column. For example, if you want to track changes in Salespersons and Sold-to-Customer, then concatenate the technical name IDs in the VAR_BOOKING_ID column as follows:

```
TO_CHAR(INP_LINE_ID)||'~'||TO_CHAR(INP_WAREHOUSE_ID)||'~'||TO_CHAR(INP_SALESREP_ID
)||'~'||TO_CHAR(INP_SHIP_TO_SITE_USE_ID)
```

To track dimensional attribute changes in bookings

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SALES_ORDLNS mapplet.

3 Double-click the EXP_SALES_ORDLNS Expression transformation to open the Edit Transformation box, as shown in the following figure.



4 In the Ports tab, edit the expression for the VAR_BOOKING_ID port, and enter the ID of the attribute for which you want to track changes.

If you want to track changes in multiple attributes, concatenate the IDs of all attributes and put the concatenated value in the VAR_BOOKING_ID column.

5 Validate and save your changes to the repository.

Assigning Sales Order Hold Types

IA_SALES_CYCLNS and IA_SALES_CYCHDR cycle time tables are populated by post-load processing to track time durations between different stages of a sales order. For example, the IA_SALES_CYCHDR table can calculate the time it took to ship an entire order. The IA_SALES_CYCLNS table can calculate the time it took to ship each individual sales order line item within the order. You can analyze time components at a variety of granular levels.

For example, if an order was shipped late, you can track all of the individual stages that led to the late shipment. One possible area of interest is orders placed on hold. How many times was the order placed on hold? How long was the order on hold in each phase? For what reason was the order placed on hold? In order to track hold information, you must configure the Sales module to accommodate your different hold types.

The Sales module prepackages both cycle time tables with nine columns for nine different types of holds. To populate these columns, you must first understand their data flow. The source system stores various types of holds; the Siebel Analytics Enterprise Data Warehouse supports storage of up to nine different types. To do so, hold information is extracted from the source system and populate those values in the IA_HOLD_TYPE column in the OD_SALES_ORDHLD table. These source values are then mapped to a set of domain values, which are used by both cycle time tables as shown in Figure 7. This configuration procedure requires that you map your source values to the set of domain values so that the values can be translated.

By default, the nine domain value hold types are 'Hold 1', 'Hold 2', 'Hold 3', 'Hold 4', 'Hold 5', 'Hold 6', 'Hold 7', 'Hold 8', and 'Hold 9', as shown in Figure 7. For more information on domain values, see *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

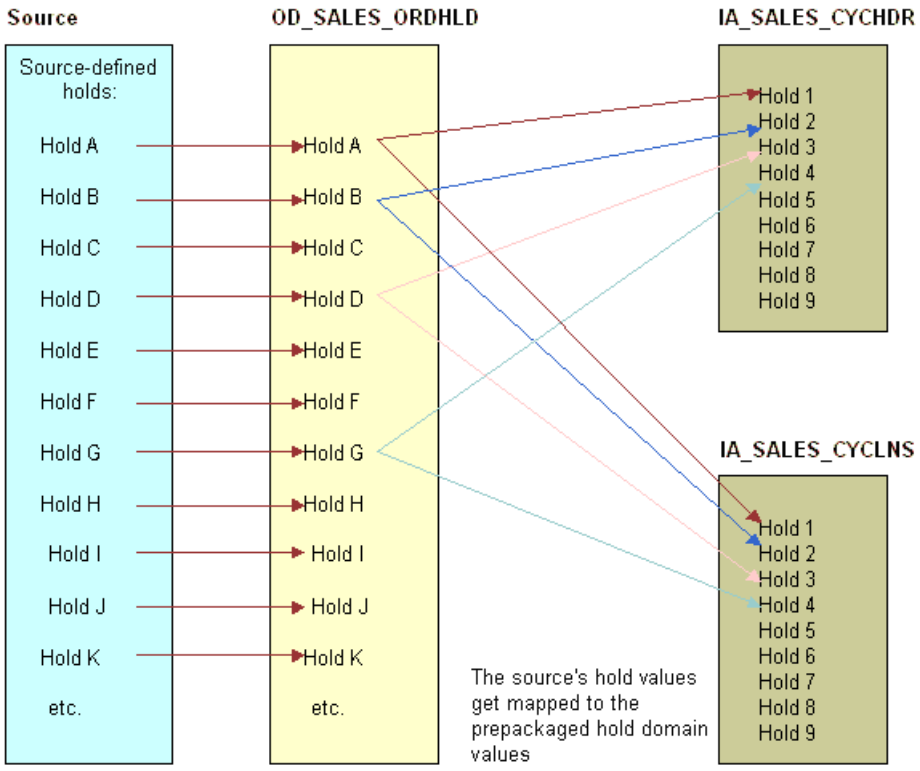
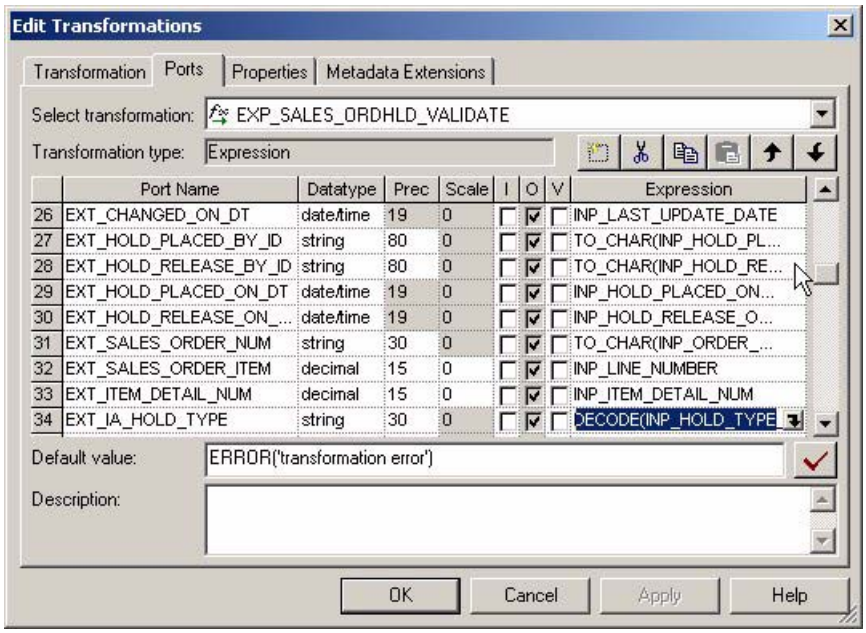


Figure 7. Oracle 11i: Map Source's Hold Types to IA Hold Type Domain Values

To map hold types to the domain value hold columns

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SALES_ORDHLD mapplet.

3 Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4 In the Ports table, edit the expression for the EXT_IA_HOLD_TYPE port.

By default, the expression is as follows:

```
DECODE(INP_HOLD_TYPE_CODE,'CONFIGURATOR','HOLD1','CREDIT','HOLD2','GSA','HOLD3','HOLD','HOLD4','HOLD5')
```

5 Validate and save your changes to the repository.

Configuring Sales Schedule Lines Data Storage

This task is a step in the Process of Configuring Sales for Oracle 11i on page 116.

Sales schedule lines detail when each order's items are slated for shipment. Each sales order can be broken into sales order lines, and each sales order line can have multiple schedule lines.

For example, you might not have enough stock to fulfill a particular sales order line, therefore you create two schedules to fulfill it. One schedule ships what you currently have in stock, and the other schedule includes enough time for you to manufacture and ship the remaining items of the sales order line. This information is stored in the IA_SALES_SCHLNS table. This topic describes how to modify the type of information stored in this table.

Loading Bookings at the Schedule Line Level

As initially configured for Oracle 11i, bookings are recorded at the Sales Order Line level. For each booked order, there is at least one row in the Bookings table, as shown in Figure 8. By default, the EXT_BOOKING_FLAG is set to 'N' in the Schedule Lines MPLT_SAI_SALES_SCHLNS Source Adapter mapplet.

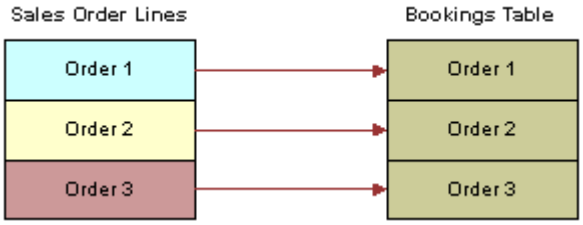


Figure 8. Oracle 11i: Bookings at the Order Line Level

Bookings may be recorded at the Sales Schedule Line level instead of the Sales Order Line level. At the Sales Order Line level, bookings provide a more granular view, as the orders are segmented by schedule line. Bookings recorded at the Schedule Line level provide one row in the Bookings table for each schedule line, as shown in Figure 9. The Booking flag (EXT_BOOKING_FLAG) is set to 'Y' in the Schedule Lines Source Adapter mapplet.

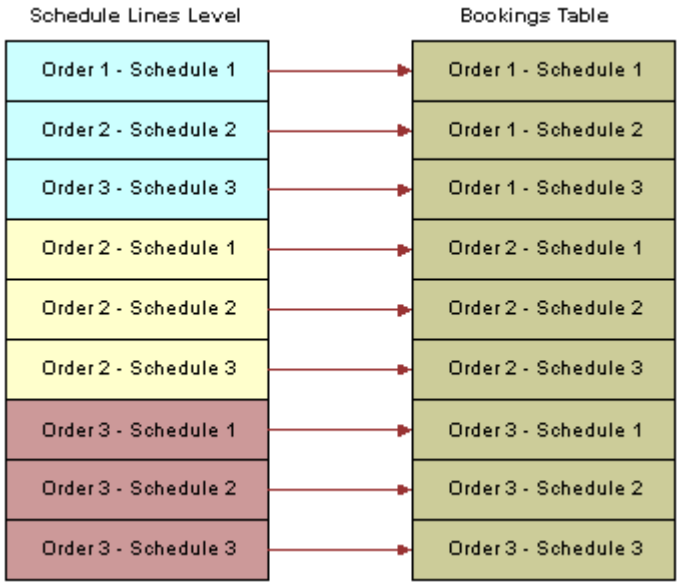


Figure 9. Oracle 11i: Bookings at the Schedule Line Level

Booking Logic

There are booking flags in both the Sales Order Lines Source Adapter mapplet and in the Sales Schedule Lines Source Adapter mapplet. Figure 10 describes how the analytic system reacts based on the booking flag settings. Usually, one of the flags is set to 'Y' and the other to 'N', resulting in the booking logic preferred. However, if both mapplets' flags are set to 'Y', the system records bookings at the default Order Lines level. If both mapplets are set to 'N', no booking logic is enforced.

Sales Order Lines Source Adapter: Booking Flag =	Sales Schedule Lines Source Adapter: Booking Flag =	Result:
'Y'	'N'	Bookings are done at the order line level
'Y'	'Y'	Bookings are done at the order line level
'N'	'Y'	Bookings are done at the schedule line level
'N'	'N'	No booking logic

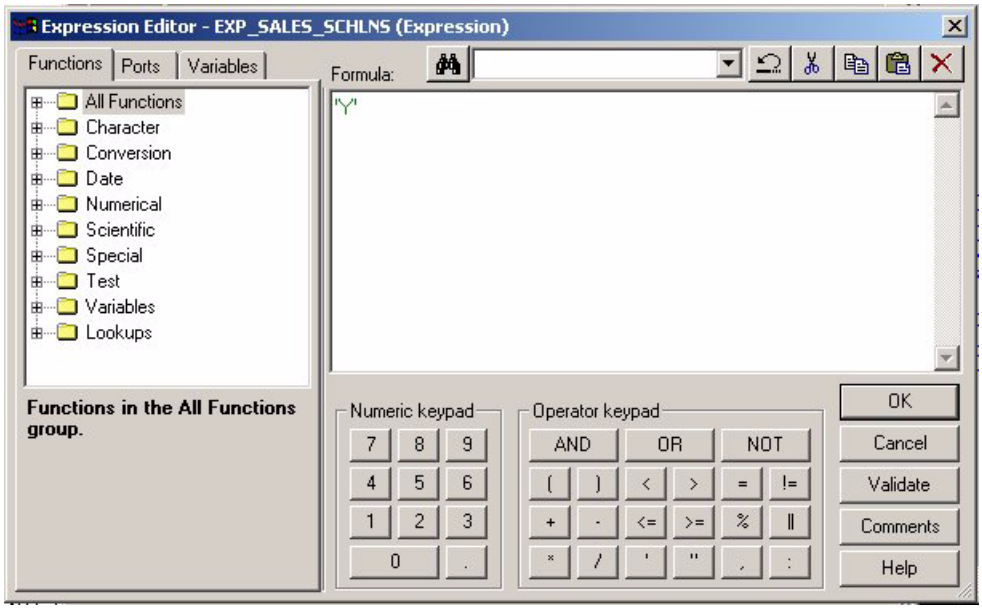
Figure 10. Oracle 11i: Bookings Logic

To configure the system to do bookings at the sales schedule lines level

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SALES_SCHLNS mapplet.
3. Double-click the EXP_SALES_SCHLNS Expression transformation to open the Edit Transformation box.

4 In the Ports tab, edit the expression for EXT_BOOKING_FLAG to row bookings at the Schedule Line level.

Modify the EXT_BOOKING_FLAG definition by inserting 'Y', as shown in the following figure.



5 In addition, change the EXT_BOOKING_FLAG in the Sales Order Lines Source Adapter to 'N' or the system records bookings at the Order Lines level by default.

6 Validate and save your changes to the repository.

Defining Early and Late Tolerances for Shipments

You can configure the definition of early and late shipments by editing the EXP_SALES_SCHLNS_PICKQTY column in the M_I_SALES_SCHLNS_PICKQTY_LOAD mapping in Oracle 11i. The M_I_SALES_SCHLNS_PICKQTY_LOAD mapping compares each of the completed pick lines against their corresponding schedule lines, and then updates the Schedule Lines table with the totals for the pick lines. This comparison allows easy querying against the Schedule Lines table to determine which schedule lines have been shipped on time, early, or late. The logic is prepackaged to flag orders scheduled to ship on a different day than their pick date as either early or late.

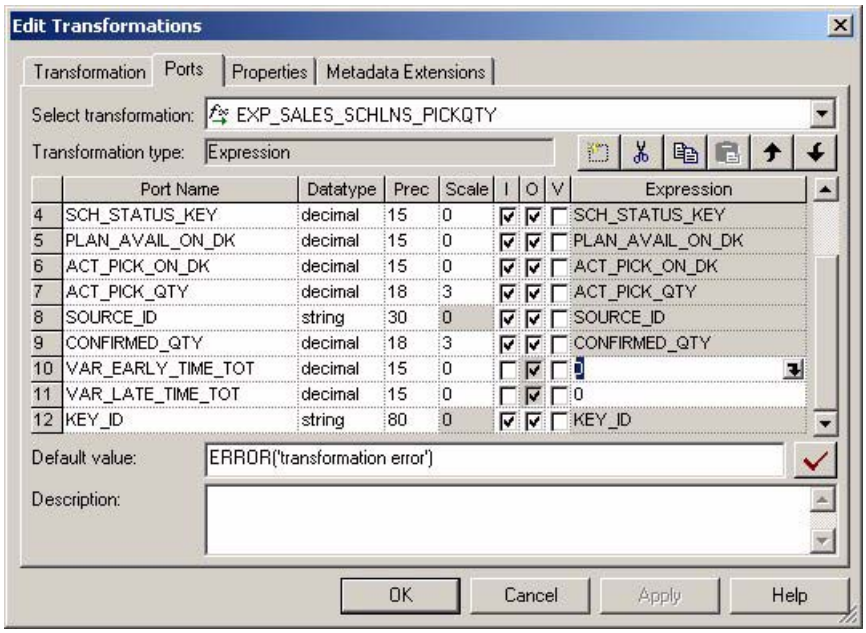
However, if you want to redefine the number of days before a pick is considered early or late, you can configure the EXP_SALES_SCHLNS_PICKQTY Expression transformation.

To configure early and late tolerances for shipments

In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

1 Open the M_I_SALES_SCHLNS_PICKQTY_LOAD mapping.

2 Double-click the EXP_SALES_SCHLNS_PICKQTY Expression transformation to open the Edit Transformation box, as shown in the following figure.



3 In the Ports tab, select the expression for the port to modify and display the SQL statement.

For example, if you want to allow two days after the scheduled pick date before you flag the pick as late, edit the VAR_LATE_TIME_TOT by entering '2'.

- To set the number of days before a pick is flagged as early, edit the SQL for the VAR_EARLY_TIME_TOT port.
- To set the number of days before a pick is flagged as late, edit the SQL for the VAR_LATE_TIME_TOT port.

4 Validate and save your changes to the repository.

Configuring Sales Invoice Lines Data Storage

Sales invoice lines are payments for items ordered by a customer. This information is stored in the IA_SALES_IVCLNS table. This topic describes how to modify the type of information stored in this table.

Configuring the Sales Invoice Extract

By default, the Sales module is configured to extract completed sales invoices when performing the Sales Invoice data extract. Oracle 11i uses a flag to indicate whether a sales invoice is complete. In particular, completed sales invoices are those where the RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG ='Y' in Oracle 11i.

To extract incomplete sales invoices, as well as complete invoices, remove the extract filter statement.

To remove the extract filter for sales invoices

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open a Sales Invoice Lines Business Component mapplet. Modify both the regular extract mapplet (MPLT_BCI_SALES_IVCLNS) and the primary extract mapplet (MPLT_BCI_SALES_IVCLNS_PRIMARY).
3. Open the Source Qualifier to edit the SQL statement in the SQL Query and User Defined Join fields.
4. In the User Defined Join field and in the SQL Query field, remove the condition:

   ```
   AND RA_CUSTOMER_TRX_ALL.COMPLETE_FLAG(+) ='Y'
   ```

5. Validate and save your changes to the repository.

Configuring Different Types of Backlog Calculations

This task is a step in the Process of Configuring Sales for Oracle 11i on page 116.

Backlog information is stored in the IA_BLGLNS and IA_SALES_BLGHIS tables. This topic describes how to modify the type of information stored in these tables. Many types of backlog exist in the Sales module: financial backlog, operational backlog, delinquent backlog, scheduled backlog, unscheduled backlog, and blocked backlog. Each type of backlog is defined by two particular dates in the sales process; therefore, calculations of backlog hits multiple fact tables.

For example, financial backlog records which items have been ordered but payment has not been received. Thus, to calculate the number of financial backlog items, you use the Sales Order Lines table (to determine which items have been ordered) and the Sales Invoice Lines table (to see which orders have been paid for). Using these two tables, you can determine the number of items and the value of those items that are on financial backlog.

Adding Closed Orders to Backlog Calculations

By default, the Sales module only extracts open sales orders from the Sales Order Lines table (IA_SALES_ORDLNS) for backlog calculations to populate the Backlog tables (IA_SALES_BLGLNS and IA_SALES_BLGHIS). *Open sales orders* are defined as orders that are not canceled or not complete. The purpose in extracting only open orders is that in most organizations those orders that are closed are no longer a part of backlog. However, if you want to extract sales orders that are marked as closed, you may remove the default filter condition from the extract mapping.

In Oracle 11i, open sales orders are flagged in the source system with one of the following two statuses—S6 and S9. If you want to remove the filter condition, you must remove the condition containing these two source system values. An example is provided in the following procedure.

During backlog calculations, the IA_SALES_ORDLNS, IA_SALES_SCHLNS, and IA_SALES_PCKLNS sales fact tables are joined to the IA_SALES_BLGLNS table to find out which sales order numbers are eligible for backlog processing; only those sales order numbers present in the IA_SALES_BLGLNS are used for backlog calculations.

For example, assume your customer orders ten items. Six items are invoiced and shipped, but four items are placed on operational and financial backlog. This backlog status continues until one of two things happens:

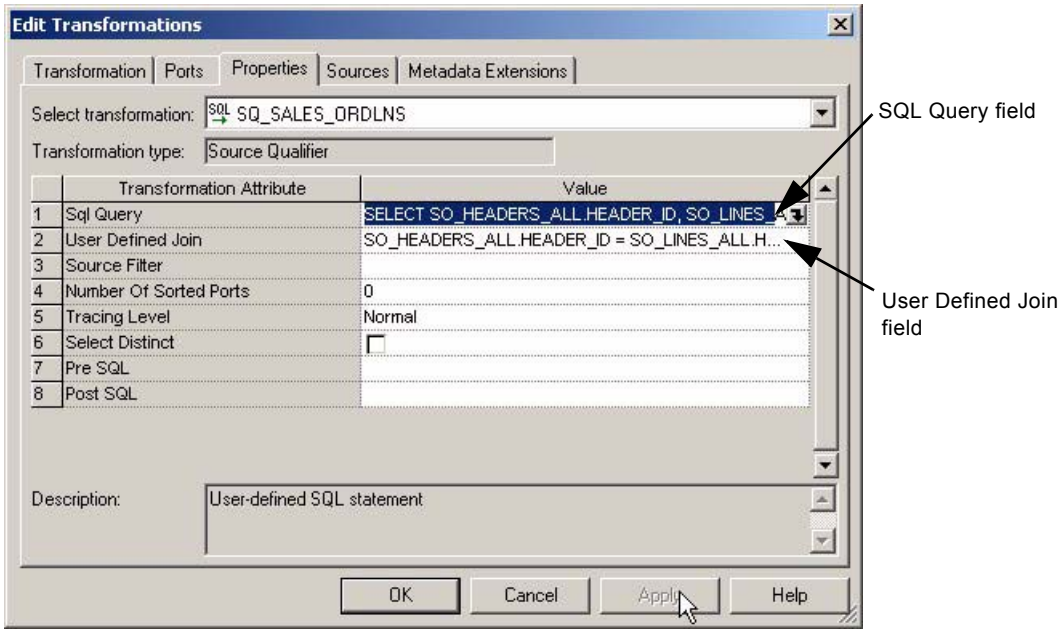
- The items are eventually shipped and invoiced. In this case, the financial and operational backlog statuses are removed and the order status is changed to CLOSED.
- The remainder of the order is canceled. In this case, the financial and operational backlog statuses are removed and the order status is changed to CLOSED.

NOTE: Canceled orders' statuses are change to CLOSED, but the rows are not purged from the table.

If you choose to extract sales orders that are flagged as closed, you must remove the filter condition for the extract. To do so, use the procedure that follows.

To remove open order extract filters

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_BCI_SALES_BLGLNS mapplet.
3. Open the Source Qualifier transformation to edit the SQL statement in the SQL Query field and the User Defined Join field, as shown in the following figure.



4. Edit the expression by modifying the filter.

5 Validate and save your changes to the repository.

Configuring Order Types for Backlog Calculations

The BACKLOG_FLAG in the IA_SALES_ORDLNS table is used to identify which sales orders are eligible for backlog calculations. By default, all sales orders have their backlog flag set to 'Y'. As a result, all sales orders are included in backlog calculations.

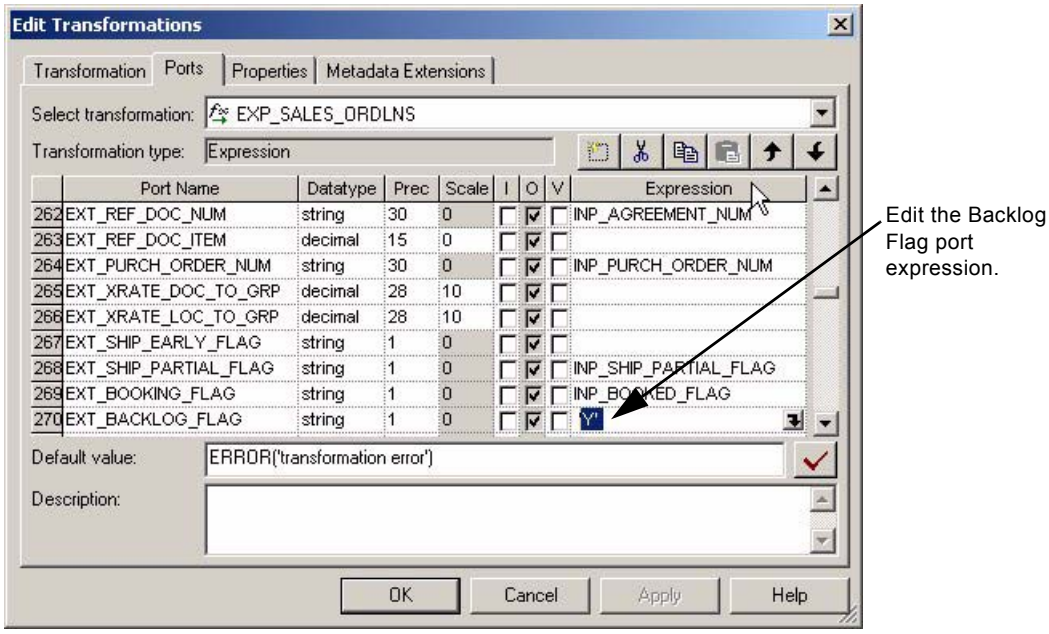
However, if you wish to process only certain types of sales orders, you must insert a conditional statement for the Backlog Flag. Valid values for the Backlog Flag are 'Y' and 'N'.

To edit the Backlog flag

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 Open the MPLT_SAI_SALES_ORDLNS mapplet.

3 Open the Expression transformation EXP_SALES_ORDLNS to edit the Backlog Flag port, as shown in the following figure.



4 Modify the SQL statement in the BACKLOG_FLAG port to include or exclude sales orders from backlog calculations.

5 Validate the expression, and save your changes to the repository.

Configuring the Backlog Period Date

The Backlog table (IA_BLGLNS) stores backlog data for the current month. In contrast, the Backlog History table (IA_BLGHIS) stores snapshots of all previous months' historical backlog data. The periods for which the Backlog History table tracks backlog data is defined by the Backlog Period Date. By default, the date is set as the last calendar day of the month; however this can be configured. You may want to view backlog history at a more detailed level, such as by day or by week, instead of by month. The following example describes how historical backlog data is stored and what the implications are for changing the backlog time period.

Example of How Backlog Data Is Stored in the Backlog History Table

Assume you represent a manufacturing company where financial backlog is defined as any item that is ordered, but not invoiced. On February 1, 2001, you received an order (Sales Order #1) for 30 products. 20 were shipped and invoiced and 10 were shipped, but not invoiced. At the end of the day, there is an entry in the Backlog table and in the Backlog History table. The entry in the Backlog History table looks like that shown in Table 20.

Table 20. Oracle 11i: Backlog History Table Entry as of February 1, 2001

SALES_ORDER_NUM (Sales Order Number)	BACKLOG _DK (Backlog Date)	BACKLOG_PERIOD_DK (Backlog Period Date)	OPEN_QTY (Backlog Quantity)
1	02/01/2001	02/28/2001	10

On February 2, 5 of the 10 financial backlog items are invoiced and, thus, removed from the backlog. Thus, there is an update to the existing row in the Backlog History table, as shown in Table 21.

Table 21. Oracle 11i: Backlog History Table Entry as of February 2, 2001

SALES_ORDER_NUM (Sales Order Number)	BACKLOG _DK (Backlog Date)	BACKLOG_PERIOD_DK (Backlog Period Date)	OPEN_QTY (Backlog Quantity)
1	~~02/01/2001~~ 02/01/2001	02/28/2001	~~40~~ 5

No further activity happens until February 28. On February 28, the remaining 5 items on financial backlog are invoiced and removed from financial backlog. In addition, a new sales order (Sales Order #2) comes in for 50 new items. All of the items are put on financial backlog.

Even though all items from Sales Order #1 are cleared from financial backlog, the last backlog row remains in the Backlog History table. The purpose in retaining the last row is to indicate that there was backlog for this particular order. The quantity, in this case 5 items, does not tell you how many items were initially on backlog, which was 10.

For the 50 new financial backlog items, there is a new entry into the Backlog History table. So, as of February 28, 2001, the Backlog History table looks like the Table 22.

Table 22. Oracle 11i: Backlog History Table Entry as of February 28, 2001

SALES_ORDER_NUM (Sales Order Number)	BACKLOG _DK (Backlog Date)	BACKLOG_PERIOD_DK (Backlog Period Date)	OPEN_QTY (Backlog Quantity)
1	~~02/01/2001~~ 02/02/2001	02/28/2001	~~40~~ 5
2	02/28/2001	02/28/2001	50

On March 1, 30 more items are ordered (Sales Order #3), all of which are on financial backlog. The resulting Backlog History table looks like Table 23.

Table 23. Oracle 11i: Backlog History Table Entry as of March 1, 2001

SALES_ORDER_NUM (Sales Order Number)	BACKLOG _DK (Backlog Date)	BACKLOG_PERIOD_DK (Backlog Period Date)	OPEN_QTY (Backlog Quantity)
1	~~02/01/2001~~ 02/02/2001	02/28/2001	5
2	02/28/2001	02/28/2001	50
2	03/01/2001	03/31/2001	50
3	03/01/2001	03/31/2001	30

Because backlog history is maintained at the monthly level, you have a partial history of your backlogs. Based on the latest state of the Backlog History table shown in Table 23, you can see that sales order number 1 and 2 ended up with 5 and 50 financial backlogged items respectively. You do not have visibility into what the initial financial backlogged item quantities were for both of these sales orders; you only have their ending quantities.

If you decide that you want to track more details on how the items moved out of backlog, then you'll have to maintain the history at a more granular level. For instance, if you want to know the number of items that were on backlog when the it was first opened, you would have to track the backlog history by day, instead of by month.

For example, if you maintained backlog history at the daily level you would be able to capture that sales order 1 had an initial backlog of 10 as of February 1 and the backlog quantity shrank to 5 as of February 2. So, by capturing history at the daily level, you could then compute cycle times on how long it took to move items out of backlog. However, if you decide to capture backlog history at a more detailed level, you may compromise performance because tracking backlog history at the daily level can increase the size of the Backlog History table exponentially.

If you choose to change the time period for which historical backlog data is kept, you must verify that all types of backlog are being stored at the same grain; which requires modification to multiple mappings. Table 24 provides a list of all applicable mappings and their corresponding Expression transformations that you must modify.

Table 24. Oracle 11i: Backlog History Applicable Mappings and Expression Transformations

Mapping	Expression Transformation
M_I_SALES_BLGLNS_LOAD	EXP_SALES_BLGLNS

The backlog history period is monthly by default. The default SQL statement in the Expression transformation of the listed mappings is as follows:

```
'trunc(DATE_DIFF(LAST_DAY(CAL_DAY_DT),to_date('01-JAN-1900','DD-MON-YYYY'),'DD')) +
2415021'
```

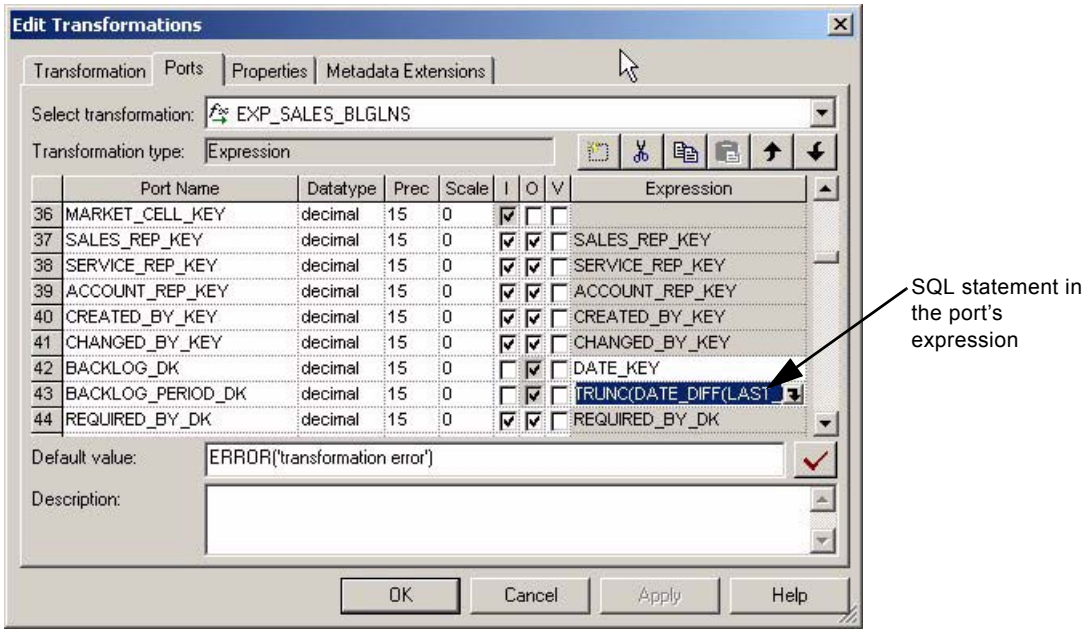
You can edit the backlog period date so that you can capture a more detailed backlog history with the following procedure. Possible periods include daily (CAL_DAY_DT), weekly (CAL_WEEK_DT), monthly (CAL_MONTH_DT), and quarterly (CAL_QTR_DT).

To redefine the backlog period date

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the applicable Sales Backlog Lines mapping.

3 Double-click the Expression transformation to display the BACKLOG_PERIOD_DK port.

The SQL statement in this port's expression contains the backlog period date, as shown in the following figure.



4 In the Ports tab, modify the default SQL statement for the BACKLOG_PERIOD_DK.

For example, if you want to store backlog history at the weekly level, replace the existing expression:

```
'trunc(DATE_DIFF(LAST_DAY(CAL_DAY_DT),to_date('01-JAN-1900','DD-MON-YYYY'),'DD'))
+ 2415021'
```

with this new expression:

```
'trunc(DATE_DIFF(CAL_WEEK_END_DT),to_date('01-JAN-1900','DD-MON-YYYY'),'DD')) +
2415021'
```

```
TO_CHAR(CAL_WEEK_END_DT, 'J')
```

NOTE: The CAL_WEEK_END_DT is not prepackaged to be extracted from IA_DATES; you have to extract this data from IA_DATES and pass it to the Expression transformation so it can be used in this calculation.

5 Validate and save your changes to the repository.

Accounting for Negative Values for Orders, Invoices, and Picks

This task is a step in the Process of Configuring Sales for Oracle 11i on page 116.

By default, the Siebel Analytics Enterprise Data Warehouse does not use negative values in the quantity or amount columns for the IA_SALES_IVCLNS table or the IA_SALES_ORDLNS table. However, you can make these values negative using a column called VAR_NEGATIVE_SIGN. By default, this column has the value 1.0. To make the values negative, modify the column value to be -1.

For example, to account for a negative return value for a Return Material Authorization (RMA) or for a negative value in a credit memo, you can use a conditional statement to define the VAR_NEGATIVE_SIGN column.

Assume that the S14 column in SO_LINES_ALL table has been configured in Oracle 11i to have the value '30' if the order line is a return. You can use this identifier for returned orders as a condition for setting the VAR_NEGATIVE_SIGN column to be -1. To do this, you can modify the VAR_NEGATIVE_SIGN column's definition in the MPLT_SAO_SALES_ORDLNS as follows:

```
DECODE(INP_LINES_S14, 30, -1, 1)
```

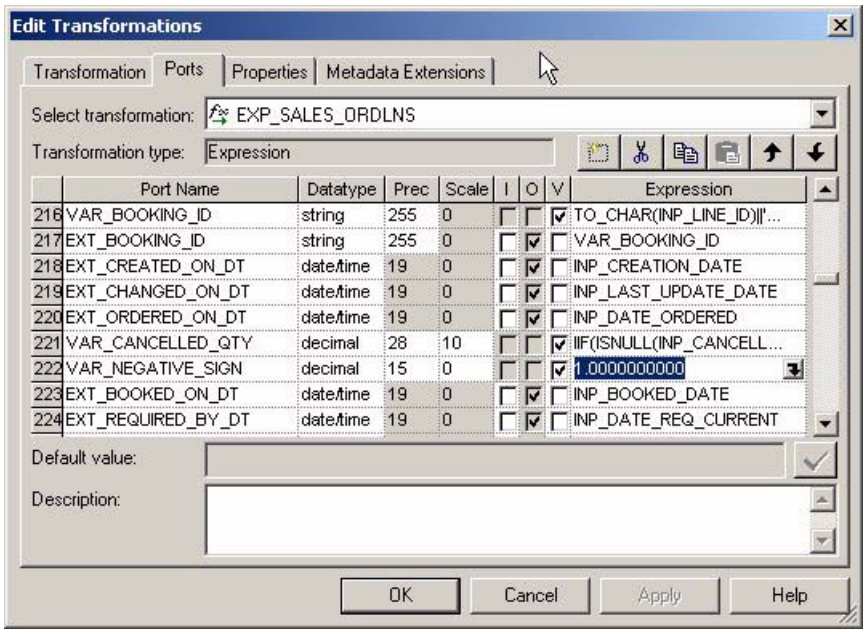
For Oracle 11i, the VAR_NEGATIVE_SIGN column is available in the following Source Adapters—MPLT_SAI_SALES_ORDLNS, MPLT_SAI_SALES_IVCLNS and MPLT_SAI_SALES_PCKLNS.

In Oracle 11i the VAR_NEGATIVE_SIGN column's value can be set based on the type of order line.

To configure mapplets to account for negative values

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the applicable Source Adapter mapplet.

3 Double-click the Expression transformation to open the Edit Transformation box, as shown in the following figure.



4 In the Ports tab, edit the VAR_NEGATIVE_SIGN port.

For example, if the S14 column in SO_LINES_ALL table has been configured in Oracle 11i to have the value '30' if the order line is a return, then you can use this identifier for returned orders as a condition for setting the VAR_NEGATIVE_SIGN column to be -1. To do so, you would set the VAR_NEGATIVE_SIGN column's definition as follows:

```
DECODE(INP_LINES_S14, 30, -1, 1)
```

5 Validate and save your changes to the repository.

Configuring Sales for Universal Source

This section contains Sales module configuration points that are specific to a universal source. Of the three functional areas in the Sales module, Sales Performance, Opportunity, and Pipeline are the only functional areas that have configuration points specific to a universal source.

Configuring the Quota Period Type

Sales quota information is stored at the sales contact, product, and quota period level in Sales Quota and Sales Quota History, and stored at the sales contact, product, and opportunity period level in the Sales Opportunity aggregate. Because the opportunity period for the aggregate table may be different from the quota period, you have to make modifications for both. This configuration point provides information on configuring the quota period type in the Sales Quota and Sales Quota History tables. For information on how the quota period impacts the Sales Opportunity aggregate table, see Configuring the Quota Period in the Sales Opportunity Aggregate Table on page 137.

Because a Universal Source provides this information, you can not provide a default time period. It is impossible to know the level at which you want to store your quota information. Therefore, you can decide the level at which you can store this type of information. The only caveat is that you use the same grain for both tables. For information on configuring the fact table, see Configuring the Quota Period in the Sales Opportunity Aggregate Table on page 137.

To accurately report sales quota information, you must specify the time period used. To flag the period at which you are tracking your sales quota information, you must specify the period type using the QOTA_PERIOD_TYPE_FLAG column in the file_sls_qota.csv flat file. By default, this flag accepts the following values:

- `W` (weekly)
- `M` (monthly)
- `Q` (quarterly)
- `Y` (yearly)

If you want to track the sales quotas at a different level, a new value can be provided.

Process of Configuring Sales for Post-Load Processing

To configure Sales for post-load processing, perform the following tasks:

Configuring the Quota Period in the Sales Opportunity Aggregate Table

This task is a step in the Process of Configuring Sales for Post-Load Processing on page 137.

Sales quota information is stored at the sales contact, product, and quota period level in the Sales Quota (IA_SLS_QOTA) and Sales Quota History (IA_SLS_QOTAHIST), and stored at the sales contact, product, and opportunity period level in the Sales Opportunity aggregate (IA_SLS_OPTY_A1). Because the opportunity period for the aggregate table may be different from the quota period, to configure the quota period, you have to make modifications for both cases. This topic provides information on configuring the quota period type in the Sales Opportunity aggregate table.

By default, the Sales Opportunity aggregate table stores data at the monthly level. Because quotas in the Sales Quota table are, by default, stored at the monthly level, these tables are in sync. However, if you change the period at which quota information is stored, you have to make changes here as well.

You can store quota information at whatever period you wish. To store this information using a different time frame, you must modify the SQL to retrieve the correct period start and end dates. You have to perform both of these changes in the Source Qualifier transformations of the applicable mappings.

Table 25 provides a list of all mappings with their corresponding Source Qualifier transformations.

Table 25. Source Qualifiers Containing Open Opportunity Calculations

Mapping	Source Qualifier Transformations
M_PLP_SLS_OPTY_A1_QUOTA_EXTRACT	SQ_SLS_OPTY_A1_QUOTA_EXTRACT
M_PLP_SLS_OPTY_A1_QUOTA_DERIVE	SQ_SLS_OPTY_A1_ QUOTA_DERIVE

To reset the quota period for the Sales Opportunity Aggregate table

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_SLS_OPTY_A1_QUOTA_EXTRACT mapping.
3. Open the SQ_SLS_OPTY_A1_QUOTA_EXTRACT Source Qualifier.
4. In the SELECT clause, modify the statement accordingly.

   For example, if you want to change the monthly period start and end dates to quarterly start and end dates, then you should replace all instances of the following default values:

   IA_DATES.CAL_MONTH_START_DT

   IA_DATES.CAL_MONTH_END_DT

   You should replace the default values with the following:

   IA_DATES.CAL_QTR_START_DT

   IA_DATES.CAL_QTR_END_DT
5. In the WHERE clause, you specify the applicable quota period type.

   For example, if you want to change the quota period type from monthly to quarterly, replace the default value for QUOTA.QOTA_PERD_TYP_FLAG (‘M’) with ‘Q’ as shown in the above figure.

6 In the WHERE clause, you also replace two default conditions.

If you change the quota period from monthly to quarterly, replace:

```
QUOTA.QOTA_PERD_TYP_FLAG = 'M'
```

with

```
QUOTA.QOTA_PERD_TYP_FLAG = 'Q'
```

then, replace

```
PL.PERIOD_START_DT = IA_DATES.CAL_MONTH_START_DT AND
PL.PERIOD_END_DT = IA_DATES.CAL_MONTH_END_DT
```

with

```
PL.PERIOD_START_DT = IA_DATES.CAL_QTR_START_DT AND
PL.PERIOD_END_DT = IA_DATES.CAL_QTR_END_DT
```

7 Repeat Step 5 and Step 6 for the SQ_SLS_OPTY_A1_ QUOTA_DERIVE Source Qualifier transformation in the M_PLP_SLS_OPTY_A1_QUOTA_DERIVE mapping.

8 Validate and save your changes to the repository.

Configuring Open Opportunity Calculations

This task is a step in the Process of Configuring Sales for Post-Load Processing on page 137.

Generally, *open opportunities* are those that are still in the sales pipeline. The Siebel Analytics Enterprise Data Warehouse uses the last period, which by default is the previous month, to determine if opportunities are open. In the Siebel Analytics Enterprise Data Warehouse, open opportunities should be denoted by setting the appropriate values for two columns: IA_STAT_TYPE_CODE and IA_STAT_SUBT_CODE. Table 26 provides a list of prepackaged domain values for the IA_STAT_TYPE_CODE and IA_STAT_SUBT_CODE columns, which stores the opportunity status and substatuses, respectively.

Table 26. Domain Values for Closed Opportunities

IA_STATUS.IA_STAT_TYPE_CODE	IA_STATUS.IA_STAT_SUBT_CODE	Description of IA_STAT_SUBT_CODE
OPEN	IN PROCESS	If the opportunity is still being pursued by both the company and the customer/prospect.
OPEN	PURSUE LATER	If the opportunity is put on hold until a future date.

NOTE: The Open Opportunity calculation is configurable in the Siebel Analytics Enterprise Data Warehouse. If you configure the calculation in the warehouse, ALL metrics that use open opportunities in their calculation are affected.

The Siebel Analytics Enterprise Data Warehouse provides a definition for open opportunities, however the definition can be modified to match your company's business requirements. For example, if the substatus 'PURSUE LATER' should not be considered for open opportunities, then you can specify this as a filter in the post-load calculation in the corresponding Source Qualifier transformation for each applicable mapping. Table 27 provides a list of all applicable mappings and their corresponding Source Qualifier transformations.

Table 27. Source Qualifiers Containing Open Opportunity Calculations

Mapping	Source Qualifier Transformation
M_PLP_SLS_OPTY_A1_OPEN_OPTY_EXTRACT	SQ_SLS_OPTY_A1_OPEN_OPTY_EXTRACT
M_PLP_SLS_OPTY_A1_OPEN_OPTY_DERIVE	SQ_SLS_OPTY_A1_OPEN_OPTY_DERIVE
M_PLP_SLS_OPTY_A1_OPEN_OPTY_UP_DN_DERIVE	SQ_SLS_OPTY_A1_OPEN_OPTY_UP_DN_DERIVE

To redefine the open opportunity calculations

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the applicable mapping.

   You must perform this procedure for each of the following mappings:

   - M_PLP_SLS_OPTY_A1_OPEN_OPTY_EXTRACT,
   - M_PLP_SLS_OPTY_A1_OPEN_OPTY_DERIVE, and
   - M_PLP_SLS_OPTY_A1_OPEN_OPTY_UP_DN_DERIVE.
3. Open the mapping's corresponding Source Qualifier transformation.
4. In the WHERE clause, add the appropriate filter condition.

   For example, to exclude opportunities with the substatus 'PURSUE LATER' add the following condition:

   ```
   AND IA_STATUS.IA_STAT_SUBT_CODE != 'PURSUE LATER'
   ```
5. Repeat the previous steps for each applicable mapping.
6. Validate and save your changes to the repository.

Configuring Closed Opportunity Calculations

This task is a step in the Process of Configuring Sales for Post-Load Processing on page 137.

Generally, closed opportunities are no longer in the sales pipeline. The Siebel Analytics Enterprise Data Warehouse uses the current period, which by default is the current month, to determine if opportunities are closed. In the Siebel Analytics Enterprise Data Warehouse, closed opportunities should be denoted by setting the appropriate values for two columns: IA_STAT_TYPE_CODE and IA_STAT_SUBT_CODE. Table 28 provides a list of prepackaged domain values for the IA_STAT_TYPE_CODE and IA_STAT_SUBT_CODE columns, which stores the opportunity status and substatuses, respectively.

Table 28. Domain Values for Closed Opportunities

IA_STATUS.IA_STAT_TYPE_CODE	IA_STATUS.IA_STAT_SUBT_CODE	Description of IA_STAT_SUBT_CODE
CLOSED	WON	Use this subcode if your company won the sale.
CLOSED	LOST	Use this subcode if your company lost the sale.
CLOSED	OTHERS	If the opportunity does not fit into any of the other categories, then use the OTHER subcode.

Although the Siebel Analytics Enterprise Data Warehouse provides a definition for closed opportunities, the definition can be modified to fit your company's business requirements. For example, if the substatus 'OTHER' should not be considered for closed opportunities, then you can specify this as a filter in the post-load calculation in the corresponding Source Qualifier transformation for each applicable mapping. Table 27 provides a list of all applicable mappings and their corresponding Source Qualifier transformations.

NOTE: The Closed Opportunity calculation is configurable in the Siebel Analytics Enterprise Data Warehouse. If you configure the calculation in the warehouse, ALL metrics that use closed opportunities in their calculation are affected.

To redefine the closed opportunity calculations

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_SLS_OPTY_A1_CLOSED_OPTY_DERIVE mapping.
3. Open the SQ_SLS_OPTY_A1_CLOSED_OPTY_DERIVE Source Qualifier.
4. Modify the CASE statements in the following code:

```
SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'WON' THEN

1 ELSE 0 END) WON_OPTY_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'LOST' THEN

1 ELSE 0 END) LOST_OPTY_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' THEN
```

```
1 ELSE 0 END) CLOSED_OPTY_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'WON' THEN

OPTY.CURR_PROD_CNT ELSE 0 END) WON_PROD_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'LOST' THEN

OPTY.CURR_PROD_CNT ELSE 0 END) LOST_PROD_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' THEN

OPTY.CURR_PROD_CNT ELSE 0 END) CLOSED_PROD_CNT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'WON' THEN

OPTY.CLOSED_REV_GRP_AMT ELSE 0 END) WON_REV_AMT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'LOST' THEN

OPTY.CURR_EXPRV_GRP_AMT ELSE 0 END) LOST_EXPRV_AMT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED'THEN

OPTY.CURR_EXPRV_GRP_AMT ELSE 0 END) CLOSED_EXPRV_AMT,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED'AND

IA_STAT_SUBT_CODE = 'WON' THEN

OPTY.OPTY_END_DK - OPTY.OPTY_START_DK ELSE 0 END) WON_OPTY_DURN,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND

IA_STAT_SUBT_CODE = 'LOST' THEN

OPTY.OPTY_END_DK - OPTY.OPTY_START_DK ELSE 0 END) LOST_OPTY_DURN,

SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' THEN

OPTY.OPTY_END_DK - OPTY.OPTY_START_DK ELSE 0 END) CLOSED_OPTY_DURN,

AVG(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND
```

For example, to exclude opportunities with the sub-status ‘OTHERS’, add the highlighted condition to the default CASE statements, as shown in the following sample:

```
SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND IA_STAT_SUBT_CODE != 'OTHERS' THEN

1 ELSE 0 END) CLOSED_OPTY_CNT,
```

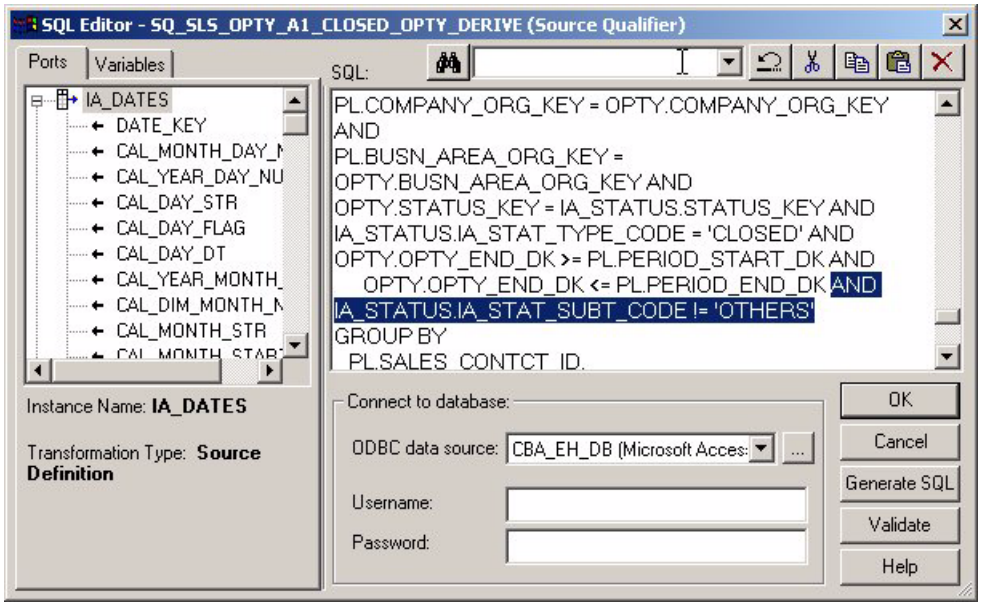
```
SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND IA_STAT_SUBT_CODE != 'OTHERS' THEN
OPTY.CURR_PROD_CNT ELSE 0 END) CLOSED_PROD_CNT,
SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND IA_STAT_SUBT_CODE != 'OTHERS' THEN
OPTY.CURR_EXPRV_GRP_AMT ELSE 0 END) CLOSED_EXPRV_AMT,
SUM(CASE WHEN IA_STAT_TYPE_CODE = 'CLOSED' AND IA_STAT_SUBT_CODE != 'OTHERS' THEN
OPTY.OPTY_END_DK - OPTY.OPTY_START_DK ELSE 0 END) CLOSED_OPTY_DURN
```

5 In the WHERE clause, add a filter condition.

For example, to exclude opportunities with the substatus 'OTHERS', add the following filter condition:

```
AND IA_STATUS.IA_STAT_SUBT_CODE != 'OTHERS'
```

The following figure illustrates this process.



6 Validate and save your changes to the repository.

Tracking Multiple Products Sold as One Package

This task is a step in the Process of Configuring Sales for Post-Load Processing on page 137.

The Sales Order Lines table contains two columns, ORDHD_KEY_ID and ORDLN_KEY_ID, that track individual products when they are grouped and sold as a single package. These two columns allow you to analyze the relationship of all products sold as a single unit. The ORDHD_KEY_ID column stores the Order ID of the entire sales order. The ORDLN_KEY_ID column stores the Line Item ID of the parent product.

For example, assume a customer purchases a package that includes a computer, scanner, and printer. In addition, the customer purchases a monitor separately. In this case, there are two parent items: the package and the monitor. The computer, scanner, and printer are all child orders of the parent order *package*, while the parent order *monitor* is a single-item purchase.

Your data warehouse may store this sales information in the Sales Order Lines table as seen in Table 29. The ORDLN_KEY_ID field contains the Line Item ID of the parent product in order to maintain the relationship between the parent and child products in a package. In this example, the ORDLN_KEY_ID field is Line_1 for each of the three child products (A1, A2, A3) that were sold as a part of the parent package, Parent A.

Table 29. Sales Order Table Columns With Parent/Child Relationships

Key_ID	SALES_ORDER_NUM	PRODUCT_ID	ORDHD_KEY_ID	ORDLN_KEY_ID	Relationship (Not a column in the table.)
Line_1	1000	Package	1000	Line_1	Parent A
Line_2	1000	Computer	1000	Line_1	Child A1
Line_3	1000	Scanner	1000	Line_1	Child A2
Line_4	1000	Printer	1000	Line_1	Child A3
Line_5	1000	Monitor	1000	Line_5	Parent B (no children)

In contrast, if each of the four items described in Table 29 were bought individually, the ORDLN_KEY_ID would have a different Line Item ID for every row. In this case, the Sales Order Lines table would look like Table 30.

Table 30. Sales Order Table Columns Without Parent/Child Relationships

Key_ID	SALES_ORDER_NUM	PRODUCT_ID	ORDHD_KEY_ID	ORDLN_KEY_ID	Relationship (Not a column in the table.)
Line_1	1000	Computer	1000	Line_1	None
Line_2	1000	Scanner	1000	Line_2	None
Line_3	1000	Printer	1000	Line_3	None
Line_4	1000	Monitor	1000	Line_4	None

12 Configuring the Service Module

After the Service module is installed, you may need to configure certain objects to meet your business needs. This chapter includes the following sections:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Service

The Service module tracks incoming and outgoing contacts and their resulting activities, and the agents performing contact center tasks. The Service module is made up of these functional areas:

- Customer Service
- Contact Center and Agent Performance
- Service Delivery and Costs

For more information on the Service module, see the *Siebel Analytics Enterprise Applications User Guide*.

Configuring Service for Universal Source

This section contains Service module configuration points that are specific to a universal source.

Dimension Key Resolution

Dimension keys uniquely identify each record in a dimension table. The purpose of dimension keys is to relate one dimension table record to a fact table record. Thus, the dimension key must be stored in both the dimension table and the fact table and resolve using the dimension table's KEY_ID and SOURCE_ID column values.

For universal business adapters, users supply the dimension KEY_ID and SOURCE_ID column values through a flat file interface. The same values for KEY_ID and SOURCE_ID are expected in both the dimension and fact business adapters so that the correct dimension key is resolved and loaded into its fact table.

Overview of the Dimension Key Resolution Process for Universal Sources

The dimension key resolution process entails two steps:

1. Run the dimension table workflows to extract and load dimension records.

   The dimension load mapping automatically creates a surrogate key for each record in the dimension table. This surrogate key value populates the dimension table's primary key column, which is referred to as the *dimension key.* Similar to the KEY_ID column, which uniquely identifies the record within the source system, the dimension key uniquely identifies the record in the data warehouse dimension table.

2. Run the fact table workflows to extract and load fact records.

   Records must contain the dimension ID column values for each fact record; these values must be the same values as the KEY_ID in the corresponding dimension tables.

The following sections describe these two steps in more detail by focusing on one fact table (IA_REP_ACTVTS) and one dimension table (IA_EVENTS). However, this process applies to all fact and dimension tables joined by a dimension key.

Loading the Dimension Table

Loading the IA_EVENTS_TYPE dimension table requires the following ETL processes:

1. The M_F_EVENT_TYPES_EXTRACT mapping extracts the data from file_event_types.csv and populates the TF_EVENT_TYPES staging table.
2. The M_F_EVENT_TYPES_LOAD mapping sources data from the staging table and passes it over to the Analytic Data Interchange (ADI). The ADI generates the surrogate key for each record in the staging table, then inserts it into IA_EVENTS_TYPE target table.

Loading the Fact Table

Loading the IA_REP_ACTVTS fact table requires the following ETL processes:

1. The M_F_REP_ACTVTS_EXTRACT mapping extracts the data from file_rep_actvts.csv and populates the TF_REP_ACTVTS staging table.
2. The M_F_REP_ACTVTS_LOAD mapping sources the data from the staging table, and the fact ADI mapplet resolves the dimension key by doing a lookup on IA_EVENT_TYPES using the values supplied in the ACTIVITY_TYPE_ID column and the SOURCE_ID column. Then, the ADI populates the IA_REP_ACTVTS fact table.

Given that you are required to supply the dimension *_ID values through the Universal Interface flat file, it is critical that you supply the same value for the KEY_ID in the dimension table and the corresponding *_ID field in the joined fact table. In addition, you must verify that the SOURCE_ID column values match (for Universal Sources, the value of the SOURCE_ID column is 'GENERIC'). If you supply different values for the two tables, the fact table load mapping is not able to resolve the dimension key. As a result, you cannot perform queries on the fact table using that dimension.

Each of the following sections describe particular dimension and fact table joins.

Configuring the ACTIVITY_TYPE_ID in file_rep_actvts.csv

Contact center representatives engage in various activities, such as logging into the Automated Call Distributor (ACD) system to handle customer calls, taking a scheduled break, taking an unscheduled break, and so on. The information regarding these activities is stored in the IA_REP_ACTVTS fact table. The ACTIVITY_TYPE_KEY in IA_REP_ACTVTS identifies the nature of the activity. This key is resolved using the IA_EVENT_TYPES table.

NOTE: IA_EVENT_TYPES is a dimension class table which tracks various types of events, such as contact center activities for the Service module. For example, some activities are categorized under 'CONTACT CENTER REP ACTIVITY', which is a fixed domain value for the EVENT_CLASS column in IA_EVENT_TYPES.

To resolve the ACTIVITY_TYPE_KEY in IA_REP_ACTVTS, the IA_REP_ACTVTS and IA_EVENT_TYPES tables are joined through the ACTIVITY_TYPE_ID column and the SOURCE_ID column. For the ACTIVITY_TYPE_KEY dimension key to resolve properly in the IA_REP_ACTVTS fact table, you must verify that the ACTIVITY_TYPE_ID column and the SOURCE_ID column values in file_rep_actvts.csv file match with the KEY_ID column and the SOURCE_ID column values in the file_event_types.csv file. If the two do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the ACTIVITY_TYPE_ID column in file_rep_actvts.csv

1. Open the file_rep_actvts.csv and file_event_types.csv files.
2. Verify that:
   - The ACTIVITY_TYPE_ID value in file_rep_actvts.csv matches with a KEY_ID value in file_event_types.csv.
   - The SOURCE_ID value in file_rep_actvts.csv matches with a SOURCE_ID value in file_event_types.csv.
3. Save your changes to file_rep_actvts.csv file.
4. Run a test load for 10 records to verify that your dimension keys are getting resolved correctly in the IA_REP_ACTVTS fact table.

Configuring the CALL_TYPE_ID in file_acd_events.csv

There are various types of calls that either customers make to your organization, or your organization makes to a customer. These call types include: placing an order for a product or service, lodging a complaint, inquiring about a product or service, responding to a marketing campaign, following up on a customer inquiry, following up on a customer complaint, surveying a customer, and so on. These call types and their associated information are tracked in the IA_ACD_EVENTS fact table. The CALL_TYPE_KEY in IA_ACD_EVENTS identifies the type of call. This key is resolved using the IA_EVENT_TYPES table.

NOTE: IA_EVENT_TYPES is a dimension class table which tracks various types of events and activities happening in a contact center. These events and activities are categorized under one fixed domain value class in the EVENT_CLASS column in the IA_EVENT_TYPES table. You can decide on your own domain values and place them in the file_event_types.csv file. It is recommended that you create a new 'CRM EVENTS' domain value under the EVENT_CLASS column.

To resolve the CALL_TYPE_KEY in IA_ACD_EVENTS and IA_EVENT_TYPES, the tables are joined through the CALL_TYPE_ID column and the SOURCE_ID column. For the CALL_TYPE_KEY dimension key to resolve properly in the IA_ACD_EVENTS fact table, you must verify that the CALL_TYPE_ID column and the SOURCE_ID column values in file_acd_events.csv file match with the KEY_ID column and the SOURCE_ID column values in the file_event_types.csv file. If the two do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the CALL_TYPE_ID column in file_acd_events.csv

1. Open the file_acd_events.csv and file_event_types.csv files.
2. Verify that:
   - The CALL_TYPE_ID value in file_acd_events.csv matches with a KEY_ID value in file_event_types.csv.
   - The SOURCE_ID value in file_acd_events.csv matches with a SOURCE_ID value in file_event_types.csv.
3. Save your changes to the file_acd_events.csv file.
4. Run a test load for 10 records to verify that your dimension keys are getting resolved correctly in the IA_ACD_EVENTS fact table.

Configuring the CALL_EVENT_TYPE_ID in file_acd_events.csv

In a call center, the Computer Telephony Integration (CTI) system tracks each activity associated with a call, including, but not limited to:

- Call placed in queue to be answered.
- Call answered by a contact representative.
- Call placed on hold by a contact representative.
- Call transferred from one contact representative to another.
- Call hung up by the customer.

The CALL_EVENT_TYPE_KEY in IA_ACD_EVENTS identifies this activity. This key is resolved using the IA_EVENT_TYPES table.

NOTE: IA_EVENT_TYPES is a dimension class table that tracks various types of events and activities happening in a contact center. These events and activities are categorized under one fixed domain value class in the EVENT_CLASS column in the IA_EVENT_TYPES table. You can decide on your own domain values and place them in the file_event_types.csv. It is recommended that you create a new 'ACD EVENTS' domain value under the EVENT_CLASS column.

To resolve the CALL_EVENT_TYPE_KEY in IA_ACD_EVENTS, the IA_ACD_EVENTS and IA_EVENT_TYPES tables are joined through the CALL_EVENT_TYPE_ID column and the SOURCE_ID column. For the CALL_EVENT_TYPE_KEY dimension key to resolve properly in the IA_ACD_EVENTS fact table, you must verify that the CALL_EVENT_TYPE_ID column and the SOURCE_ID column values in file_acd_events.csv file match with the KEY_ID column and the SOURCE_ID column values in the file_event_types.csv file. If the two do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the CALL_EVENT_TYPE_ID column in file_acd_events.csv

1. Open the file_acd_events.csv and file_event_types.csv files.
2. Verify that:
   - The CALL_EVENT_TYPE_ID value in file_acd_events.csv matches with a KEY_ID value in file_event_types.csv.
   - The SOURCE_ID value in file_acd_events.csv matches with a SOURCE_ID value in file_event_types.csv.
3. Save your changes to the file_acd_events.csv file.
4. Run a test load for 10 records to verify that your dimension keys are getting resolved correctly in the IA_ACD_EVENTS fact table.

Configuring the CONTACT_TYPE_ID in file_cntctrep_snp.csv and file_cntct_snpsht.csv

In a contact center, typically, there are various reasons for which customers contact an organization or the organization contacts a customer. Regardless of whether the contact is inbound or outbound, the contact can happen through various channels, such as phone, online chat, email, fax, and so on. Some of the reasons that a customer contacts your organization include to:

- Place an order for a product or service.
- Lodge a complaint.
- Inquire about a product offering.

Some of the reasons why your organization might contact a customer include to:

- Perform a customer satisfaction survey.
- Follow up on an inquiry.

These contact reasons can be classified as *types of calls*. The call types and the associated information are tracked in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables. The CONTACT_TYPE_KEY in IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT identifies the type of call. This key is resolved using the IA_EVENT_TYPES table.

NOTE: IA_EVENT_TYPES is a dimension class table that tracks various types of events and activities happening in a contact center. These events and activities are categorized under one fixed domain value class in the EVENT_CLASS column in the IA_EVENT_TYPES table. You can decide on your own domain values and place them in the file_event_types.csv file. It is recommended that you create a new 'CRM EVENTS' domain value under the EVENT_CLASS column.

To resolve the CONTACT_TYPE_KEY in IA_CNTCTREP_SNP and in IA_CNTCT_SNPSHT, the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT tables are joined with IA_EVENT_TYPES separately through the CONTACT_TYPE_ID column and the SOURCE_ID column.

For the CONTACT_TYPE_KEY dimension key to resolve properly in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables, you must verify that the CONTACT_TYPE_ID column and the SOURCE_ID column values in the file_cntctrep_snp.csv and file_cntct_snpsht.csv files match with the KEY_ID column and the SOURCE_ID column values in the file_event_types.csv file. If they do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the CONTACT_TYPE_ID column in file_cntct_snpsht.csv and file_cntctrep_snp

1. Open the file_cntct_snpsht.csv and file_event_types.csv files.
2. Verify that:
   - The CONTACT_TYPE_ID value in file_cntct_snpsht.csv matches with a KEY_ID value in file_event_types.csv.
   - The SOURCE_ID value in file_cntct_snpsht.csv matches with a SOURCE_ID value in file_event_types.csv.
3. Save your changes to the file_cntct_snpsht.csv file.
4. Run a test load for 10 records to verify that your dimension keys are getting resolved in the IA_CNTCT_SNPSHT fact table.
5. Repeat the previous steps for the file_cntctrep_snp.csv file.

Configuring the ACW_ACT_TYPE_ID in file_cntctrep_snp.csv and file_cntct_snpsht.csv

On many occasions, a contact representative might be required to perform certain tasks after the contact. These tasks might include creating a follow-up action item list, dispatching the case from the contact to a particular group, and so on. These activities are known as *after-call work* activities. The nature of the after call work activity and the associated information are tracked in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables. You can resolve the ACW_ACT_TYPE_KEY in IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT by using the IA_EVENT_TYPES table.

NOTE: IA_EVENT_TYPES is a dimension class table that tracks various types of events and activities happening in a contact center. These events and activities are categorized under one fixed domain value class in the EVENT_CLASS column in IA_EVENT_TYPES table. You can decide on your own domain values and place them in the file_event_types.csv file. It is recommended that you create a new 'ACW EVENTS' domain value under the EVENT_CLASS column.

To resolve the ACW_ACT_TYPE_KEY column in IA_CNTCTREP_SNP and in IA_CNTCT_SNPSHT, the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT tables are joined with IA_EVENT_TYPES separately through the ACW_ACT_TYPE_ID column and the SOURCE_ID column. For the ACW_ACT_TYPE_KEY dimension key to resolve properly in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables, you must verify that the ACW_TYPE_TYPE_ID column and the SOURCE_ID column values in the file_cntctrep_snp.csv and file_cntct_snpsht.csv files match with the KEY_ID column and the SOURCE_ID column values in the file_event_types.csv file. If they do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the ACW_ACT_TYPE_ID column in file_cntct_snpsht.csv and file_cntctrep_snp

1. Open the file_cntct_snpsht.csv and file_event_types.csv files.
2. Verify that:
   - The ACW_ACT_TYPE_ID value in file_cntct_snpsht.csv matches with a KEY_ID value in file_event_types.csv.
   - The SOURCE_ID value in file_cntct_snpsht.csv matches with a SOURCE_ID value in file_event_types.csv.
3. Save your changes to the file_cntct_snpsht.csv file.
4. Run a test load for 10 records to verify that your dimension keys are getting resolved in the IA_CNTCT_SNPSHT fact table.
5. Repeat the previous steps for the file_cntctrep_snp.csv.

Configuring the CNTCT_STATUS_ID in file_cntctrep_snp.csv and file_cntct_snpsht.csv

All contacts made either by the customer to your organization, or by your organization to a customer, end with a definite status, such as customer abandoned call before contact, customer abandoned call during contact, contact completed, and so on. The various statuses of a contact and its associated information are tracked in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables. The CNTCT_STATUS_KEY in IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT identify the status of a contact. This key is resolved using the IA_STATUS table.

The IA_STATUS table tracks attributes associated with the statuses of purchase orders, requisitions, shipments, inventory, customer cases, contact statuses and other documents. There could be several classes of status such as order status, requisition status, schedule status, document status, case status, call status, and so on. Depending on the class of transactions, the table could have different categories of values. The status class is identified by the column STATUS_TYPE. For contact statuses, the STATUS_TYPE column must contain the fixed domain value class 'CONTACT STATUS'.

To resolve the CNTCT_STATUS_KEY column in IA_CNTCTREP_SNP and in IA_CNTCT_SNPSHT, the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT tables are joined with IA_STATUS separately through the CNTCT_STATUS_ID column and the SOURCE_ID column. For the CNTCT_STATUS_KEY dimension key to resolve properly in the IA_CNTCTREP_SNP and IA_CNTCT_SNPSHT fact tables, you must verify that the CNTCT_STATUS_ID column and the SOURCE_ID column values in file_cntctrep_snp.csv and file_cntct_snpsht.csv match with the KEY_ID column and the SOURCE_ID column values in the file_status.csv file. If they do not match for a particular record, the fact load mapping does not properly resolve the dimension key. As a result, the fact table does not store the dimension key for that record.

To configure the CNTCT_STATUS_ID column in file_cntct_snpsht.csv and file_cntctrep_snp

1. Open the file_cntct_snpsht.csv and file_status.csv files.
2. Verify that:
   - The CNTCT_STATUS_ID value in file_cntct_snpsht.csv matches with a KEY_ID value in file_status.csv.
   - The SOURCE_ID value in file_cntct_snpsht.csv matches with a SOURCE_ID value in file_status.csv.
3. Save your changes to the file_cntct_snpsht.csv file.
4. Run a test load for 10 records to verify that your dimension keys are resolved in the IA_CNTCT_SNPSHT fact table.
5. Repeat the previous steps for the file_cntctrep_snp.csv file.

Configuring Dates and Times

Local date and time in Service module calculations and analysis refers to the date and time at the contact's location. However, given different time zones and reporting requirements, you may need to use another date and time. To do so, you must supply these values through the flat file interface.

For example, when doing any period-based calculations or analysis on representative activities, prepackaged logic uses the ACTIVITY_START_LDT and ACTIVITY_END_LDT local date column values from the IA_REP_ACTVTS table. However, if you do not want to use the local dates, then you should pass the new dates in the ACTIVITY_START_DT and ACTIVITY_END_DT columns into the file_rep_actvts.csv flat file interface.

If you change the dates and times, you must do it consistently for all rows in the given table. For example, you cannot have some rows using local dates and times, while other rows are use a different time zone. Table 31 provides a list of the applicable local date columns for each relevant table.

Table 31. Providing New Dates in the Flat File Interface

Source (flat file interface)	Applicable Date Column	Table Using the Local Date
file_rep_actvts.csv	ACTIVITY_START_DT ACTIVITY_END_DT	IA_REP_ACTVTS
file_acd_events.csv	EVENT_START_DT EVENT_END_DT	IA_ACD_EVENTS
files_ cntctrep_snp.csv	CNTCT_START_DT CNTCT_END_DT	IA_CNTCTREP_SNP
file_cntct_snpsht.csv	CNTCT_START_DT CNTCT_END_DT	IA_CNTCT_SNPSHT
file_case_actvts.csv	ACTIVITY_DT	IA_CASE_ACTVTS

To provide new dates in the flat file interface

1. Open the applicable flat file interface.
2. In the flat file interface, input the new dates in the *_DT fields.
3. Save your changes to the flat file interface.
4. Run a test load for 10 records to verify that your new dates are loaded into the applicable table.

Configuring Flags

Many of the fact and dimension tables within the Service module use flag fields to provide value-added information pertaining to a contact or contact representative. These flag fields are configurable and include the following:

- SCHEDULE_BRK_FLG
- CONSULT_ FLAG
- CONFERENCE_FLAG
- PERTINENT_ INFO_FLG

- CNTCT_MTCH_FLAG
- IVR_FLAG
- INTRNL_ACTV_FLAG

The possible values for these flag fields in the data warehouse tables are 'Y' or 'N'. However, there is a conversion process to set these values. If you input any value other than 'N' in the flat file interface, the flag defaults to 'Y'. On the other hand, if you supply 'N', then the flag retains 'N' as the flag's value.

If you want to change this default logic, you can do so by changing the expression clause in the Expression transformation within the extract mapping. For example, if you want to change the default value of flag fields to 'N', so that any input value for the flag fields in the flat file interface other than 'Y' should be set to 'N', then you have to change the expression clause from:

```
IIF(<name_of_flag_port> != 'N', 'Y', 'N')
```

to:

```
IIF(<name_of_flag_port> != 'Y', 'N', 'Y')
```

See Table 32 for a list of all flags that can be reconfigured, the corresponding Expression transformations and mappings that contains the flag's default definition, and a description of each flag's value.

Table 32. Configurable Flag Values and Descriptions

Flag	Applicable Mappings and Expression Transformations	Flag Values	Descriptions
SCHEDULE_ BRK_FLG	M_F_REP_ACTVTS_EXTRACT EXP_REP_ACTVTS_EXTRACT	Y	Indicates that the break taken by the contact representative was a scheduled break.
		N	Indicates that the break taken by the contact representative was a nonscheduled break.
CONSULT_ FLAG	M_F_CNTCTREP_SNP_EXTRACT EXP_CNTCTREP_SNP_EXTRACT	Y	Indicates that the contact representative consulted with other contact representatives during the course of the call or contact.
		N	Indicates that the contact representative did not consult with other contact representatives during the course of the call or contact.
CONFERENCE_FLAG	M_F_CNTCTREP_SNP_EXTRACT EXP_CNTCTREP_SNP_EXTRACT	Y	Indicates that the contact representative conferenced with other contact representatives during the course of the call or contact.
		N	Indicates that the representative did not conference with other representatives during the course of the call or contact.
PERTINENT_ INFO_FLG	M_F_CNTCTREP_SNP_EXTRACT EXP_CNTCTREP_SNP_EXTRACT	Y	Indicates that the pertinent information was available for the contact.
		N	Indicates that the pertinent information was not available for the contact.
PERTINENT_ INFO_FLG	M_F_CNTCT_SNPSHT_EXTRACT EXP_CNTCT_SNPSHT_EXTRACT	Y	Indicates that the pertinent information was available for the contact.
		N	Indicates that the pertinent information was not available for the contact.

Table 32. Configurable Flag Values and Descriptions

Flag	Applicable Mappings and Expression Transformations	Flag Values	Descriptions
CNTCT_MTCH_FLAG	M_F_CNTCTREP_SNP_EXTRACT EXP_CNTCTREP_SNP_EXTRACT	Y	Indicates that the contact was matched with the existing customer data using Customer Entered Digits (CED), such as PIN numbers, account numbers, or social security numbers.
		N	Indicates that the contact could not be matched with the existing customer data using Customer Entered Digits (CED), such as PIN numbers, account numbers, or social security numbers.
CNTCT_MTCH_FLAG	M_F_CNTCT_SNPSHT_EXTRACT EXP_CNTCT_SNPSHT_EXTRACT	Y	Indicates that the contact was matched with the existing customer data using Customer Entered Digits (CED), such as PIN numbers, account numbers, or social security numbers.
		N	Indicates that the contact could not be matched with the existing customer data using Customer Entered Digits (CED), such as PIN numbers, account numbers, or social security numbers.
IVR_FLAG	M_F_ACD_EVENTS_EXTRACT EXP_ACD_EVENTS_EXTRACT	Y	Indicates that the event associated with the call was recorded in the Interactive Voice Response (IVR) system.
		N	Indicates that the event associated with the call was not recorded in the IVR system.
IVR_FLAG	M_F_CNTCT_SNPSHT_EXTRACT EXP_CNTCT_SNPSHT_EXTRACT	Y	Indicates that the call was routed through the IVR system.
		N	Indicates that the call was not routed through the IVR system.
INTRNL_ACTV_FLAG	M_F_CASE_ACTVTS_EXTRACT EXP_CASE_ACTVTS_EXTRACT	Y	Indicates that the activity carried out for a case is an internal activity within the organization.
		N	Indicates that the activity carried out for a case is an external activity, such as a customer activity.

To configure a flag

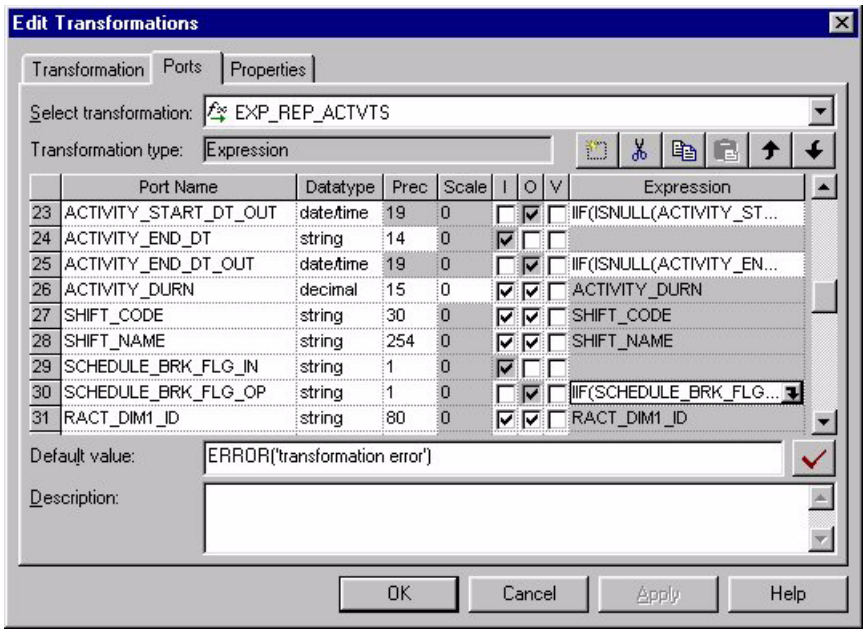
1 In PowerCenter Designer, open the Configuration for Universal Source folder.

2 Open the applicable mapping.

For example, if you want to change the default logic for the SCHEDULE_BRK_FLAG, open the M_F_REP_ACTVTS_EXTRACT mapping.

3 Double-click the applicable Expression transformation to display the applicable port.

For example, if you want to change the logic for the SCHEDULE_BRK_FLAG port, double-click the EXP_REP_ACTVTS_EXTRACT Expression transformation. Locate the SCHEDULE_BRK_FLAG port, as shown in the following figure.



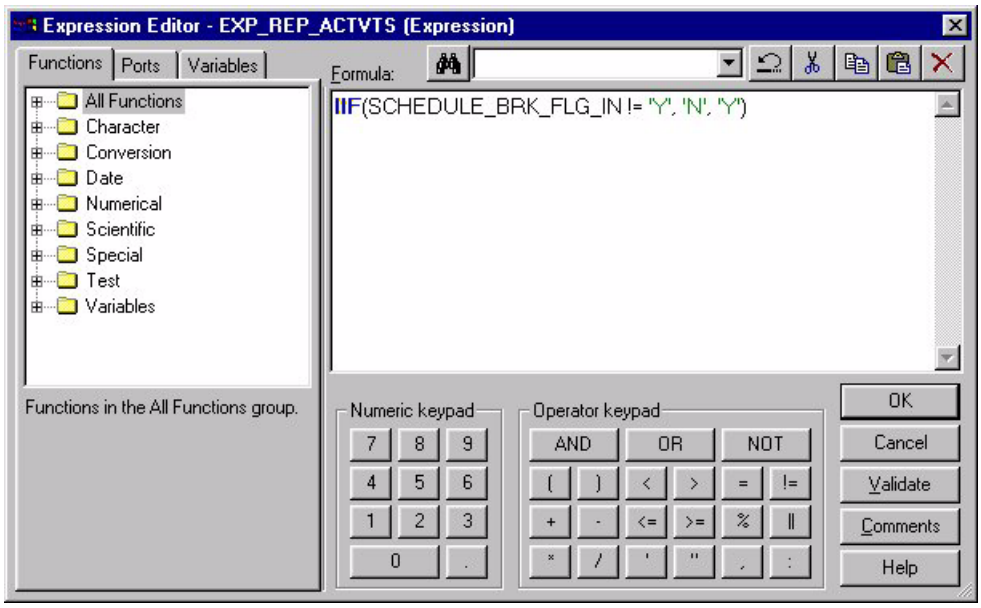
4 In the Ports tab, modify the default SQL for the flag port.

For example, if you wanted to change the default Schedule Break Flag logic:

```
IIF(SCHEDULE_BRK_FLG != 'N','Y','N')
```

You can change it as follows:

```
IIF(SCHEDULE_BRK_FLG != 'Y','N','Y')
```



5 Validate and save your changes to the repository.

Configuring Service for Post-Load Processing

This section contains Service module configuration points that are specific to post-load processing. Post-load processes source data from data warehouse tables to populates aggregate and key figure tables, as shown in Figure 11.

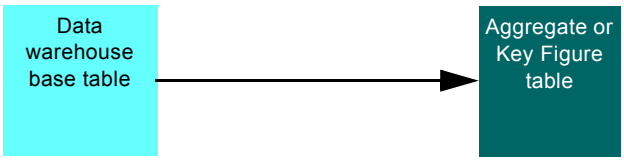


Figure 11. Post-Load Process for Universal Source

The following sections provide post-load processing configuration procedures:

- Excluding Representative Data from the Contact Representative Aggregate Table on page 159
- Excluding a Contact Group's Data from the Aggregate Table on page 161
- Configuring Contact References on page 164

Excluding Representative Data from the Contact Representative Aggregate Table

The IA_CC_REP_A1 aggregate table stores hourly aggregated contact statistics for each contact representative who handles customer contacts. You may want to exclude certain contact representatives from the aggregation process for various reasons. For example, you may want to exclude representatives who are on vacation, or who are working on special projects.

The default configuration calculates contact-related information for all contact representatives in the enterprise. If you choose to exclude a particular representative's data, you must modify the Select statement in each of the following Source Qualifier transformations shown in Table 33.

Table 33. Source Qualifiers Used to Exclude Representative Data from Aggregation Calculations

Mapping containing Source Qualifier	Source Qualifier
M_PLP_CC_REP_A1_REP_ACTVTS_EXTRACT	SQ_CC_REP_A1_REP_ACTVTS_EXTRACT
M_PLP_CC_REP_A1_CNTCTREP_SNP_EXTRACT	SQ_CC_REP_A1_CNTCTREP_SNP_EXTRACT
M_PLP_CC_REP_A1_CASE_ACTVTS_EXTRACT	SQ_CC_REP_A1_CASE_ACTVTS_EXTRACT
M_PLP_CC_REP_A1_EVENT_CASES_EXTRACT1	SQ_CC_REP_A1_EVENT_CASES_EXTRACT1
M_PLP_CC_REP_A1_EVENT_CASES_EXTRACT2	SQ_CC_REP_A1_EVENT_CASES_EXTRACT2
M_PLP_CC_REP_A1_EVENT_CASES_EXTRACT3	SQ_CC_REP_A1_EVENT_CASES_EXTRACT3

You must modify the 'N' EXCLUSION_IND_FLG portion in the following default Select statement.

```
SELECT DISTINCT

NU_CNTCTREP_SNP.CNTCT_REP_ID, NU_CNTCTREP_SNP.SUPERVISOR_ID,
NU_CNTCTREP_SNP.CNTCT_REP_GRP_ID,

NU_CNTCTREP_SNP.CNTCT_EMP_ORG_ID, NU_CNTCTREP_SNP.BUSN_AREA_ORG_ID,
NU_CNTCTREP_SNP.SALES_AREA_ORG_ID,

NU_CNTCTREP_SNP.SALES_GEO_ORG_ID, NU_CNTCTREP_SNP.COMPANY_ORG_ID,
NU_CNTCTREP_SNP.CNTCT_CNTR_LOC_ID,

NU_CNTCTREP_SNP.CHNL_TYPE_ID,A.KEY_ID AGGR_CHNL_TYPE_ID,
NU_CNTCTREP_SNP.SOURCE_ID,

'N' EXCLUSION_IND_FLG,

IA_DATES.CAL_DAY_DT CNTCT_DT,NU_CNTCTREP_SNP.CNTCT_START_DK
CNTCT_DK,IA_HOUR_OF_DAY.HOUR_KEY,

IA_HOUR_OF_DAY.FROM_TIME_KEY,IA_HOUR_OF_DAY.TO_TIME_KEY
```

as follows:

```
SELECT DISTINCT
```

```
NU_CNTCTREP_SNP.CNTCT_REP_ID, NU_CNTCTREP_SNP.SUPERVISOR_ID,
NU_CNTCTREP_SNP.CNTCT_REP_GRP_ID,

NU_CNTCTREP_SNP.CNTCT_EMP_ORG_ID, NU_CNTCTREP_SNP.BUSN_AREA_ORG_ID,
NU_CNTCTREP_SNP.SALES_AREA_ORG_ID,

NU_CNTCTREP_SNP.SALES_GEO_ORG_ID, NU_CNTCTREP_SNP.COMPANY_ORG_ID,
NU_CNTCTREP_SNP.CNTCT_CNTR_LOC_ID,

NU_CNTCTREP_SNP.CHNL_TYPE_ID,A.KEY_ID AGGR_CHNL_TYPE_ID,
NU_CNTCTREP_SNP.SOURCE_ID,

CASE WHEN NU_CNTCTREP_SNP.CNTCT_REP_ID = 'SALES REP' THEN 'Y' ELSE 'N' END
EXCLUSION_IND_FLG,

IA_DATES.CAL_DAY_DT CNTCT_DT,NU_CNTCTREP_SNP.CNTCT_START_DK
CNTCT_DK,IA_HOUR_OF_DAY.HOUR_KEY,

IA_HOUR_OF_DAY.FROM_TIME_KEY,IA_HOUR_OF_DAY.TO_TIME_KEY
```

To exclude data about specific representatives from the aggregation calculation

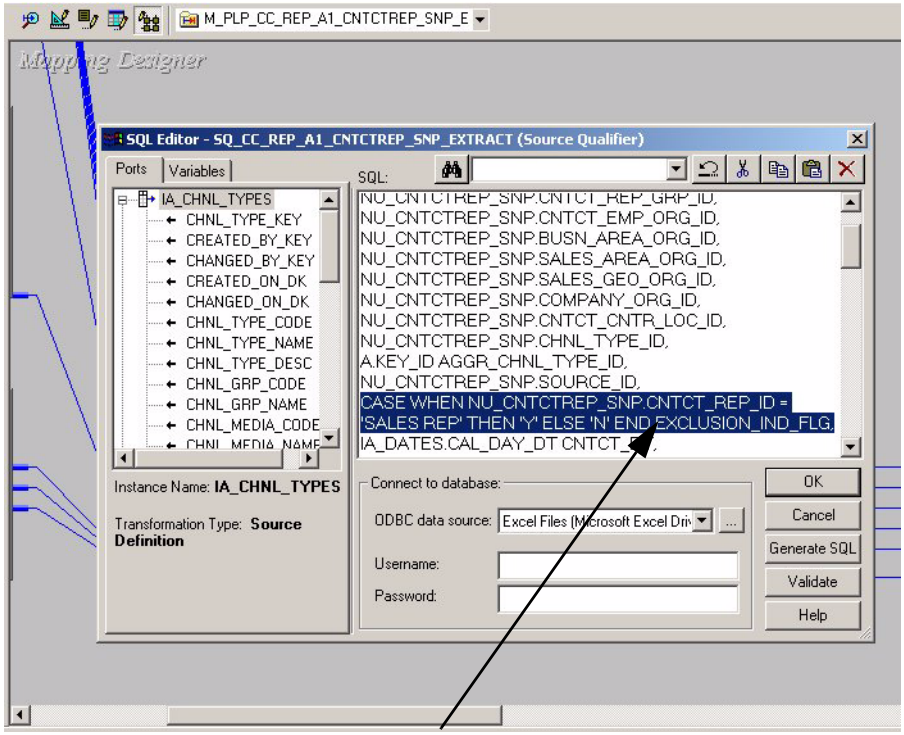
1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the applicable mapping.

   For example, open the M_PLP_CC_REP_A1_CNTCTREP_SNP_EXTRACT mapping.
3. Double-click the Source Qualifier to view the default SQL.

   For example, open the SQ_CC_REP_A1_CNTCTREP_SNP_EXTRACT Source Qualifier transformation.

4 Edit the SQL statement as necessary, but do not modify the WHERE statement.

For example, to exclude all Sales Representative data, edit the SQL as shown in the following figure.



Edit the default SQL to exclude sales representative data.

5 Modify every applicable mapping and Source Qualifier transformation.

For a list of all applicable mappings and Source Qualifier transformations, see Table 33 on page 159

6 Validate the mappings, and save your changes to the repository.

Excluding a Contact Group's Data from the Aggregate Table

The IA_ORGLOC_A1 aggregate table stores hourly aggregated contact statistics for each contact group that handles customer contacts. You may want to exclude certain contact groups from the aggregation calculation process for various reasons. For example, you may want to exclude a group that was recently dissolved or exclude a specific group to keep unauthorized viewers from accessing this group's information.

The default configuration aggregates the contact-related information for all groups or organizations within the enterprise. If you choose to exclude a particular group or organization's data, you must modify the Select statement in each of the Source Qualifier transformations shown in Table 34.

Table 34. Source Qualifiers Used to Exclude Group Data from Aggregation Calculations

Mapping Containing Source Qualifier	Source Qualifier
M_PLP_CC_ORGLOC_A1_REP_ACTVTS_EXTRACT	SQ_CC_ORGLOC_A1_REP_ACTVTS_EXTRACT
M_PLP_CC_ORGLOC_A1_CNTCTREP_SNP_EXTRACT	SQ_CC_ORGLOC_A1_CNTCTREP_SNP_EXTRACT
M_PLP_CC_ORGLOC_A1_CASE_ACTVTS_EXTRACT	SQ_CC_ORGLOC_A1_CASE_ACTVTS_EXTRACT
M_PLP_CC_ORGLOC_A1_EVENT_CASES_EXTRACT1	SQ_CC_ORGLOC_A1_EVENT_CASES_EXTRACT1
M_PLP_CC_ORGLOC_A1_EVENT_CASES_EXTRACT2	SQ_CC_ORGLOC_A1_EVENT_CASES_EXTRACT2
M_PLP_CC_ORGLOC_A1_EVENT_CASES_EXTRACT3	SQ_CC_ORGLOC_A1_EVENT_CASES_EXTRACT3

You must modify the `'N' EXCLUSION_IND_FLG,` portion in the following default Select statement.

```
SELECT DISTINCT

NU_CNTCTREP_SNP.CNTCT_REP_GRP_ID, NU_CNTCTREP_SNP.BUSN_AREA_ORG_ID,
NU_CNTCTREP_SNP.SALES_AREA_ORG_ID,

NU_CNTCTREP_SNP.SALES_GEO_ORG_ID, NU_CNTCTREP_SNP.COMPANY_ORG_ID,
NU_CNTCTREP_SNP.CNTCT_CNTR_LOC_ID,

NU_CNTCTREP_SNP.CHNL_TYPE_ID, A.KEY_ID AGGR_CHNL_TYPE_ID,
NU_CNTCTREP_SNP.SOURCE_ID,

'N' EXCLUSION_IND_FLG,

IA_DATES.CAL_DAY_DT CNTCT_DT, NU_CNTCTREP_SNP.CNTCT_START_DK,
IA_HOUR_OF_DAY.HOUR_KEY,

IA_HOUR_OF_DAY.FROM_TIME_KEY,IA_HOUR_OF_DAY.TO_TIME_KEY
```

as follows:

```
SELECT DISTINCT

NU_CNTCTREP_SNP.CNTCT_REP_GRP_ID, NU_CNTCTREP_SNP.BUSN_AREA_ORG_ID,
NU_CNTCTREP_SNP.SALES_AREA_ORG_ID,

NU_CNTCTREP_SNP.SALES_GEO_ORG_ID, NU_CNTCTREP_SNP.COMPANY_ORG_ID,
NU_CNTCTREP_SNP.CNTCT_CNTR_LOC_ID,

NU_CNTCTREP_SNP.CHNL_TYPE_ID, A.KEY_ID AGGR_CHNL_TYPE_ID,
NU_CNTCTREP_SNP.SOURCE_ID,

CASE WHEN NU_CNTCTREP_SNP.CNTCT_REP_GRP_ID = 'SALES GROUP' THEN 'Y' ELSE 'N' END
EXCLUSION_IND_FLG,
```

```
IA_DATES.CAL_DAY_DT CNTCT_DT, NU_CNTCTREP_SNP.CNTCT_START_DK,
IA_HOUR_OF_DAY.HOUR_KEY,

IA_HOUR_OF_DAY.FROM_TIME_KEY,IA_HOUR_OF_DAY.TO_TIME_KEY
```

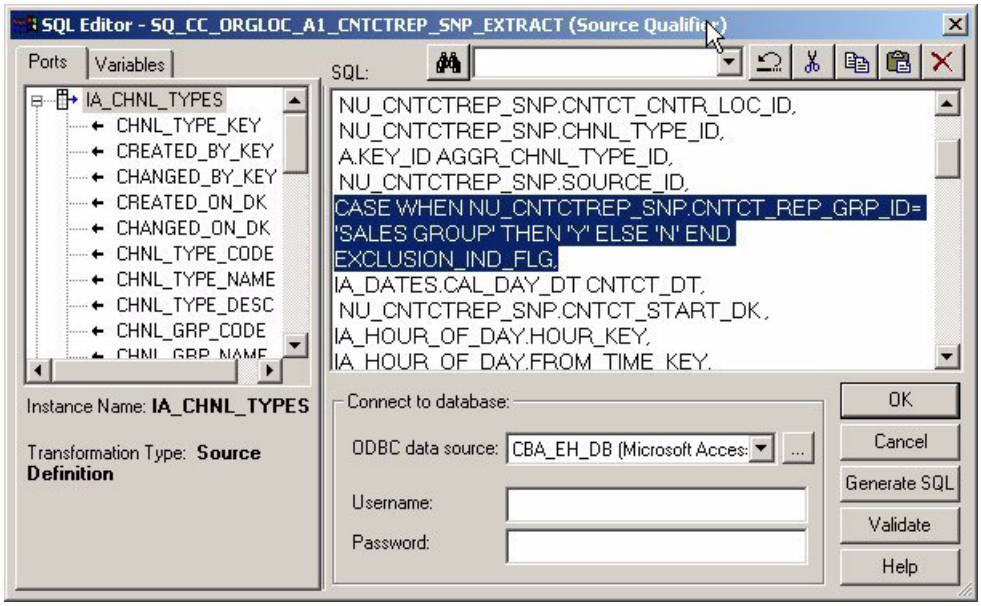
To exclude data about a specific group from the aggregation calculation

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the applicable mapping.

   For example, open the M_PLP_CC_ORGLOC_A1_CNTCTREP_SNP_EXTRACT mapping.
3. Double-click the Source Qualifier to view the default SQL.

   For example, open the SQ_CC_ORGLOC_A1_CNTCTREP_SNP_EXTRACT Source Qualifier transformation.
4. Edit the SQL statement, but do not modify the WHERE statement.

   For example, to exclude all Sales Group data, edit the SQL as shown in the following figure.



5. Modify every applicable mapping and Source Qualifier transformation.

   For a list of all applicable mappings and Source Qualifier transformations, see Table 34 on page 162.
6. Validate the mappings, and save your changes to the repository.

Configuring Contact References

IA_CNTCT_SNPSHT is populated from IA_CNTCTREP_SNP through a post-load process. IA_CNTCTREP_SNP is a fact table that has one record per contact representative per contact. IA_CNTCT_SNPSHT is a fact table that records the snapshot information for each contact. Therefore, there is only one record in this table for each contact. To show the relationship between these two tables, if a contact was handled by four representatives, there would be four records in IA_CNTCTREP_SNP and one record in IA_CNTCT_SNPSHT for that contact.

Because the IA_CNTCT_SNPSHT table only stores one record per contact, each record can only reference one contact representative, known as the *primary contact*. As a result, the contact is associated with other attributes related to that primary contact, such as the primary contact's contact group, contact center location, business unit, sales area, and sales territory, all of which are stored in IA_CNTCTREP_SNP. These other attributes are known as *contact references.*

In the prepackaged post-load process for loading IA_CNTCT_SNPSHT from IA_CNTCTREP_SNP, the primary contact is defaulted to the last contact representative who handled the call. Thus, the contact group, business unit, sales area, sales territory, and other contact references all refer to the last contact representative. These contact references are known as the *primary contact references.*

If you want to change the default configuration of the primary contact references so they point to the first representative who handled the call, instead of the last representative, then you must modify the WHERE clause in the SQL within the Source Qualifier transformation.

You must modify the portion in the following default WHERE clause.

```
WHERE

{

A LEFT OUTER JOIN IA_CNTCT_SNPSHT ON

A.CNTCT_NUM = IA_CNTCT_SNPSHT.CNTCT_NUM AND

A.CNTCT_CNTR_LOC_ID = IA_CNTCT_SNPSHT.CNTCT_CNTR_LOC_ID AND

A.SOURCE_ID = IA_CNTCT_SNPSHT.SOURCE_ID

} AND

A.CNTCT_NUM = PL_CNTCT_SNPSHT.CNTCT_NUM AND

A.CNTCT_CNTR_LOC_ID = PL_CNTCT_SNPSHT.CNTCT_CNTR_LOC_ID AND

A.SOURCE_ID = PL_CNTCT_SNPSHT.SOURCE_ID AND

A.CNTCT_END_DK = PL_CNTCT_SNPSHT.CNTCT_END_DK AND

A.CNTCT_END_TK = PL_CNTCT_SNPSHT.CNTCT_END_TK AND

A.CNTCT_NUM = B.CNTCT_NUM AND

A.CNTCT_CNTR_LOC_ID = B.CNTCT_CNTR_LOC_ID AND

A.SOURCE_ID = B.SOURCE_ID
```

as follows:

```
WHERE
{
A LEFT OUTER JOIN IA_CNTCT_SNPSHT ON
A.CNTCT_NUM = IA_CNTCT_SNPSHT.CNTCT_NUM AND
A.CNTCT_CNTR_LOC_ID = IA_CNTCT_SNPSHT.CNTCT_CNTR_LOC_ID AND
A.SOURCE_ID = IA_CNTCT_SNPSHT.SOURCE_ID
} AND
A.CNTCT_NUM = PL_CNTCT_SNPSHT.CNTCT_NUM AND
A.CNTCT_CNTR_LOC_ID = PL_CNTCT_SNPSHT.CNTCT_CNTR_LOC_ID AND
A.SOURCE_ID = PL_CNTCT_SNPSHT.SOURCE_ID AND
A.CNTCT_START_DK = PL_CNTCT_SNPSHT.CNTCT_START_DK AND
A.CNTCT_START_TK = PL_CNTCT_SNPSHT.CNTCT_START_TK AND
A.CNTCT_NUM = B.CNTCT_NUM AND
A.CNTCT_CNTR_LOC_ID = B.CNTCT_CNTR_LOC_ID AND
A.SOURCE_ID = B.SOURCE_ID
```

To configure the contact references

1. In PowerCenter Designer, open the Configuration for Post-Load Processing folder.
2. Open the applicable mapping.

   For example, open the M_PLP_CNTCT_SNPSHT_CNTCTREP_SNP_LOAD mapping.
3. Double-click to open the Source Qualifier.

   For example, open the SQ_CNTCT_SNPSHT_LOAD Source Qualifier transformation.
4. Edit the SQL statement.

   For example, in the SQ_CNTCT_SNPSHT_LOAD Source Qualifier transformation, you can change from the default last activity reference to the first activity by modifying the statement as follows:

   Change from:

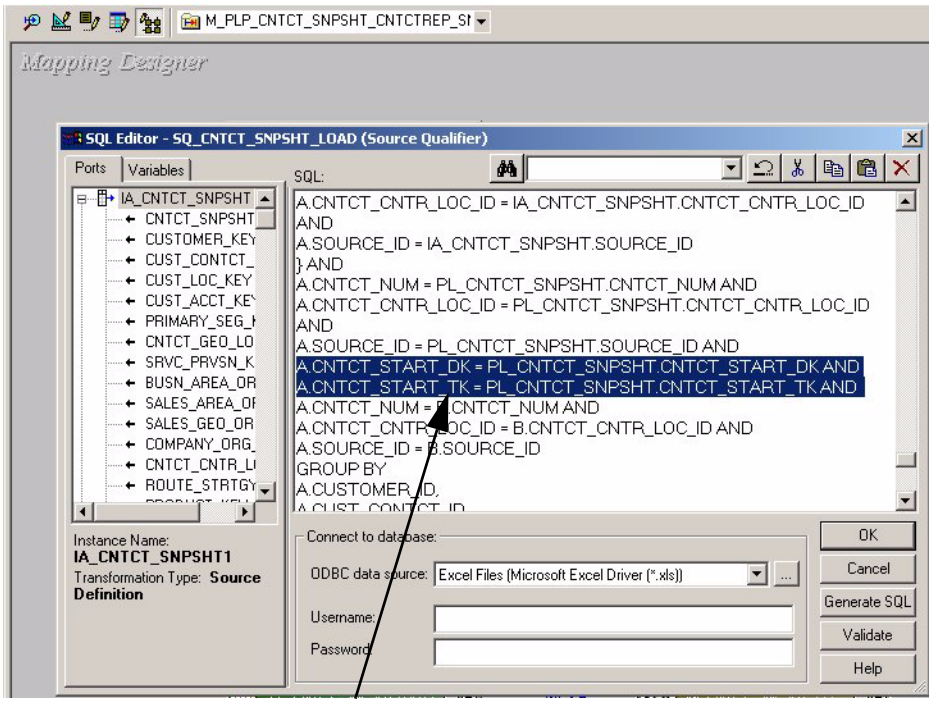
   ```
   A.CNTCT_END_DK = PL_CNTCT_SNPSHT.CNTCT_END_DK AND
   A.CNTCT_END_TK = PL_CNTCT_SNPSHT.CNTCT_END_TK AND
   ```

   Change to:

   ```
   A.CNTCT_START_DK = PL_CNTCT_SNPSHT.CNTCT_START_DK AND
   ```

```
A.CNTCT_START_TK = PL_CNTCT_SNPSHT.CNTCT_START_TK AND
```

For example, to use the first activity reference, rather than the last activity by default, edit the SQL statement as shown in the following figure.



Edit the SQL to reflect your contact reference.

5 Validate the mapping, and save your changes to the repository.

13 Configuring the Sourcing Module

After the Sourcing module is installed, you may need to configure certain objects for particular sources to meet your business needs. This chapter includes the following sections:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Sourcing

The Sourcing module allows you to analyze your procurement expenses and costs variances, as well as track your suppliers and supplier-related spending. It helps you to understand where and how spending occurs across your entire organization and to evaluate the price, quality, and delivery timing in procuring materials. Furthermore, it allows you to evaluate both direct and indirect spending (indirect spend being MRO and employee expenses). Through visibility into this complex data, you can reduce spending, intelligently select suppliers, and achieve a new balance and flexibility in your sourcing.

The Sourcing module is comprised of these functional areas:

- Expenses
- Spend
- Suppliers

The Expenses functional area contains targeted metrics and reports that examine travel and expense costs in relationship to your organization's overall spending patterns. In contrast to analyzing direct spending patterns, where you may review purchasing, Expenses examines indirect spending—the cost of employee related expenses.

The Spend functional area contains targeted reports and metrics that allow you to analyze both direct and indirect spend in detail to allow complete visibility of spending across your organization.

The Suppliers functional area contains targeted reports and metrics that allow you to analyze the timeliness, reliability, cost, and quality of goods provided by your suppliers.

For more information on the Sourcing module, see the *Siebel Analytics Enterprise Applications User Guide*.

Process of Configuring Sourcing for Oracle 11i

This section contains Sourcing configuration points that are specific to Oracle 11i. It contains the following topics:

Configuring the Handling of Currency Types (Oracle 11i)

The Sourcing module, like all other modules, uses the same method for handling currency conversions from document currency to local and group currencies. This guide provides a functional overview of how each of the local and group currencies is derived depending on what is supplied to the Siebel Analytics Enterprise Data Warehouse. For more information on how to configure various components that relate to local, document, and group currencies see Chapter 16, "Performing Cross-Module Configuration."

Configuring Region, State, and Country Name Definitions (Oracle 11i)

For Oracle 11i, you can reconfigure the region, state, and country names for supplier locations only. This topic provides detailed procedures for performing these tasks.

Configuring the Region Name Definition

This configuration point applies only to Plant locations, Storage locations, as well as Supplier locations. By default, the Region Name column (EXT_REGION_NAME) is populated using the same code value as the Region Code column (EXT_REGION_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied region name instead of the code. If you want to reconfigure the load in this manner, you can load the region code and region name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Chapter 16, "Performing Cross-Module Configuration."

After you load the region code and region name into the IA_CODES table, you can remove the expression in the Source Adapter mapplet that defines the Region Name column. If you leave the Region Name's expression blank, the ADI looks up the Region Name in the IA_CODES table, using the supplied region code when the load occurs. The load mapping then inserts the region name and region code into the data warehouse table.

To configure the Region Name definition

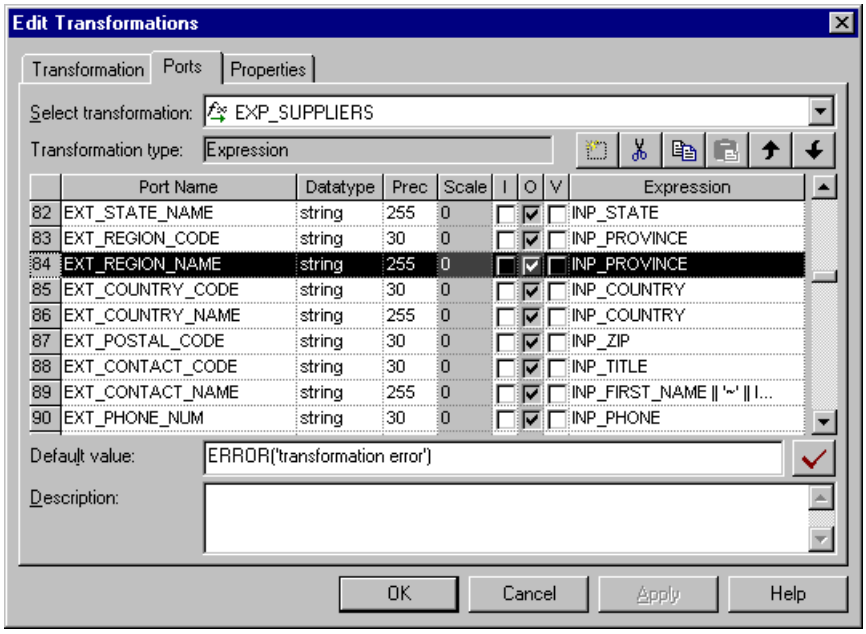
1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 Open the mapplet you want to edit.

The following is a list of all Source Adapter mapplets that use the EXT_REGION_NAME column:

- MPLT_SAI_SUPPLIERS
- MPLT_SAI_BUSN_LOCS_PLANT
- MPLT_SAI_BUSN_LOCS_STORAGE_LOC

3 Double-click the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_REGION_NAME port, as shown in the following figure.



4 Edit the condition by removing the assigned value, if you want the lookup to occur.

5 Click Apply to save changes.

6 Validate the mapplet, and click OK to exit.

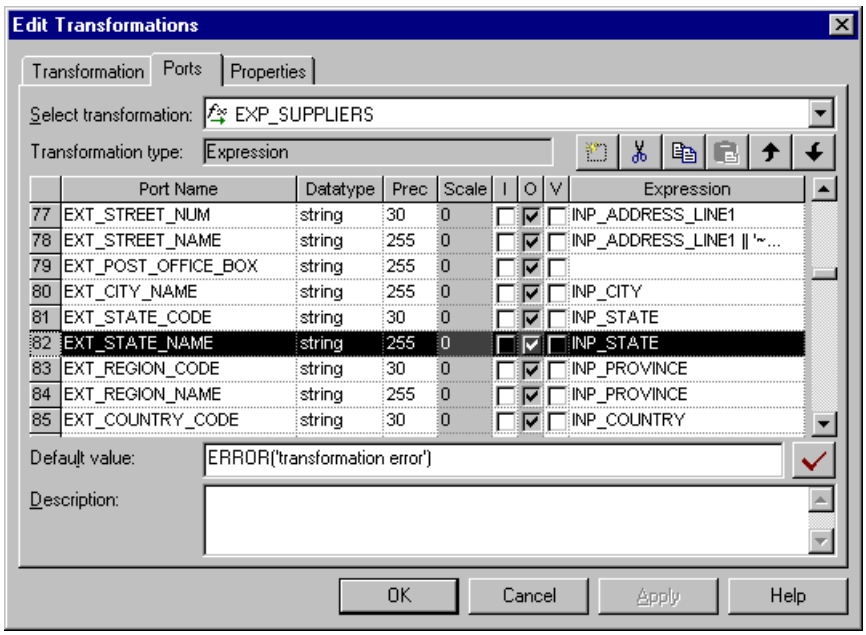
Configuring the State Name Definition

By default, the State Name column (EXT_STATE_NAME) is populated using the same code value as the State Code column (EXT_STATE_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied state name instead of the code. If you want to reconfigure the load in this manner, you can load the state code and state name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Chapter 16, "Performing Cross-Module Configuration."

After you load the state code and state name into the IA_CODES table, you can remove the expression in the Source Adapter mapplet that defines the State Name column. If you set the State Name's expression to null, the ADI looks up the State Name in the IA_CODES table using the supplied state code, when the load occurs. The load mapping then inserts the state name and state code into the data warehouse table.

To configure the State Name definition

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SUPPLIERS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_STATE_NAME port, as shown in the following figure.



4. Edit the condition by removing the assigned value, if you want the lookup to occur.
5. Click Apply to save changes.
6. Validate the mapplet, and click OK to exit.

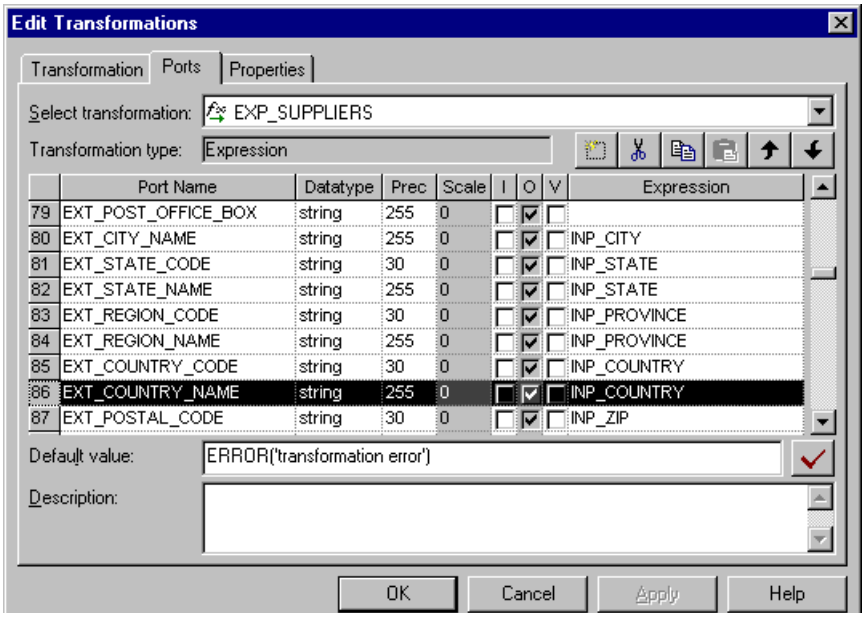
Configuring the Country Name Definition

By default, the Country Name column (EXT_COUNTRY_NAME) is populated using the same code value as the Country Code column (EXT_COUNTRY_CODE). However, you can redefine the load mapping's Source Adapter mapplet to load a source-supplied country name instead of the code. If you want to reconfigure the load in this manner, you can load the country code and country name into the IA_CODES table. For information on loading codes and code names into the IA_CODES table, see Chapter 16, "Performing Cross-Module Configuration."

After you load the country code and country name into the IA_CODES table, you can remove the expression in the Source Adapter mapplet that defines the Country Name column. If you set the Country Name's expression to null, when the load occurs, the ADI looks up the country name in the IA_CODES table, using the supplied country code. The load mapping then inserts the country name and country code into the data warehouse table.

To configure the Country Name definition

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_SUPPLIERS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_COUNTRY_NAME port, as shown in the following figure.



4. Edit the condition by removing the assigned value, if you want the lookup to occur.
5. Click Apply to save changes.
6. Validate the mapplet, and click OK to exit.

Configuring Spend (Oracle 11i)

The following configurations are applicable to the Spend functional area.

Configuring the Make-Buy Indicator

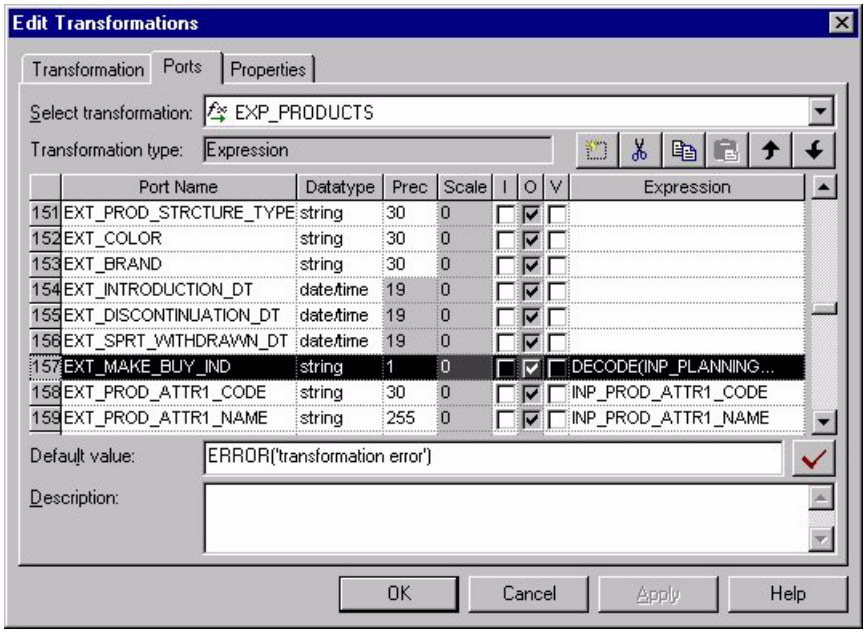
The Make-Buy indicator specifies whether a material that was used to manufacture a product was made in-house or bought from an outside vendor. By default, the indicator is set using the INP_PLANNING_MAKE_BUY_CODE. If the code is set to '1', then the indicator is set to make ('M'). However, if the code is set to '2', then the indicator is set to 'B'. Otherwise, the indicator is set to null.

Your organization may require different indicator codes. If so, you can modify the indicator logic by reconfiguring the condition in the MPLT_SAI_PRODUCTS mapplet. For example, you may want your indicator code to be '0' for make and '1' for buy.

This configuration also applies to the Supplier functional area.

To configure the Make-Buy indicator

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_SAI_PRODUCTS mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_MAKE_BUY_IND port, as shown in the following figure.



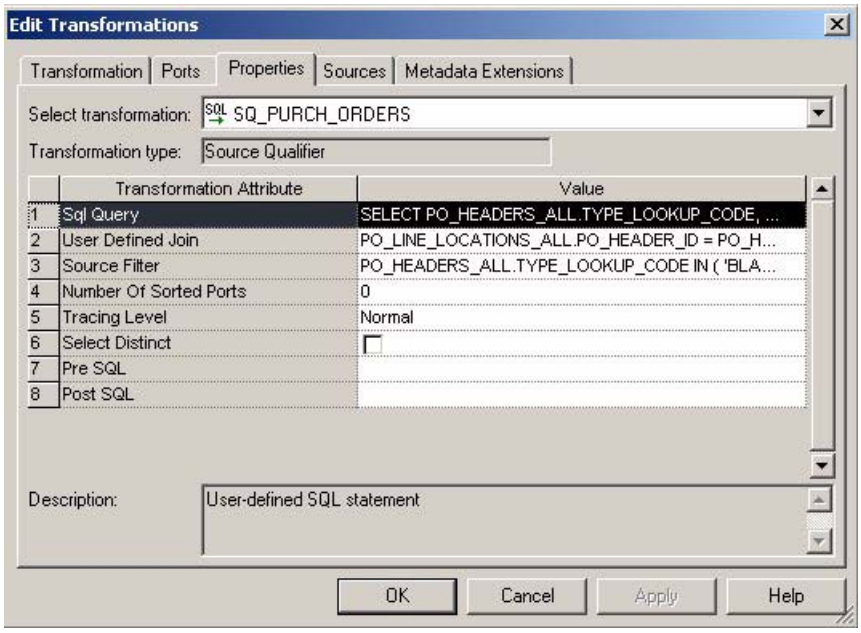
4. Edit the condition by replacing the prepackaged condition with your desired logic.
5. Click Apply to save changes.
6. Validate the mapplet, and click OK to exit.

Extracting Particular Purchase Order Records

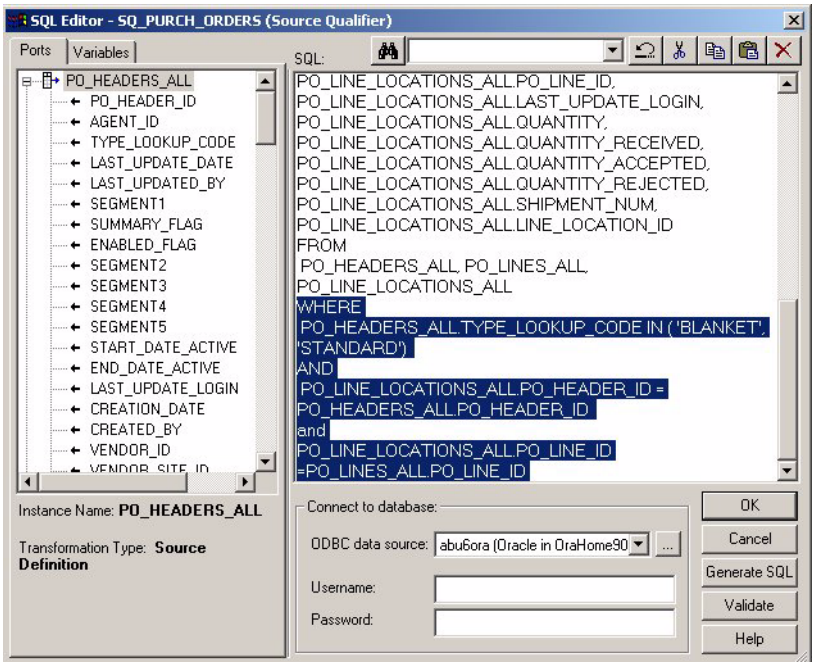
You may not want to extract particular types of records from purchase orders in your source system. In these cases, you can modify the filter condition in the Source Qualifier of the mapplet. By default, the filter condition is set to 'BLANKET' or 'STANDARD'. However, you can change this value to some conditional statement that only allows particular types of records to be extracted.

To extract particular types of purchase order records

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_BCI_PURCH_ORDERS mapplet.
3. Double-click the Source Qualifier to open the Edit Transformations box, and select the Properties tab to display the SQL Query, as shown in the following figure.



4 Double-click the value in the SQL Query to open the SQL Editor box and edit the condition, as shown in the following figure.



5 Replace the prepackaged filter condition with the new filter condition that reflects your business needs.

6 You should only edit the `WHERE` clause of the statement.

7 Click Apply to save the changes, and click OK to exit.

8 Validate the expression, and save your changes to the repository.

Configuring the Purchase Organization Hierarchy

The Purchase Organization hierarchy can contain 10 different levels. Each level is denoted by a number, where '1' is the top of the hierarchy and '10' is the bottom of the hierarchy. By default, the first three levels of the Purchase Organization hierarchy are set as follows:

- `EXT_BORG_HIER1_CODE: Organization ID`
- `EXT_BORG_HIER2_CODE: Legal Entity ID`
- `EXT_BORG_HIER3_CODE: Set of Books ID`

The remainder of the hierarchy ports (`EXT_BORG_HIERX_CODE`) are populated using whatever values are input into the `INP_BORG_HIERX_CODE` ports (X denotes the level within the hierarchy).

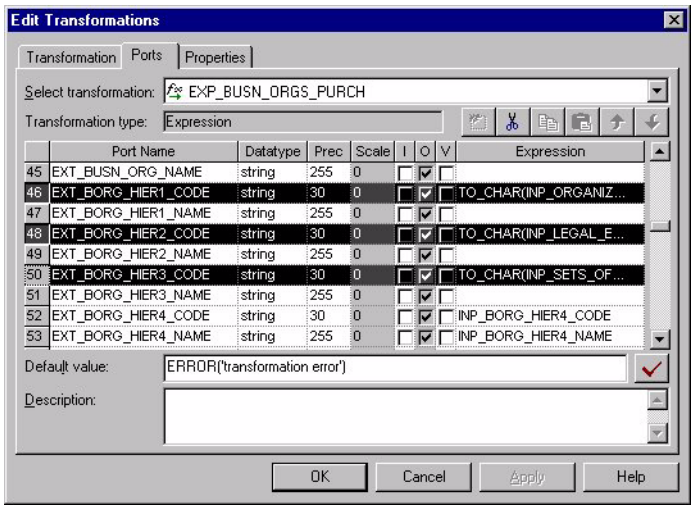
If you want to configure your hierarchy differently than what is prepackaged, you must modify the `EXT_BORG_HIERX_CODE` ports in the Source Adapter mapplet.

To modify the Purchase Organization hierarchy

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 Open the MPLT_SAI_BUSN_ORGS mapplet.

3 Edit the expression for the EXT_BORG_HIERX_CODE port, as shown in the following figure.



4 Validate the expression, and save your changes to the repository.

If applicable, you may need to modify the hierarchy in the front end so that end users can use the hierarchy levels for queries and reports.

Process of Configuring Sourcing for a Universal Source

This section contains Sourcing configuration points that are specific to a universal source. Universal business adapters are built to source data from a flat file interface and populate the data warehouse base tables. This section contains the following information:

Configuring Expenses for a Universal Source

Expenses has one fact table (IA_EXPENSE) and one fact aggregate table (IA_EXPENSE_A1) that support metrics and reports for examining employee expenses. Several mappings populate these tables to complete extracts, loads and updates; these can be configured to suit your organization's business rules. The following sections discuss decisions you must make before you begin adapting individual PowerCenter objects, and provide specific configuration procedures for the universal source.

Universal Source Adapter mapplets extract data from a flat file interface to populate the Siebel Analytics Enterprise Data Warehouse. In this phase of your project, you can configure the following:

- **System Flags and Indicators.** You may configure various system flags to indicate record rejection settings, as well as to indicate if your employees are using your preferred vendors, if you can forward expenses to your customers, and if receipts are available for expensed items.
- **Currency and Payment Options.** You may configure the date used to establish your exchange rates, determine if you allow expenses to be distributed across multiple cost centers, and define payment types in your data warehouse.

Before you begin, you must make the following decisions:

- **Cash Advances.** Cash advance records should have a unique expense item number. If your system allows multiple cash advance records for one expense report, each of these advances must have their own identifiers.
- **Violations.** Many organizations capture violations of company expense policies at the item level (for example, the line item airfare exceeds $2000), cash advance level (for example, cash advance exceeds $500) and at the expense report level (for example, the report's total expenses exceed $5000). Currently the Siebel Analytics Enterprise Data Warehouse stores item level violations within the corresponding item record, but the cash advance record stores both cash advance and report-level violations. Furthermore, each record has a VIOLATION_KEY that can point to IA_REASONS, where violation details are stored. Depending on how you want your analytic system to perform, you must edit your universal business adapter file to reflect the violation counts and keys appropriately. For example:
  - If a requestor violates a cash advance policy, but there are no other violations at the report level, the VIOLATION_ID refers to the cash advance violation only. The violation count equals the cash advance violation counts.
  - If a requestor violates company policy with their expense report, but has not taken a cash advance, you must add a dummy record in the flat file for a cash advance and set the cash advance amount to zero, and enter the violation count as the total number of expense report violations. In this scenario, VIOLATION_ID refers to the expense report violation data only.
  - If a requestor violates a cash advance policy and an expense report policy, you must total the violation counts and enter them in your flat file record, and the VIOLATION_ID has no value. However, if your organization wants to prioritize the violations and have the VIOLATION_ID point to that which is most important, you may point it to the appropriate entry in IA_REASONS.

- **Maintaining Aggregate Information.** If you plan to run the ETL process that populates IA_EXPENSES at a different frequency than the process that populates IA_EXPENSES_A1, you must build additional mappings to retain incremental data in the aggregate table NU_EXPENSES. This is used as a source during post-load processing, required for updating invoice records. See the discussion on setting the S_M_PLP_EXPENSES_INVOICE_UPD_ALT session to run when the incremental invoice load frequency differs for the expense functional area, in Configuring Sourcing for Post-Load Processing on page 182.
- **Deciding to Run Post-Load Processing Mappings.** Depending on your organization's needs and source system setup, you may not need to run the post-load process mappings packaged with Expenses.
  - If your source system or flat file is set up to supply credit invoice details (invoice number, invoice date, posted_on date, and so on.) with all other information, then the required fields are populated with corresponding invoice values. As a result, you need not schedule the following PLP sessions—S_M_PLP_EXPENSES_INVOICE_DERIVE, S_M_PLP_EXPENSES_INVOICE_UPD, and S_M_PLP_EXPENSES_INVOICE_UPD_ALT.
  - If your system or file does not supply invoice details of an expense report, you must use the PLP mappings and sessions as discussed in the previous bullet point. However, you only need to run S_M_PLP_EXPENSES_INVOICE_UPD or S_M_PLP_EXPENSES_INVOICE_UPD_ALT. The alternative mapping is provided if you decide to update your existing expense records in the Siebel Analytics Enterprise Data Warehouse at a different frequency level than you load IA_EXPENSE and IA_EXPENSE_A1.

Configuring the Preferred Merchant Flag for a Universal Source

The Siebel Analytics Enterprise Data Warehouse provides a preferred merchant flag to indicate whether the requestor used a preferred merchant for an expensed item. The flag can have only one value—'Y' (item acquired from a preferred merchant) or 'N' (item acquired from a merchant not recorded). If you use custom logic to determine merchant status, you must include that logic in the expenses Source Adapter.

To configure the preferred merchant flag

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box and select the Port tab.
4. Select the Add Port icon to add the EXT_PREF_MERCHANT_FLAG_OUT port.
5. Enter preferred merchant flag logic.
6. Validate the mapplet and click OK to exit.
7. Save your changes to the repository.

Configuring the Customer Billable Indicator for a Universal Source

The Siebel Analytics Enterprise provides a customer billable indicator that registers whether an expense item can be billed to a customer or should be paid by your organization. The flag can have only one value—'Y' (cost can be passed to the customer) or 'N' (costs paid by your organization). If you use custom logic to determine customer billable status, you must include that logic in the expenses Source Adapter.

To configure the customer billable indicator

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box, and select the Port tab.

   Add a port called EXT_CUST_BILLABLE_IND_OUT = <insert your expression here>.
4. Validate the mapplet and click OK to exit.
5. Save your changes to the repository.

Configuring the Receipts Indicator for a Universal Source

The Siebel Analytics Enterprise provides a receipts indicator that registers whether a requestor has submitted a receipt for a line item in their expense report. The flag can have only one value—'Y' (receipts are available) or 'N' (receipts are not available). If you use custom logic to indicate receipt availability, you must include that logic in the expenses Source Adapter.

To configure the receipts indicator

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box, and select the Port tab.

   Add a port called EXT_RECEIPT_IND_OUT = <insert your expression here>.
4. Validate the mapplet and click OK to exit.
5. Save your changes to the repository.

Configuring Expense Payment Types

The Siebel Analytics Enterprise Data Warehouse supports analysis on three types of payment—Reimbursable Expense, (type E), expenses prepaid by your company, (type P), and cash advance, (type C). All of your organization's payment types must be mapped to one of these types described earlier; do this by modifying MPLT_SAF_EXPENSES.

To configure additional payment types

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_EXP_PAY_TYPE port.
4. Copy and paste this port, rename it EXT_EXP_PAY_TYPE_IND_OUT.
5. Select O to make your new port an output port.
6. Add a decode logic in the expression to decode source-supplied values to the Siebel Analytics Enterprise Data Warehouse payment type of your choice.
7. Validate your mapping, and save your changes to the repository.

Configuring the Default Expense Distribution Percentage for a Universal Source

At times, employee expenses may be distributed across multiple cost centers. For example, technical support associates frequently travel to work in an office with many cost centers; their expenses could be split between those who used their services. This cost center distribution is expected as a percentage from the source system or file; if it is not present a null value is returned. However, this prevents further calculations, so it is preferable to configure the default to be 100% if only one cost center is charged, rather than allow the system to return a null value.

To configure the default expense distribution percentage

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box and select the Port tab.
4. Add a port named EXT_DIST_PERCENTAGE_OUT = <expression that sets this to 100%>.
5. Validate your mapplet.
6. Save your changes.

Configuring Lookup Dates for Currency Conversion for a Universal Source

The Siebel Analytics Enterprise Data Warehouse supports conversion of currency to document (transactional, or source, currency) and group (corporate umbrella currency) for exchange rates. The Siebel Analytics Enterprise uses a specific lookup date to determine the rate of exchange on the date an expense was incurred (ACTUAL_EXP_DT). If you decide to use a different date as your currency conversion lookup, you must use the following procedure.

To configure the exchange rate lookup date

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_XRATE_LKP_DATE port.

   The expression contains the exchange rate lookup date logic.
4. Select the expression in the EXT_XRATE_LOOKUP_DATE port to open the Expression Editor box and edit the expression.
5. Edit the lookup date logic by substituting your lookup date for the prepackaged expression.
6. Validate the mapplet and click OK to exit.
7. Save your changes.

Configuring Domain Values for Expense Type for a Universal Source

The Siebel Analytics Enterprise Data Warehouse prepackages domain values for expense type; these refer to the status of the expense report captured in the source system or flat file (Table 35).

Table 35. Expense Type Domain Values and Descriptions

Domain Value	Description
Composing	Indicates that the requestor is still compiling the report.
Submitted	Indicates that the expense report has been submitted to the source system, but have not yet been approved.
Denied	Indicates the expense report was submitted, reviewed, and denied.
Approved	Indicates the expense report was submitted, reviewed, and approved.
Request for More Information	Indicates the expense report was submitted, reviewed, but the approving manager has requested more information about the expenses incurred.

If you track expense report status types other than those listed in the table, you must map your type value to one of those provided by the Siebel Analytics Enterprise. Do this by editing the expenses Source Adapter.

To configure the expense report status type

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. Open the MPLT_SAF_EXPENSES Source Adapter mapplet.
3. Select the Expression transformation to open the Edit Transformations box, and select the Port tab to display the EXT_EXPENSE_TYPE port.
4. Copy and paste the EXT_EXPENSE_TYPE port and rename it EXT_EXPENSE_TYPE_OUT.
5. Clear the Output check box in the EXT_EXPENSE_TYPE port and make it an input-only port.
6. Clear the Input check box in the EXT_EXPENSE_TYPE_OUT port and make it an output only port.
7. Select the expression in the EXT_EXPENSE_TYPE_OUT port to open the Expression Editor box, and edit the expression by adding your condition.
8. Validate the mapplet and click OK to exit.
9. Link the new port to the EXT_EXPENSE_TYPE port in MAPO_SAF_EXPENSES and save your changes.

Configuring Sourcing for Post-Load Processing

Post-load processing procedures for Expenses allow expense cycle time analysis and update aggregate tables when data captures are performed at different frequencies. Each of these features require some configuration to best suit your organization's needs.

This section contains Sourcing configuration procedures that are specific to post-load processing. The post-load process mappings extract data from an IA table or NU table and process the data through a series of transformations, then load the transformed data into a separate key figure or aggregate table. These mappings are stored in the Configuring Post Load Processing folder in the repository.

Configuring the Extraction of Invoice Details from IA_AP_XACTS for Expense-Related Payments for the Expense Functional Area

The Siebel Analytics Enterprise prepackages post-load processing (PLP) mappings to populate the IA_EXPENSE table with up-to-date invoice information from the IA_AP_XACTS table. The IA_AP_XACTS table updates expense invoice information using the REF_DOC_ITEM and REF_DOC_NUM to uniquely identify the expense document:

```
IA_EXPENSE.INVOICE_DOC_NUM = IA_AP_XACTS.REF_DOC_NUM
```

and

```
IA_EXPENSE.INVOICE_DOC_ITEM = IA_AP_XACTS.REF_DOC_ITEM
```

However, if an expense record can be distinctly identified by values other than the REF_DOC_ITEM and REF_DOC_NUM, you can use those values by adding a condition to the expression in the applicable PLP mapping.

NOTE: Using your values should allow you to narrow the data set.

To configure the Expenses-Accounts Payable reference

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_EXPENSE_UPD post-load processing mapping.
3. Select the Source Qualifier to open the Edit Transformations box, and select the Properties tab to display the User Defined Join.
4. Select the value in the User Defined Join to open the SQL Editor box and edit the expression.
5. Edit the filter condition by adding your desired condition to the prepackaged expression.
6. Open the SQL Query, and then select Generate SQL.
7. Click OK, and then save your changes.

Setting the S_M_PLP_EXPENSES_INVOICE_UPD_ALT Session to Run When the Incremental Invoice Load Frequency Differs for the Expense Functional Area

The Siebel Analytics Enterprise prepackages M_PLP_EXPENSES_INVOICE_UPD to run at the same frequency as the NU_AP_XACTS table load, which is its source. NU_AP_XACTS is truncated before each load of the IA_AP_XACTS table. This truncation can lead to data loss if you decide to run the sessions that update NU_AP_XACTS at different times other than when you execute your post-load update processes.

CAUTION: To prevent data loss resulting from truncation of the NU_AP_XACTS table, you must run an alternate PLP mapping that uses IA_AP_XACTS as a source, as described in this topic. This is because this table always contains a set of invoice data.

Depending on how frequently you want to update data that is already loaded in Siebel Analytics Enterprise Data Warehouse, you must use one of the following tasks:

- If you change the update mapping frequency on a regular basis, you should disable the S_M_PLP_EXPENSES_INVOICE_UPD session and schedule S_M_PLP_EXPENSES_INVOICE_UPD_ALT session to run in its place.
- If you miss a single run of the S_M_PLP_EXPENSES_INVOICE_UPD session, run the S_M_PLP_EXPENSES_INVOICE_UPD session, but do not disable the S_M_PLP_EXPENSES_INVOICE_UPD_ALT session.

The S_M_PLP_EXPENSES_INVOICE_UPD_ALT session can be found in the Sessions directory.

Implementing Temporary Storage When Aggregate Load Frequencies Are Modified for the Expense Functional Area

The Siebel Analytics Enterprise designed IA_EXPENSES and IA_EXPENSES_A1 loads to be synchronized in order to populate NU_EXPENSES at the same time. An incremental snapshot table, NU_EXPENSES is truncated before every load, which can cause data loss for the time period if IA_EXPENSES and IA_EXPENSES_A1 are loaded asynchronously.

CAUTION: To prevent data loss resulting from truncation of the NU_EXPENSES table, the incremental data from NU_EXPENSES must be stored temporarily until it is loaded into the aggregate table. Therefore, you must build an intermediate table and modify M_PLP_EXPENSES_A1_DERIVE to source from it. Your new intermediate table should have the same structure as the NU_EXPENSES table and be set to truncate before loading the IA_EXPENSES_A1.

14 Configuring the Compensation Module

When the Compensation module is installed, you may need to configure certain objects for particular sources to meet your business needs.

This chapter includes the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of Compensation Module

This chapter contains information about configuration for PeopleSoft 7.5 and Oracle 11i for the Compensation module. This module allows you to analyze the salaries, benefits, and rewards that comprise your employee compensation plan. The metrics provided as part of the application allow you to measure several areas of performance and perform a variety of comparative analyses at various levels of granularity.

Within the Compensation module there is one functional area, Payroll, which provides your company with the employee payroll information that can be vital to success in today's economy. Over or under-compensating employees can both have serious effects on your company's ability to maintain a competitive edge. The Compensation module provides the information your Workforce Management department needs to manage compensation costs, such as identifying emerging trends within the organization, or within specific areas of compensation, and evaluating the effectiveness of the level of compensation as an incentive.

For more information about the Compensation module, see the *Siebel Analytics Enterprise Applications User Guide*.

Configuring Compensation for PeopleSoft 7.5

The Compensation module contains the Payroll functional area, which allows you to perform the following tasks: This section provides the necessary information for configuring Payroll for PeopleSoft 7.5.

Configuring Payroll for PeopleSoft 7.5

This section contains configuration instructions for Compensation specific to PeopleSoft 7.5.

Configuring the Pay Detail Flag

The IA_PAYROLL fact table contains payroll transactional information relating to pay stub data. This information includes compensation, deduction, and tax information. The Pay Detail Flag in IA_PAYROLL is an administrative flag that distinguishes pay details from pay totals. For example, total gross, total taxes, total deductions or net pay would be flagged with 'N' to distinguish it as a total, not a detail.

Table 36 lists the mappings and Expression transformations to configure the value of the Pay Detail Flag according to your business rules.

Table 36. Mappings and Expression Transformation for Pay Detail Flag

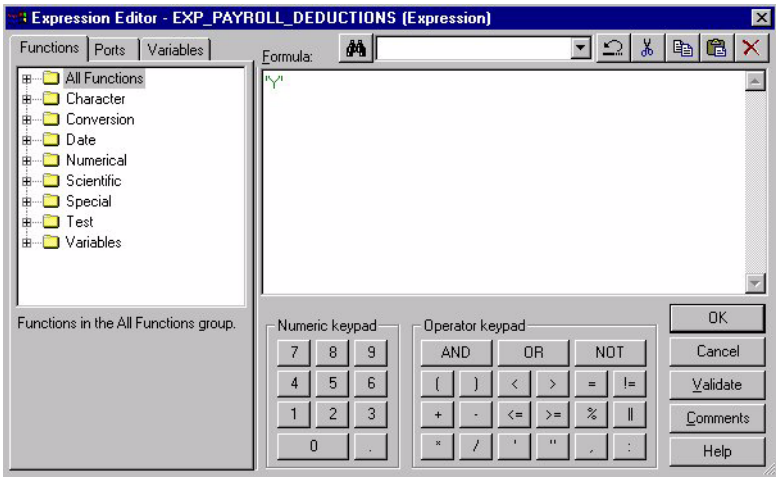
Mapping	Expression Transformation
M_P_PAYROLL_DEDUCTIONS_EXTRACT	EXP_PAYROLL_DEDUCTIONS
M_P_PAYROLL_EARNINGS_EXTRACT	EXP_PAYROLL_EARNINGS
M_P_PAYROLL_EARNINGS_OTH_EXTRACT	EXP_PAYROLL_EARNINGS_OTH
M_P_PAYROLL_TAXES_EXTRACT	EXP_PAYROLL_TAXES

To configure the Pay Detail flag

1. In the PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. Open the mapping.

   Table 36 lists every mapping and Expression transformation that you need to modify to configure the Pay Detail flag.
3. Double-click the Expression transformation to open the Edit Transformations window.

4 Edit the PAY_DETAIL_FLAG port in the Expression Editor to either 'Y' or 'N' for payment detail records based on your business rules, as shown in the following figure.



5 Validate the expression, and save your changes.

Repeat steps as necessary for every mapping listed in Table 36.

Configuring Currency Codes for GRP and LOC for PeopleSoft 7.5

By default, Siebel Analytics Enterprise Data Warehouse sets the group and local currencies to United States dollars. If your group currency, local currency, or both are different from U.S. dollars, then you need to reconfigure these values. For general information about how Siebel Analytics Enterprise Data Warehouse handles currencies, see Working with Document, Local, and Group Currencies on page 246.

If your group or local currencies are different, then you need to modify their definitions for the following tables—Pay Grades, Employee Events, Payroll, and Employee Snapshots. Table 37 states every mapplet that you need to modify if you want to reconfigure the local and group currency definition.

Table 37. Mapplets to Modify When Reconfiguring the Local and Group Currency Definitions

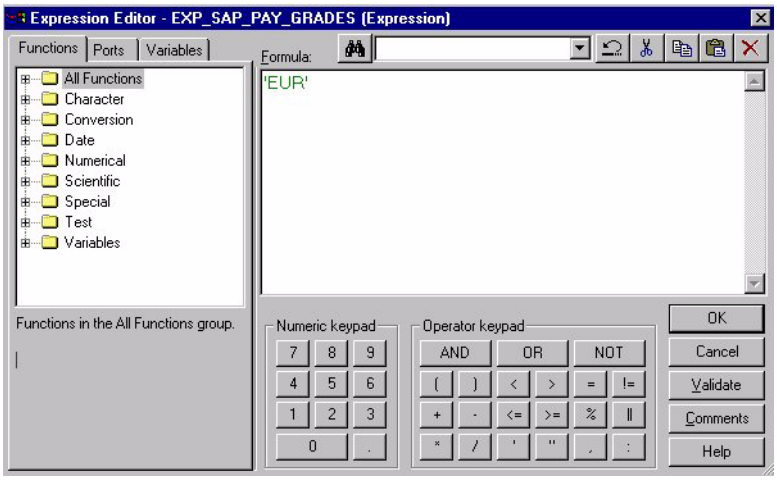
Table	Source Adapter Mapplet	Expression Transformation
Pay Grades	MPLT_SAP_PAY_GRADES	EXP_SAP_PAY_GRADES
Employee Events	MPLT_SAP_EMP_EVENTS	EXP_SAP_EMP_EVENTS
Payroll	MPLT_SAP_PAYROLL	EXP_SAP_PAYROLL
Employee Snapshot	MPLT_SAP_EMP_SNAPSHOT	EXP_SAP_EMP_SNAPSHOT

To configure the currency

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. Open the Source Adapter mapplet.

   Table 37 states every mapplet that you need to modify if you want to reconfigure the local and group currency definitions.
3. Double-click the Expression transformation to open the Edit Transformations box.
4. Edit the applicable Currency Code in the Expression Editor.

   The currency code specified in the transformation must exist as a valid currency code in the table IA_XRATES. The Group Currency Code port is EXT_GRP_CURR_CODE. The Local Currency Code port is EXT_LOC_CURR_CODE. Edit the expression to redefine the currency, as shown in the following figure.



5. Validate the new expression.
6. Repeat these steps for every mapplet listed in Table 37.
7. After you configure currencies for all necessary mapplet, save your changes to the repository.

Configuring Exchange Rate Types for PeopleSoft 7.5

In the Lookup transformation LKP_XRATES, Compensation sessions are set with the value 'MODEL:0:OFFIC' for XRATE_TYPE_CODE. This value can be configured according to your business requirements. These values are a concatenation of the columns RT_RATE_INDEX, TERM and RT_TYPE from the PeopleSoft table PS_RT_RATE_TBL.

You can configure this Lookup transformation in the following sessions:

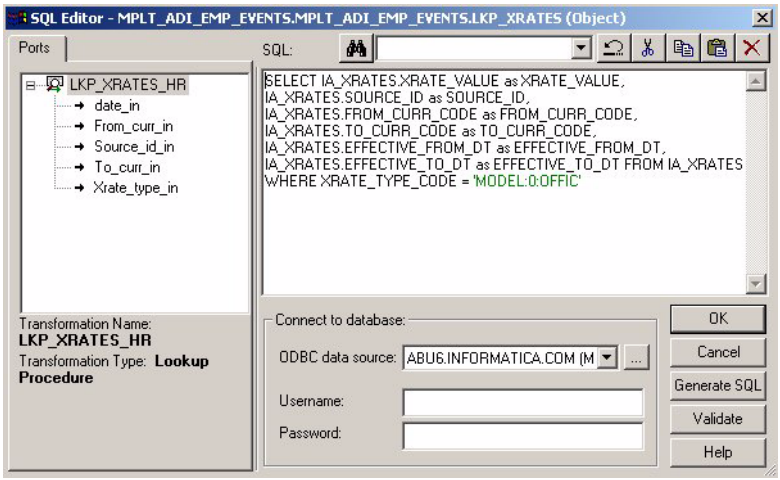
- S_M_P_EMP_EVENTS_LOAD_P1
- S_M_P_EMP_SNAPSHOT_LOAD_P2
- S_M_P_PAY_GRADES_LOAD_P1

- S_M_P_PAYROLL_A1_DEDUCTIONS_LOAD_P1
- S_M_P_PAYROLL_A1_EARNINGS_LOAD_P2
- S_M_P_PAYROLL_A1_TAXES_LOAD_P3
- S_M_P_PAYROLL_DEDUCTIONS_LOAD_P1
- S_M_P_PAYROLL_EARNINGS_LOAD_P2
- S_M_P_PAYROLL_TAXES_LOAD_P3
- S_M_P_PAYROLL_TOTAL_LOAD_P4

To configure sessions for the Exchange Rate Type code

1. In PowerCenter Workflow Manager, open the session you want to edit.
2. Open the session to open the Edit Tasks box.
3. In the Transformation tab, locate the lookup entry for LKP_XRATES prefixed with the corresponding ADI mapplet name.

   Scroll right to open the SQL override, as shown in the following figure.



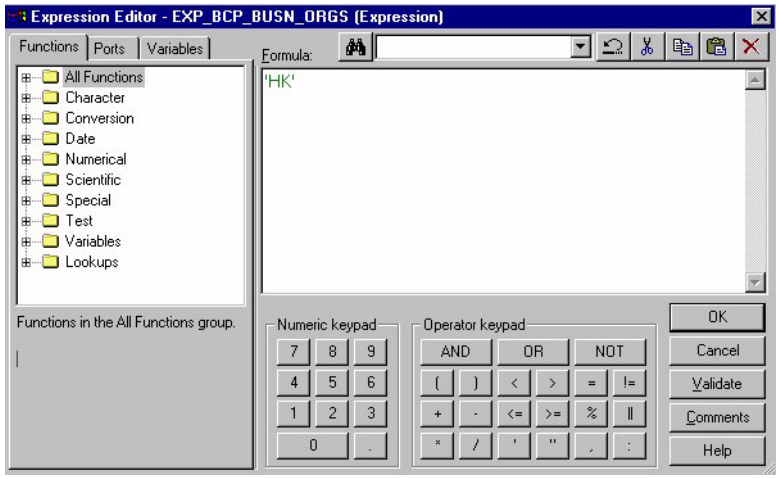
4. Configure XRATE_TYPE_CODE according to your business requirements.
5. Click OK in the SQL Editor.
6. Click OK in the session dialog box.

Configuring Business Organizations for PeopleSoft 7.5

The PeopleSoft Source Adapter has default values for the Business Organizations table. One of the defaults is for the V_SETID column. The V_SETID column determines at what segment of the entire organization you want to look. For example, you may segment your entire company by country. In this case, one value for the V_SETID column is USA to denote your business in the United States of America. Please note, the value of this column depends on how your company wishes to segment and define its organizations. By default, the V_SETID port in the Business Component mapplet MPLT_BCP_EMP_BUSN_ORGS is set to USA. The V_SETID should be configured to source the business organizations for a specific segment of your corporation. It corresponds to the segmentation used in PS_JOB.SETID_DEPT.

To configure the V_SETID for Business Organizations

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. In the MPLT_BCP_EMP_BUSN_ORGS, open the EXP_BCP_BUSN_ORGS Expression transformation.
3. Edit the V_SETID port in the Expression Editor according to your business rules, as shown in the following figure.



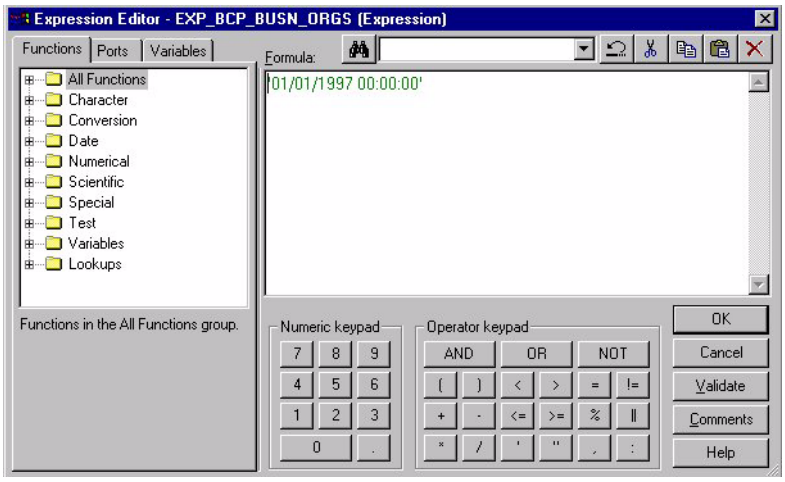
4. Validate and save your changes to the repository.

   The V_EFFDT port in the Business Component mapplet MPLT_BCP_EMP_BUSN_ORGS is set to '01/01/1900 00:00:00' using the four-digit year code.

To configure the V_EFFDT for Business Organizations

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. In the MPLT_BCP_EMP_BUSN_ORGS, open the EXP_BCP_BUSN_ORGS Expression transformation.

3 Edit the V_EFFDT port in the Expression Editor according to your business rules, as shown in the following figure.



4 Validate and save your changes to the repository.

The organizational structure in the PeopleSoft source table DEPT_SECURITY has various hierarchies based on the effective dates and SETIDs (regions). If you want to configure the hierarchy for a particular date and for a particular SETID, you need to create a new set of Business Components. For more information on creating and modifying Business Adapters, see Creating and Modifying Business Adapters on page 373. After they have been created, you can configure these Business Components as necessary for the desired values.

Configuring the Pay Types Table Flags for PeopleSoft 7.5

The IA_PAY_TYPES table defines the types of compensation and deductions for all types of pay stub-related entries. This definition includes information such as expenses, bonuses, taxes, deductions, and salary.

The IA_PAY_TYPES table contains several flags that can be configured to meet your business rules.

- **Pay Type flag.** See Configuring the Pay Type Flag PeopleSoft 7.5 on page 193 for further description and procedures and Figure 12.
- **Compensation flag.** See Configuring the Compensation Flag PeopleSoft 7.5 on page 194 for further description and procedures and Figure 12.
- **Taxable flag.** See Configuring the Taxable Flag PeopleSoft 7.5 on page 196 for further description and procedures and Figure 13.

- **Pension Compensation flag.** See Configuring the Pension Compensation Flag PeopleSoft 7.5 on page 196 for further description and procedures and Figure 13.

for further description

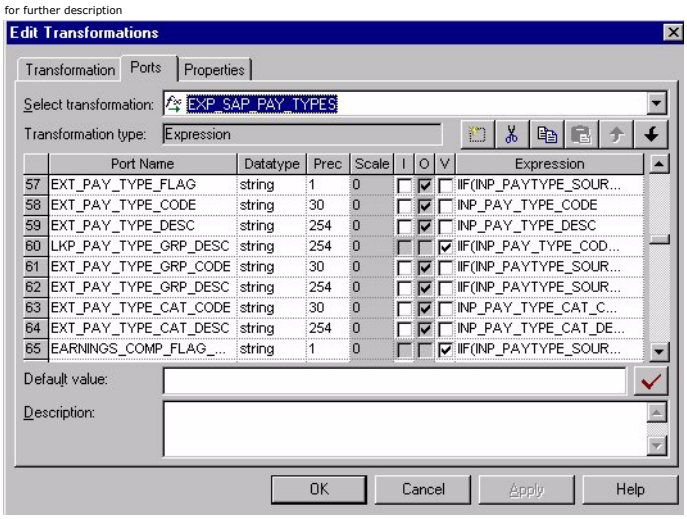


Figure 12. PeopleSoft 7.5: Ports for Pay Type and Compensation Flags

Pension Compensation Flag

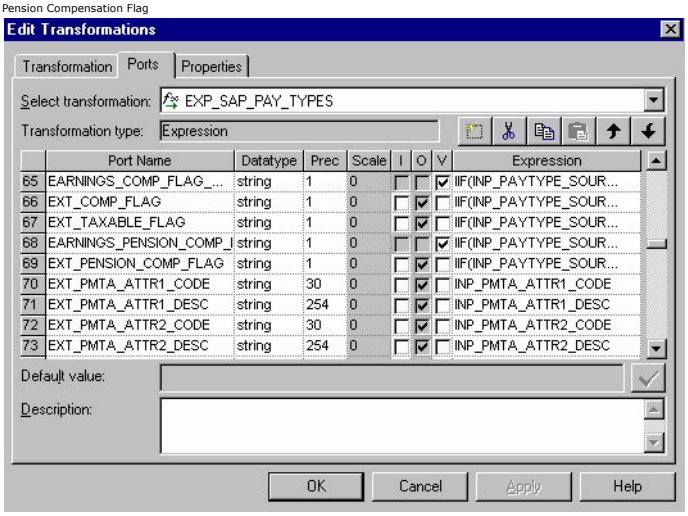


Figure 13. PeopleSoft 7.5: Ports for Taxable and Pension Compensation Flags

Each of these flags is set in the MPLT_SAP_PAY_TYPES mapplet. Configuration for each of the flags is described in the following sections.

Configuring the Pay Type Flag PeopleSoft 7.5

The Pay Type flag indicates whether a particular amount is a compensation or a deduction. Pay Type Flags should be set for amounts such as net compensation, net deductions, gross compensation, and gross deductions.

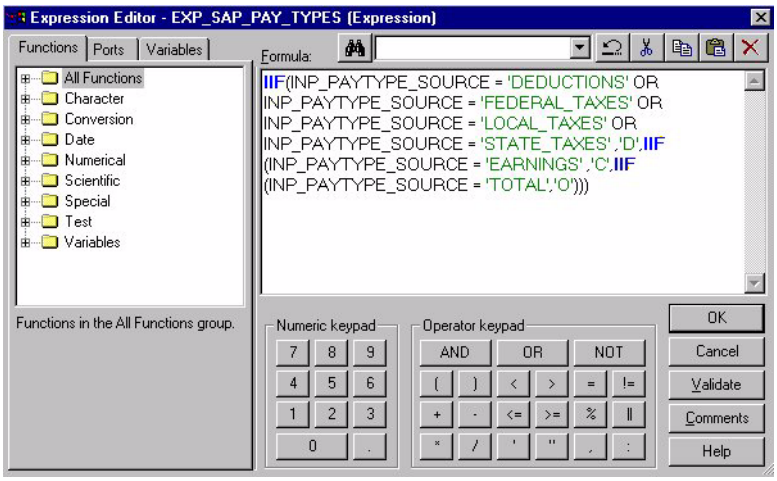
The PeopleSoft Source Adapter is preconfigured with the following logic for Pay Type Flags:

- DEDUCTIONS 'D'
- FEDERAL_TAXES 'D'
- LOCAL_TAXES 'D'
- STATE_TAXES 'D'
- EARNINGS 'C'
- TOTAL 'O'

Using the mapplet MPLT_SAP_PAY_TYPES, you can configure PeopleSoft Source Adapter to determine the Pay Type Flags.

To configure the Pay Type flag

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. In the MPLT_SAP_PAY_TYPES, open the EXP_SAP_PAY_TYPES Expression transformation.
3. Edit the EXT_PAY_TYPE_FLAG port in the Expression Editor to handle the Pay Type flag to meet your business rules, as shown in the following figure.



4. Validate the expression, and save your changes to the repository.

Configuring the Compensation Flag PeopleSoft 7.5

The Compensation Flag indicates whether the payment type is part of compensation. The Compensation module is preconfigured with the Compensation Flag setting shown in Table 36.

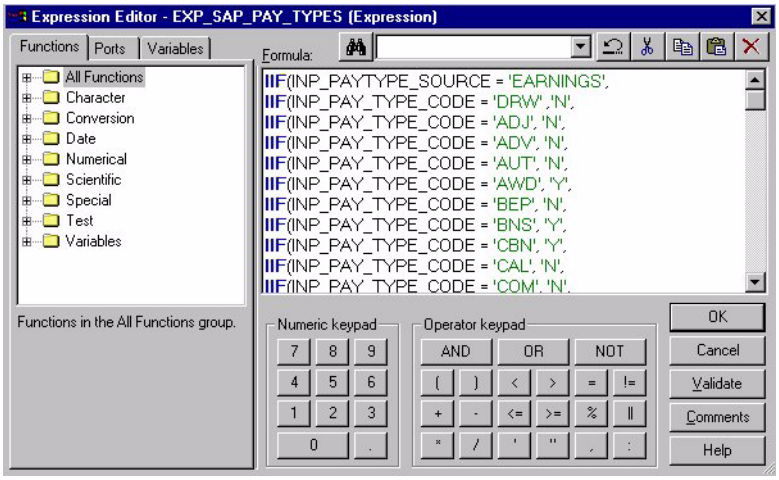
Table 38. Compensation Flag Settings

Payment Type	Subtype	Setting	Comment
Deductions		N	These flags are set in the COMP_FLAG port.
Federal Taxes		N	
Local Taxes		N	
State Taxes		N	
Total		N	
Earnings	Allowances	N	
	Bonuses (with the exception of profit-related pay)	Y	These flags are set in the EARNING_COMP_FLAG_VAR port. The EARNING_COMP_FLAG_VAR port passes the value to the COMP_FLAG port.
	Commission	N	
	Expatriate/ Relocation	N	
	Expenses	N	
	Insurance Adjustments	N	
	Miscellaneous	N	
	Overtime (with the exception of night pay)	N	
	Paid Time	N	
	Payoff Time	N	
	Regular Pay	Y	
	Retro	N	
	Severance	N	
	Training Pay	N	
	Unpaid Time	N	

Using the mapplet MPLT_SAP_PAY_TYPES, you can configure Compensation to define the Compensation Flag, as described in the following procedure.

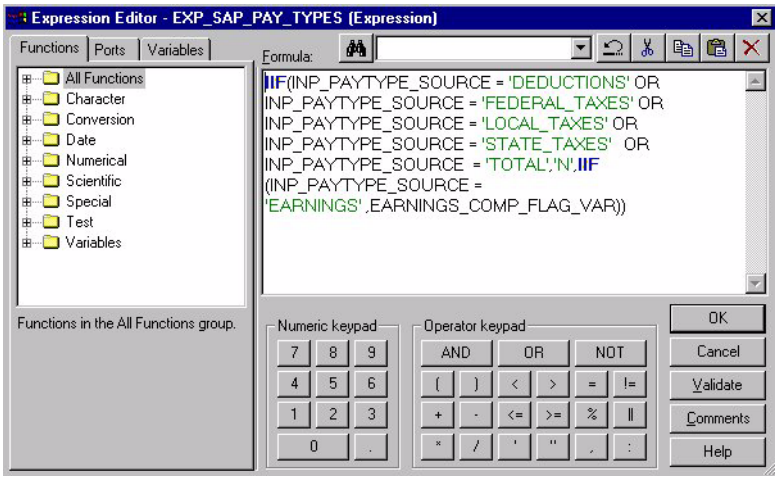
To configure the Compensation flag

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. In the MPLT_SAP_PAY_TYPES, open the EXP_SAP_PAY_TYPES Expression transformation.
3. Edit the EARNINGS_COMP_FLAG_VAR port in the Expression Editor to handle the Compensation Flag for earnings, as shown in the following figure.



The value that is set for each earning in the EARNINGS_COMP_FLAG_VAR port is the value that is passed on to the EXT_COMP_FLAG port.

4. Validate the new expression.
5. Edit the expression in the EXT_COMP_FLAG port to handle the Compensation Flag for deductions, totals, and taxes, as shown in the following figure.



6 Validate the expression and save your changes to the repository.

Configuring the Taxable Flag PeopleSoft 7.5

The Taxable Flag indicates whether the compensation is taxable. PeopleSoft Source Adapter is preconfigured with the following logic for Taxable Flags:

- DEDUCTIONS, FEDERAL_TAXES, LOCAL_TAXES, STATE_TAXES and TOTAL are set as 'N'.
- EARNINGS are derived from the source.

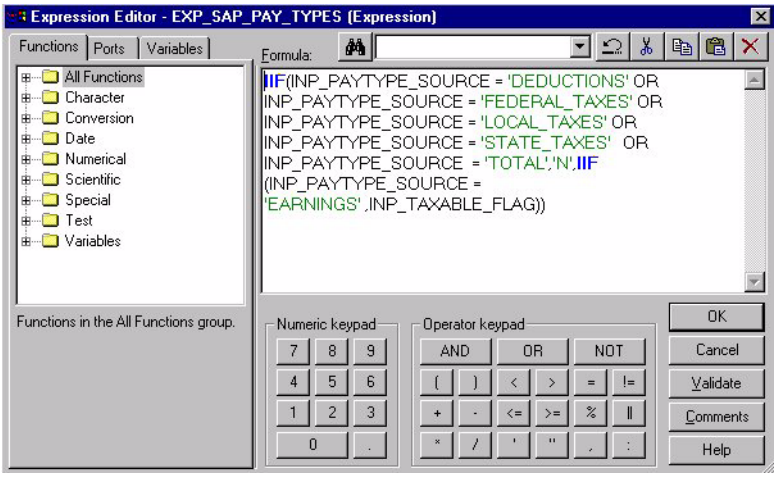
Using the mapplet MPLT_SAP_PAY_TYPES, you can configure PeopleSoft Source Adapter to determine the Taxable Flag.

To configure the Taxable flag

1 In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.

2 In the MPLT_SAP_PAY_TYPES, open the EXP_SAP_PAY_TYPES Expression transformation.

3 Edit the EXT_TAXABLE_FLAG port in the Expression Editor to handle the Taxable flag to meet your business rules, as shown in the following figure.



4 Validate the expression and save your changes to the repository.

Configuring the Pension Compensation Flag PeopleSoft 7.5

The Pension Compensation Flag indicates whether the compensation is considered for pension purposes. PeopleSoft Source Adapter is preconfigured with the following logic for Pension Compensation Flag.

Table 39. Pension Compensation Flag Settings for the PENSION_COMP_FLAG Port

Payment Type	Setting
Deductions	N
Federal Taxes	N
Local Taxes	N
State Taxes	N
Total	N

Table 40. Pension Compensation Flag Settings for the EARNINGS_PENSION_COMP_FLAG_VAR Port

Earnings Payment Subtype	Setting
Accumulative Draw	N
Adjustments	Y
Advance	N
Automobile Allowance	N
Beeper Pay	N
Bonus	N
Car Allowance	N
Commission Test Change	N
Comp Time	N
Cost of Living Adjustment	N
Dental Credits	N
Double Time	N
Expense Reimbursement	N
Extra pay above regular	N
Family Medical Leave	N
Family Medical Leave Adjustment	N
Flex Credits	N
Float Pay	N
Float Pay (Emergency Room)	N
Float Pay (Operating Room)	N
Foreign Housing (Rent)	N
Foreign Housing (Utilities)	N

Earnings Payment Subtype	Setting
Goods and Services Differential	N
Health Credit	N
Holiday (Non-Statutory)	N
Holiday (Statutory)	N
Holiday Bonuses	Y
Hours Only (no pay)	N
Interim Living (Host)	N
Jury Pay	N
Legislated Vacation Pay	Y
Life Insurance Credits	N
Long Term Disability	N
Mileage Reimbursement	N
Night Pay	N
Non Cash Award	N
On-Call Earnings	N
On-Call Pay	Y
Orientation Pay	N
Overtime	Y
Overtime Pay	Y
Per Diem Pay	Y
Premium Pay at 0.5	N
Profit Related Pay	N
Regular	N
Reimbursement - Tuition Costs	N
Relocation (Non-Taxable)	N
Relocation (Taxable)	N
Relocation Reimbursement	N
Retro - Negative Offset	N
Retro Pay/Earnings	N
Retro Pay-Premium/Extra	N
Retro/Deduction Refund	N
Retroactive Earnings - TRG, Severance Pay	N

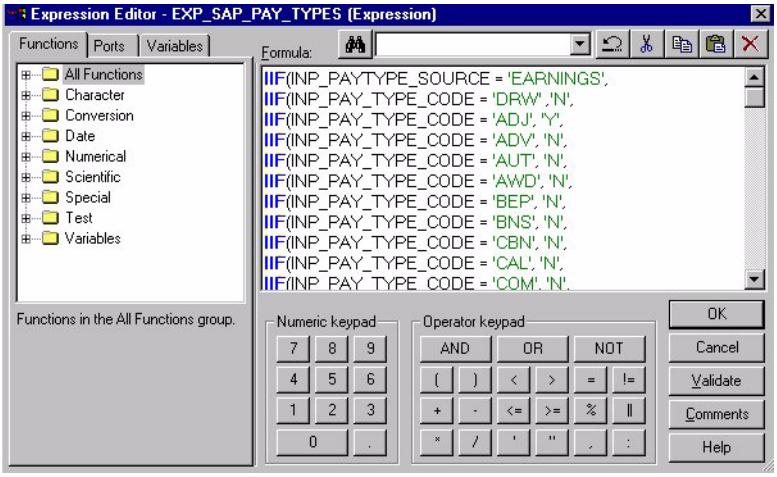
Earnings Payment Subtype	Setting
Salary	N
Shift 2 Differential	N
Short Term Disability	N
Short-Term Disability at 60%	N
Sick Pay	N
Sick Payoff	N
Special Bonus	N
Stock - Ordinary Income	N
Stock-Tax Payment	N
Straight Overtime	N
Tip Adjustments	Y
Tip Allocations	Y
Tip Credits	Y
Tips Reported	Y
Top-up Earnings	N
Unemployment Insurance	N
Uniform Reimbursement	N
Unpaid Leave	N
Vacation Advance	N
Vacation Buy	N
Vacation Pay	Y
Vacation Payoff	N
Vacation Sell	N
Visits Pay (A.M.)	N
Visits Pay (P.M.)	N
Weekend Pay	N
Worker’s Compensation	Y

Using the mapplet MPLT_SAP_PAY_TYPES, you can configure the PeopleSoft Source Adapter to determine the Pension Compensation Flag, as described in the following procedure.

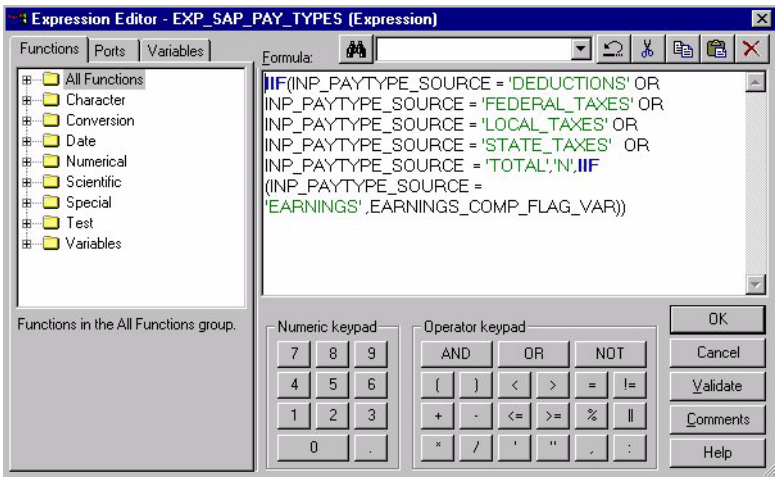
To configure the Pension Compensation flag

1. In PowerCenter Designer, open the Configuration for PeopleSoft 7.5 folder.
2. In the MPLT_SAP_PAY_TYPES, open the EXP_SAP_PAY_TYPES Expression transformation.
3. Edit the EXT_PENSION_COMP_FLAG_VAR port in the Expression Editor to handle the Pension Compensation flag for earnings.

   The value that is set for each earning in this port is the value that is passed on to the EXT_PENSION_COMP_FLAG port, as shown in the following figure.



4. Validate the new expression.
5. Now, edit the EXT_PENSION_COMP_FLAG port. Edit the expression to handle the Pension Compensation Flag for deductions, totals, and taxes, as shown in the following figure.



6. Validate the expression and save your changes to the repository.

Configuring Compensation for Oracle 11i

This section contains configuration instructions for compensation specific to Oracle 11i.

There is only one configuration task for Payroll for Oracle 11i, which is modifying the payroll aggregate method. This section describes the necessary procedure necessary for this configuration.

Modifying the Payroll Aggregate Method

By default, the $$PAYROLL_AGG_METHOD is set to 'AGGREGATE' for Oracle 11i. When the setting is 'AGGREGATE', the payroll information is incrementally aggregated to the corresponding grain level. When it is 'UPDATE', the payroll information is fully refreshed for the corresponding period. If 'UPDATE' is used, make sure that the corresponding staging table contains full data for the corresponding period. If you need to modify this definition to 'UPDATE', you can do so by modifying the FILE_PARAMETERS_ORA11i.CSV file. This file contains the default values for the sessions described in Table 41.

Table 41. Sessions with 'AGGREGATE' as the Default Payroll Aggregate Method

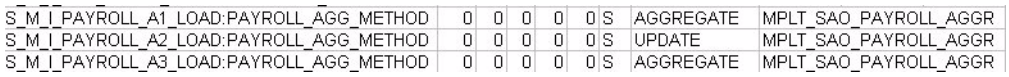
Session	Frequency
S_M_I_PAYROLL_A1_LOAD	Monthly
S_M_I_PAYROLL_A2_LOAD	Quarterly
S_M_I_PAYROLL_A3_LOAD	Yearly

After you have modified the file, the next time you run the session, the new Payroll Aggregate method is populated with updated data.

To redefine the Payroll Aggregate method

1. Open the FILE_PARAMETERS_ORA11i.CSV file in the installation directory in `$pmserver\srcfiles`.
2. Replace the default Payroll Aggregate method with the new Payroll Aggregate method, as shown in the following figure.

   Perform this action for applicable sessions, as shown in the following figure.



S_M_I_PAYROLL_A1_LOAD:PAYROLL_AGG_METHOD	0	0	0	0	0	S	AGGREGATE	MPLT_SAO_PAYROLL_AGGR
S_M_I_PAYROLL_A2_LOAD:PAYROLL_AGG_METHOD	0	0	0	0	0	S	UPDATE	MPLT_SAO_PAYROLL_AGGR
S_M_I_PAYROLL_A3_LOAD:PAYROLL_AGG_METHOD	0	0	0	0	0	S	AGGREGATE	MPLT_SAO_PAYROLL_AGGR

3. Save and close the file.

15 Configuring the Workforce Management Operations and Workforce Management Retention Module

When the Workforce Management Operations and Workforce Management Retention module is installed, you may want to configure certain objects for particular sources to meet your business needs.

This chapter contains the following topics:

For additional configuration information that applies to this module, see the following chapters of this guide:

- Chapter 16, "Performing Cross-Module Configuration"
- Chapter 17, "Storing, Extracting, and Loading Additional Data"
- Chapter 18, "Integrating Additional Data"

Overview of the Workforce Management Operations and Workforce Management Retention Module

This chapter contains information about configuration for PeopleSoft 7.5 and Oracle 11i for the Workforce Management Operations and Workforce Management Retention module, which allows you to analyze your personnel profiles and the events that constitute changes in personnel status. The information stored in this module allows you to measure several areas of performance, as well as comply with government requirements in regard to equal employment opportunity and reporting.

The Workforce Management Operations and Workforce Management Retention module has three different functional areas:

- Retention
- U.S. Statutory Compliance
- Workforce Profile

The three functional areas of this module address sensitive and nonsensitive information about your employees, outside of compensation, such as employee and benefits-related costs.

Under the *Retention* functional area you can find the events that are the hallmarks of employees' professional life cycle. These events include their hiring information, their promotional opportunities realized and not realized, the quality of the employees' job performance as measured by performance ranking, their length of service, and the reasons for termination, both voluntary and involuntary. Monitoring retention rates within departments is useful in determining potential problem areas that may want to be addressed by senior management.

The *U.S. Statutory Compliance* functional area stores information that help Human Resources departments prepare government-required reports.

The *Workforce Profile* functional area provides you with the tools to separate sensitive from nonsensitive information, and to restrict access to sensitive data. Sensitive information includes such data as ethnicity, age, native language, marital status, and performance ratings. Nonsensitive information includes information such as job title, work location, and position status. For more information about the Workforce Management Operations and Workforce Management Retention module, see the *Siebel Analytics Enterprise Applications User Guide*.

Configuring Workforce Management Operations and Workforce Management Retention for Oracle 11i

This section contains Workforce Management Operations and Workforce Management Retention configuration information that is specific to Oracle 11i. To configure Workforce Management Operations and Workforce Management Retention for Oracle 11i, perform the tasks in the following sections:

- Configuring U.S. Statutory Compliance on page 204
- Configuring Workforce Profile for Oracle 11i on page 219

NOTE: Currently, no configuration changes are required for Oracle 11i for the Retention functional area.

Configuring U.S. Statutory Compliance

There are two dimensions in the configuration for the Oracle 11i folder for which there are mandatory changes to the configuration information—the Employees dimension and the Jobs dimension. Failure to configure the following mandatory configuration information can result in incorrect data being fed to reports and other ETL processes.

NOTE: If you want to retain values from the source system or previously existing values that are not included in the domain values, enter an Else statement in the expression for the code.

For example, if you want to retain a 'None' option as an ethnic group code, enter the expression as shown in Figure 14.

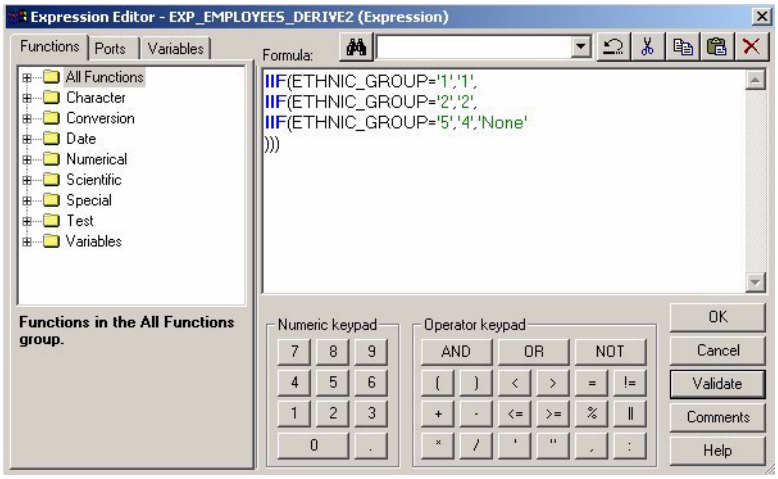


Figure 14. Oracle 11i: Modified Code with Else Statement

For more information on using domain values, see Working with Domain Values on page 268. For a list of domain values, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

Configuring the Employees Dimension

Within the Employees dimension there are mandatory changes to the configuration information in the Expression transformation EXP_EMPLOYEES for the mapping M_I_EMPLOYEES_EXTRACT.

The configuration information includes the domain values for the following:

- Ethnic Group Code (ETHN_GRP_CODE_OUT)
- Ethnic Group Description (ETHN_GRP_DESC_OUT)
- Veteran Status Code (VETERAN_STAT_CODE_OUT)
- Veteran Status Description (VETERAN_STAT_DESC_OUT)

Configuring Ethnic Group Codes

The domain values for Ethnic Group Code and Ethnic Group Description in Siebel Analytics Enterprise Data Warehouse are shown in Table 42.

Table 42. Domain Values for Ethnic Group Code and Ethnic Group Description

Ethnic Group Code	Ethnic Group Description
1	White
2	Black
3	Asian

Table 42. Domain Values for Ethnic Group Code and Ethnic Group Description

Ethnic Group Code	Ethnic Group Description
4	American Indian/Alaskan Native
5	Native Hawaiian or Other Pacific Islander
6	Hispanic or Latino (White)
7	Hispanic or Latino (All Other Races)
8	Race Unknown
9	Others

For each of these ports' expressions, it is necessary that the source-supplied values are mapped to the expected domain values so that correct ethnic group and veteran status information is supplied. For example, if the source-supplied values were as shown in Table 43, there would be a discrepancy between the domain values in Siebel Analytics Enterprise Data Warehouse and source-supplied value for Ethnic Group Code 1, 2, and 4.

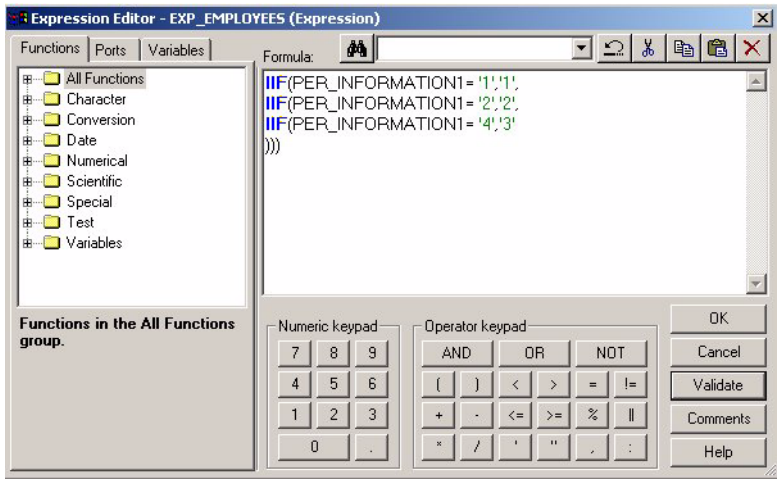
Table 43. Sample Source Values for Ethnic Group Code and Ethnic Group Description

Ethnic Group Code	Ethnic Group Description
1	Caucasian
2	African American
4	Asian

To modify the Ethnic Group code

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In the M_I_EMPLOYEES_EXTRACT mapping, double-click the EXP_EMPLOYEES Expression transformation.

3 Modify the ETHN_GRP_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



4 Validate and save your changes to the repository.

Configuring Ethnic Group Descriptions

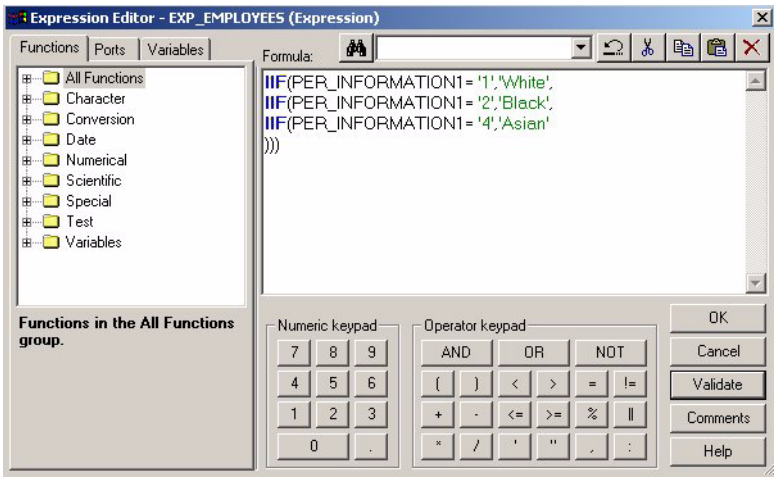
For the Ethnic Group Description, make sure that the description used with the source-specified code matches the description defined in the domain values. For example, if the source system uses the description, African American, instead of Black for code 2, the description must be modified to reflect the descriptive value of Black.

To modify the Ethnic Group description

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In the M_I_EMPLOYEES_EXTRACT mapping, double-click the EXP_EMPLOYEES Expression transformation.

3 Modify the ETHN_GRP_DESC_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



4 Validate and save your changes to the repository.

Configuring Veteran Status Codes

The steps to resolve the Veteran Status Code and the Veteran Status Description requirements are very similar to those for Ethnic Group. The domain values for Veteran Status Code and Veteran Status Description are shown in Table 44.

Table 44. Domain Value for Veteran Status Code and Veteran Status Description

Veteran Status Code	Veteran Status Description
1	Special Disabled Veterans
2	Vietnam Era Veterans
3	Other Protected Veterans
4	Newly Separated Veterans

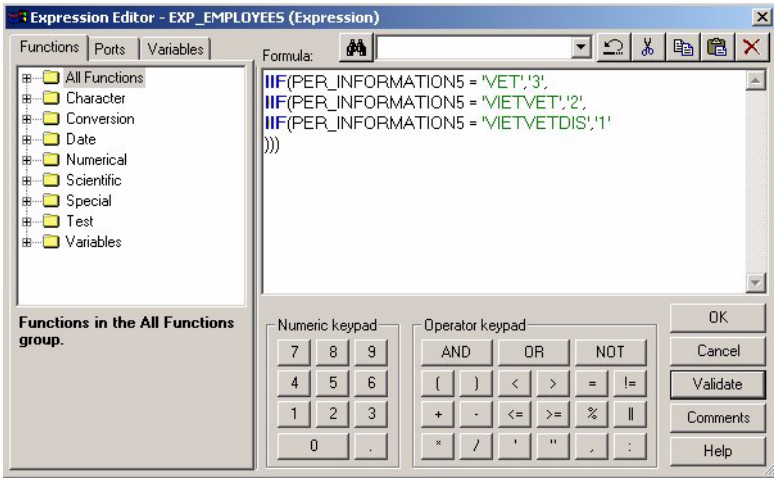
For each of these ports' expressions, it is necessary that the source-supplied values are mapped to the expected domain values so that the correct veteran status information is supplied. For example, if the source-supplied values for Veteran Status Code and Veteran Status Description were as shown in Table 45, there would be a discrepancy between the domain values and source-supplied values for all codes.

Table 45. Sample Source-Supplied Veteran Status Code and Veteran Status Description

Veteran Status Code	Veteran Status Description
VET	Other Veterans
VIETVET	Vietnam Veterans
VIETVETDIS	Disabled Vietnam Veteran

To modify Veteran Status code

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In the M_I_EMPLOYEES_EXTRACT mapping, double-click the EXP_EMPLOYEES Expression transformation.
3. Modify the VETERAN_STAT_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



4. Validate and save your changes to the repository.

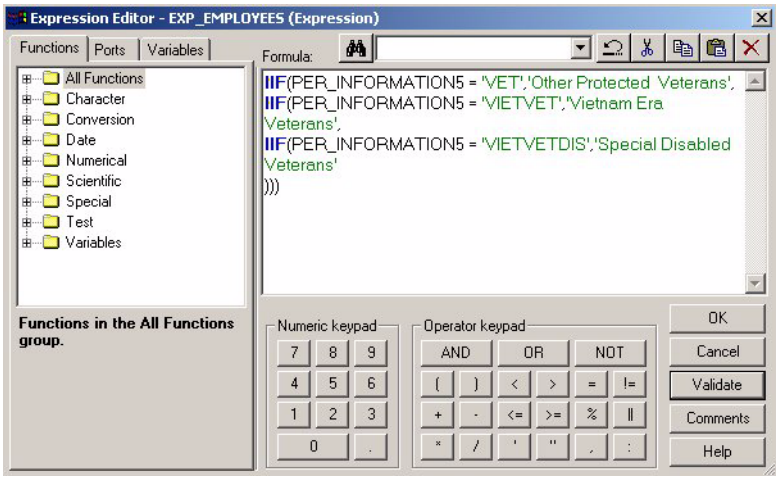
Configuring Veteran Status Descriptions

As with the Ethnic Group Description, make sure that the Veteran Status Description used with the source-specified code matches the description defined in the domain values. For example, in the sample source data in Table 45, the source system uses the description Other Veterans instead of Other Protected Veterans for source-supplied code VET.

To modify Veteran Status description

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In the M_I_EMPLOYEES_EXTRACT mapping, double-click the EXP_EMPLOYEES Expression transformation.

3 Modify the VETERAN_STAT_DESC_OUT port's expression by mapping the source-supplied values to the domain values, as shown the following figure.



4 Validate and save your changes to the repository.

Configuring the Jobs Dimension

Within the Jobs dimension there are two mandatory changes to the configuration information in the Expression transformation EXP_JOBS for the mapping M_I_JOBS_EXTRACT. The configuration information includes the domain values for the following:

- EEO Job Category Code (EEO_JOB_CAT_CODE_OUT)
- EEO Job Category Description (EEO_JOB_CAT_DESC_OUT)

Configuring EEO Job Category Codes

The domain values for EEO Job Category Code and EEO Job Category Description in Siebel Analytics Enterprise Data Warehouse are shown in Table 46.

Table 46. Domain Values for EEO Job Category Code and EEO Job Category Description

EEO Job Category Code	EEO Job Category Description
1	Officials and Managers
2	Professionals
3	Technicians
4	Sales Workers
5	Office and Clerical
6	Craft Workers
7	Operatives

Table 46. Domain Values for EEO Job Category Code and EEO Job Category Description

EEO Job Category Code	EEO Job Category Description
8	Laborers
9	Service Workers

For each of these ports' expressions, it is necessary that the source-supplied values are mapped to the expected domain values so that the correct EEO job category information is supplied. For example, if the source-supplied values were as shown in Table 47 there would be a discrepancy between the domain values and source-supplied values for EEO Job Categories 1, 2, 5, and 8.

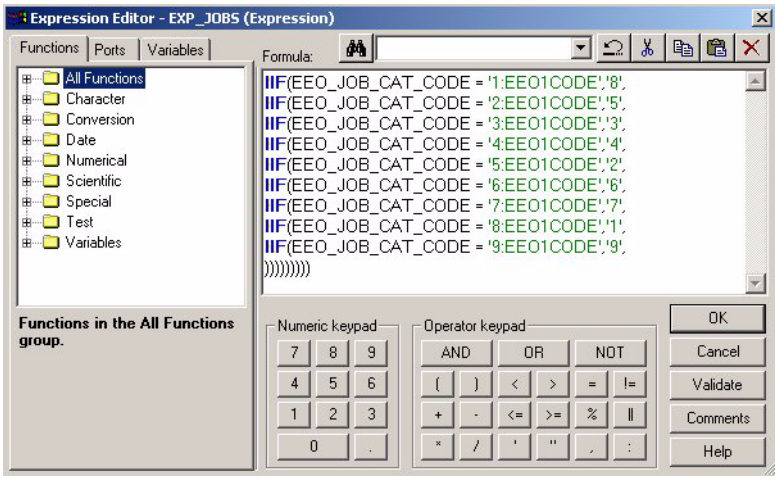
Table 47. Sample Source-Supplied EEO Job Category Code and EEO Job Category Description

EEO Job Category Code	EEO Job Category Description
1:EEO1CODE	Laborers (Unskilled)
2:EEO1CODE	Office-Clerical
3:EEO1CODE	Technicians
4:EEO1CODE	Sales Workers
5:EEO1CODE	Professionals
6:EEO1CODE	Craft Workers
7:EEO1CODE	Operatives (Semi-Skilled)
8:EEO1CODE	Officials and Managers
9:EEO1CODE	Service Workers

To modify EEO Job Category code

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. In the M_I_JOBS_EXTRACT mapping, double-click the EXP_JOBS Expression transformation.

3 Modify the EEO_JOB_CAT_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



4 Validate and save your changes to the repository.

Configuring EEO Job Category Descriptions

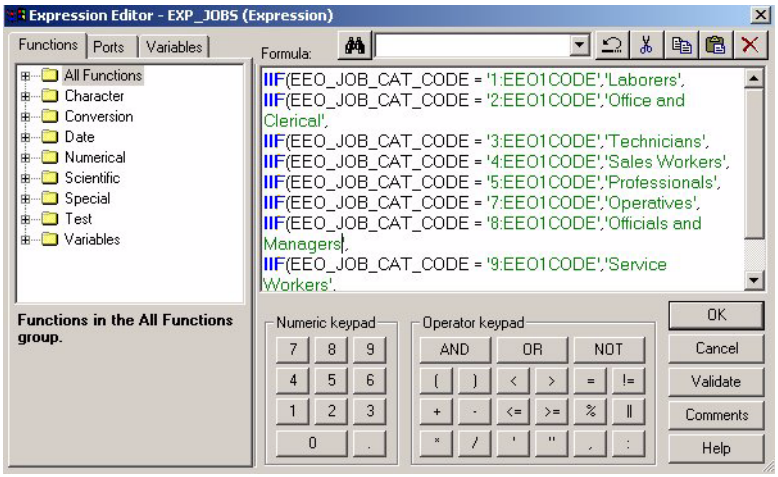
As with the Ethnic Group Description and Veteran Status Description, make sure that the EEO Job Category Description used with the source-specified code matches the description defined in the domain values. For example, in the sample source data in Table 47, the source system uses the description Office-Clerical instead of Office and Clerical for source-supplied Code '2'.

To modify the EEO Job Category description

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In the M_I_JOBS_EXTRACT mapping, double-click the EXP_JOBS Expression transformation.

3 Modify the EEO_JOB_CAT_DESC_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



4 Validate and save your changes to the repository.

Upgrading Existing Domain Values for U.S. Statutory Compliance

If you have existing information in the data warehouse tables, you must upgrade existing domain values for U.S. Statutory Compliance by running the upgrade mappings to update the historical values for IA_EMPLOYEES, OD_EMPLOYEES, IA_JOBS, and OD_JOBS.

The first part of the process is to open the upgrade mappings in the IA_52_MIGRATION repository and modify the domain values in the Expression transformations. Table 48 lists the four mappings and corresponding sessions in the migration repository that must be configured and run.

Table 48. Upgrade Mappings and Sessions

Mappings	Sessions
M_I_EMPLOYEES_IA_UPGRADE	S_M_I_EMPLOYEES_IA_UPGRADE
M_I_EMPLOYEES_OD_UPGRADE	S_M_I_EMPLOYEES_OD_UPGRADE
M_I_JOBS_IA_UPGRADE	S_M_I_JOBS_IA_UPGRADE
M_I_JOBS_OD_UPGRADE	S_M_I_JOBS_OD_UPGRADE

After you configure the domain values in the upgrade mappings, move on to the second part of the process, and enhance the existing mappings for U.S. statutory compliance. See the discussion on enhancing the existing mappings for U.S. statutory compliance, in Configuring U.S. Statutory Compliance on page 204. After both processes are completed, run the four upgrade mappings to upgrade the existing data warehouse and load control tables.

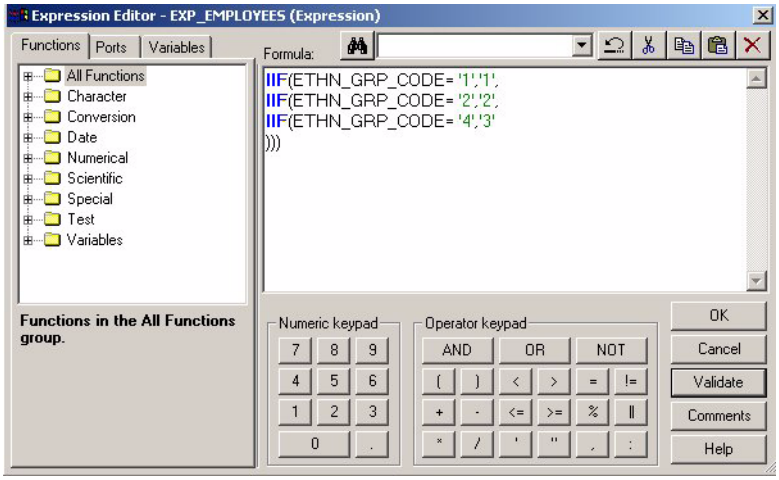
NOTE: Run the session only once to properly reflect data.

Upgrading Ethnic Group Codes and Descriptions

Refer to the domain values listed in Table 42 to modify the employee ethnic group code and description in the following procedure.

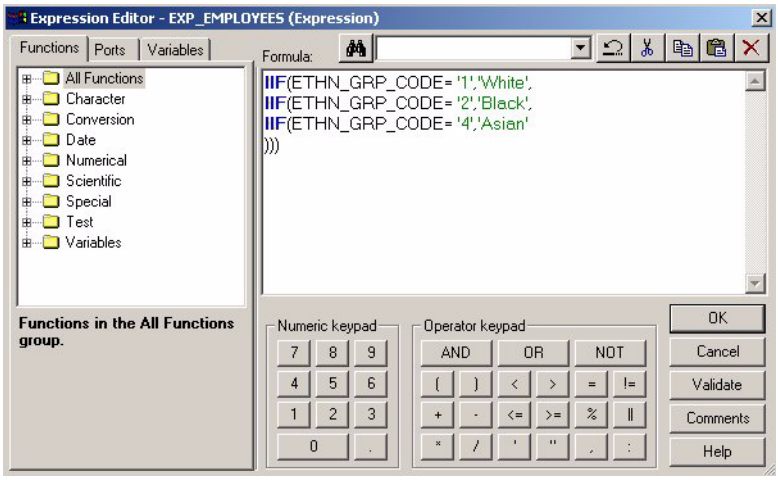
To modify the upgrade mapping for the employee Ethnic Group code and description

1. In PowerCenter Designer, connect to the IA_52_MIGRATION repository.
2. Open the Data Upgrade for Oracle Applications v11i from 5.x to 6.0 folder.
3. In the M_I_EMPLOYEES_IA_UPGRADE mapping, double-click the EXP_EMPLOYEES Expression transformation.
4. Modify the ETHN_GRP_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



5. Validate and save your changes to the repository.

6 Return to the Ports tab of EXP_EMPLOYEES, scroll to the ETHN_GRP_DESC_OUT port, and modify the expression by mapping the source-supplied values to the domain values, as shown in the following figure.



7 Validate and save your changes to the repository.

8 Repeat Step 1 to Step 7 to modify the upgrade mapping for M_I_EMPLOYEES_OD_UPGRADE, modifying the Expression transformation EXP_EMPLOYEES_UPGRADE_OD for the same columns identified in Step 4 and Step 6.

Upgrading Veteran Status Codes and Descriptions

Refer to the domain values listed in Table 44 to modify the employee veteran status code and description in the following procedure.

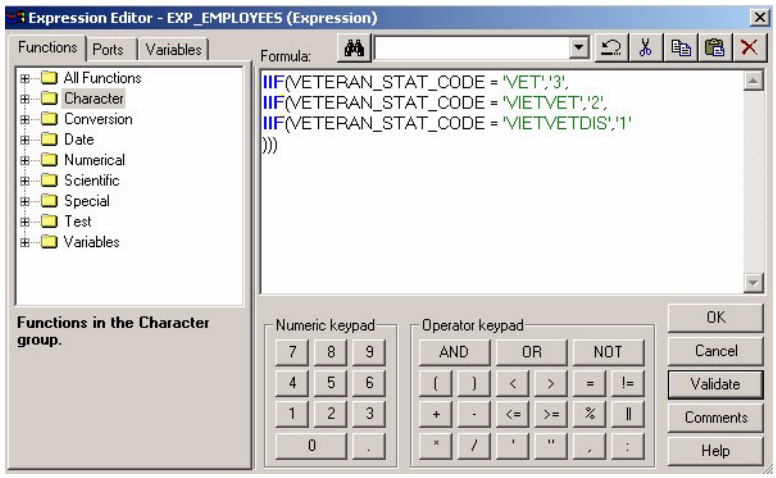
To modify the upgrade mapping for the employee Veteran Status code and description

1 In PowerCenter Designer, connect to the IA_52_MIGRATION repository.

2 Open the Data Upgrade for Oracle Applications v11i from 5.x to 6.0 folder.

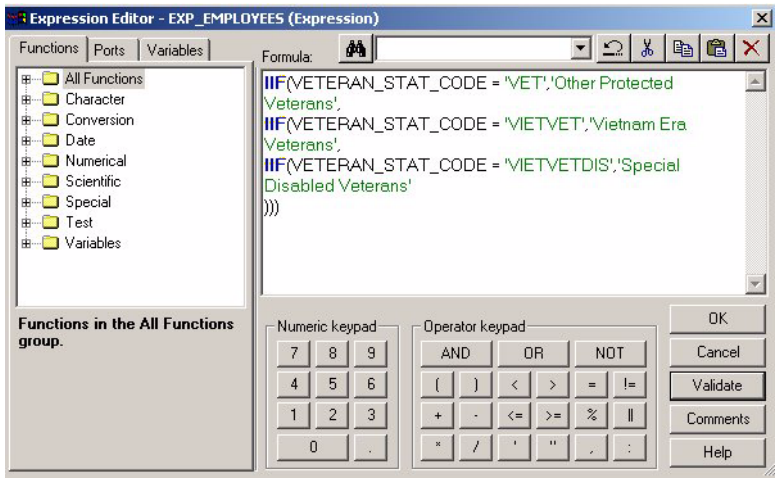
3 In the M_I_EMPLOYEES_IA_UPGRADE mapping, double-click the EXP_EMPLOYEES Expression transformation.

4 Modify the VETERAN_STAT_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



5 Validate and save your changes to the repository.

6 Return to the Ports tab of EXP_EMPLOYEES, scroll to the VETERAN_STAT_DESC_OUT port, and modify the expression by mapping the source-supplied values to the domain values, as shown in the following figure.



7 Validate and save your changes to the repository.

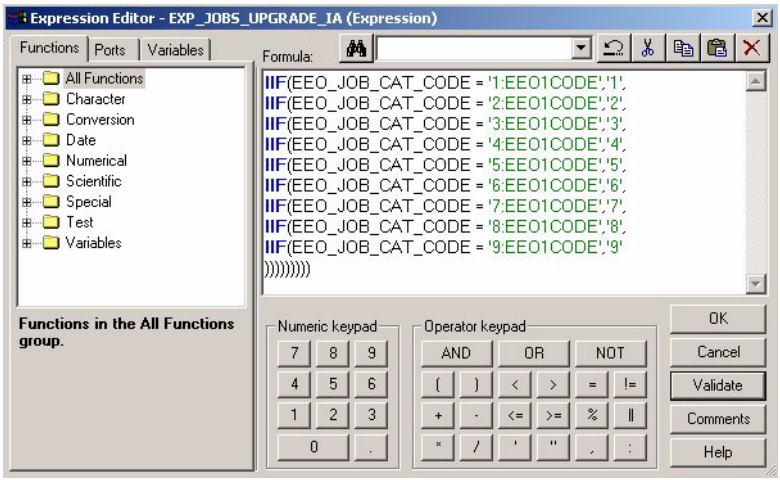
8 Repeat Step 1 to Step 7 to modify the upgrade mapping for M_I_EMPLOYEES_OD_UPGRADE, modifying the Expression transformation EXP_EMPLOYEES_UPGRADE_OD for the same columns identified in Step 4 and Step 6.

Upgrading EEO Job Category Codes and Descriptions

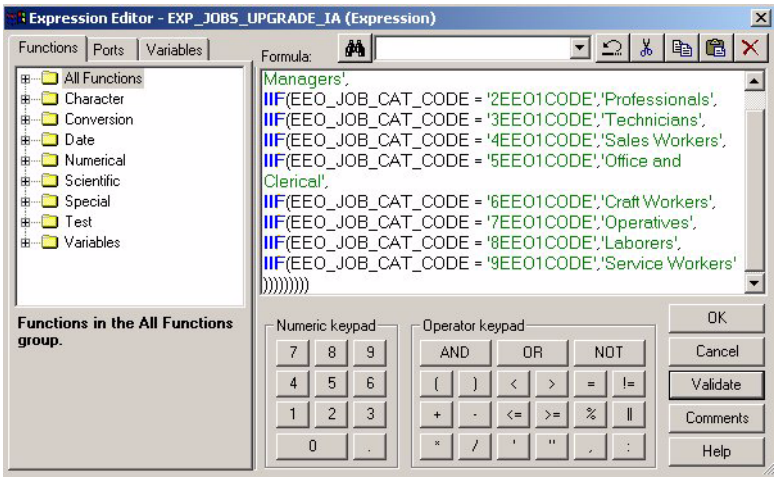
Refer to the domain values listed in Table 46 to modify the EEO job category code and description in the following procedure.

To modify the upgrade mapping for the EEO Job Category code and description

1. In PowerCenter Designer, connect to the IA_52_MIGRATION repository.
2. Open the Data Upgrade for Oracle Applications v11i from 5.x to 6.0 folder.
3. In the M_I_JOBS_IA_UPGRADE mapping, double-click the EXP_JOBS_UPGRADE_IA Expression transformation.
4. Modify the EEO_JOB_CAT_CODE_OUT port's expression by mapping the source-supplied values to the domain values, as shown in the following figure.



5. Validate and save your changes to the repository.
6. Return to the Ports tab of EXP_EMPLOYEES, scroll to the EEO_JOB_CAT_DESC_OUT port, and modify the expression by mapping the source-supplied values to the domain values, as shown in the following figure.



7. Validate and save your changes to the repository.

8 Repeat Step 1 to Step 7 to modify the upgrade mapping for M_I_JOBS_OD_UPGRADE, modifying the Expression transformation EXP_JOBS_UPGRADE_OD for the same columns identified in Step 4 and Step 6.

Before running the upgrade mappings, proceed to the following procedure to modify existing mappings.

Enhancing Existing Mappings for U.S. Statutory Compliance

The final task before running the upgrade mappings involves:

- Adding the EEO_JOB_CAT_DESC port to the corresponding staging table, TI_JOBS. (TI_EMPLOYEES already has the ETHN_GRP_DESC port, therefore no enhancement is required.)
- Enhancing the Source Qualifier for the existing load mappings, M_I_EMPLOYEES_LOAD and M_I_JOBS_LOAD, to accommodate the added ETHN_GRP_DESC and EEO_JOB_CAT_DESC columns.

First, modify the staging table, TI_JOBS, to add port EEO_JOB_CAT_DESC. You must modify the staging tables in both the Target and Source folder, as well as in the back-end database.

To modify the TI_JOBS staging table

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In the Source folder, double-click the TI_JOBS staging table located in the IA_ORA_STAGE sub-folder to open the Edit Tables window.

3 Select Replace after dragging and dropping the table into the Warehouse Designer.

4 In the Columns tab, add a new port directly below EEO_JOB_CAT_CODE port.

5 In the new field, enter the port name of EEO_JOB_CAT_DESC, and select a data type of String (254).

6 Click OK and save changes to the repository.

NOTE: Open the Target folder, and repeat the preceding procedure. Please note that you must modify the corresponding staging table in the back-end database to accommodate the new port.

The next task is to modify the Source Qualifier for the load mappings M_I_EMPLOYEES_LOAD and M_I_JOBS_LOAD.

To modify the Source Qualifier in the load mapping for M_I_EMPLOYEES_LOAD

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In the M_I_EMPLOYEES_LOAD mapping, double-click the SQ_EMPLOYEES Source Qualifier.

3 In the Ports tab, add a new port directly below the ETHN_GRP_CODE port.

4 In the new field, enter the port name of ETHN_GRP_DESC, and select a data type of String (254).

5 Click OK.

6 In the Properties tab, open SQL Query field. There are two options:

- If you have not modified the SQL, you can select Generate SQL and click OK.
- If you are not using the preconfigured SQL, you must modify the join condition manually.

7 Validate and save changes to the repository.

To modify the Source Qualifier in the load mapping for M_I_JOBS_LOAD

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In the M_I_JOBS_LOAD mapping, double-click the SQ_JOBS Source Qualifier.

3 In the Ports tab, add a new port directly below the EEO_JOB_CAT_CODE port.

4 In the new field, enter the port name of EEO_JOB_CAT_DESC, select a data type of String (254), and click OK.

5 In the Properties tab, open SQL Query field. There are two options:

- If you have not modified the SQL, you can select Generate SQL and click OK.
- If you are not using the preconfigured SQL, you must modify the join condition manually.

6 Validate and save changes to the repository.

After you have modified the domain values in the upgrade mappings, and enhanced the column structure in the existing mappings, run the upgrade mappings prior to proceeding with configuring the extract mappings. For a discussion on configuring the Employees Dimension and configuring the Jobs Dimension, see Configuring U.S. Statutory Compliance on page 204. Make sure to run the mapping only once to properly reflect data.

Configuring Workforce Profile for Oracle 11i

This section provides the necessary information on configuring the Workforce Profile functional area for Oracle 11i—configuring address types, configuring phone types, modifying the derive flag, and modifying the snapshot extract date.

Configuring Address Types

By default the address type for employee information is type 'M' for mailing type. To modify the address type you must configure the MPLT_BCO_EMPLOYEES Business Component.

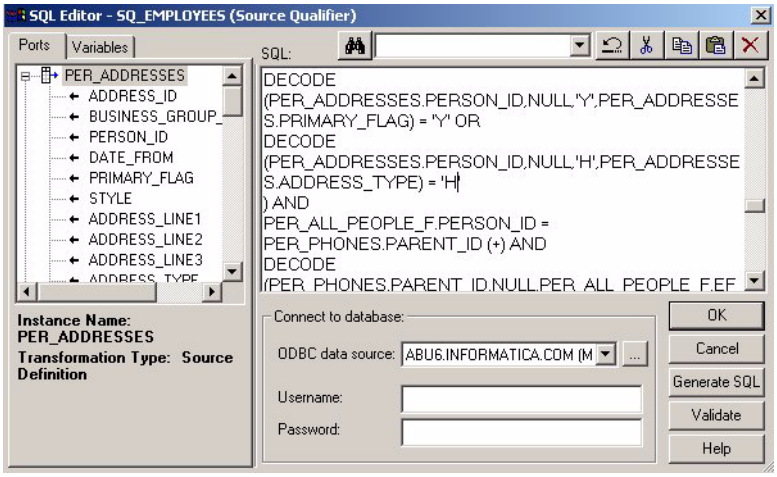
To configure Address Type

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In Mapplet Designer, open the MPLT_BCI_EMPLOYEES mapplet.

3 In the Properties tab, modify the SQL query to accommodate the new address type.

For example, the address type has been modified to 'H' for home address instead of the default 'M' for mailing, as shown in the following figure.



4 Validate and save changes to the repository.

Configuring Phone Types

By default the phone type for employee information is 'P'. To modify the phone type you must configure the MPLT_BCI_EMPLOYEES Business Component.

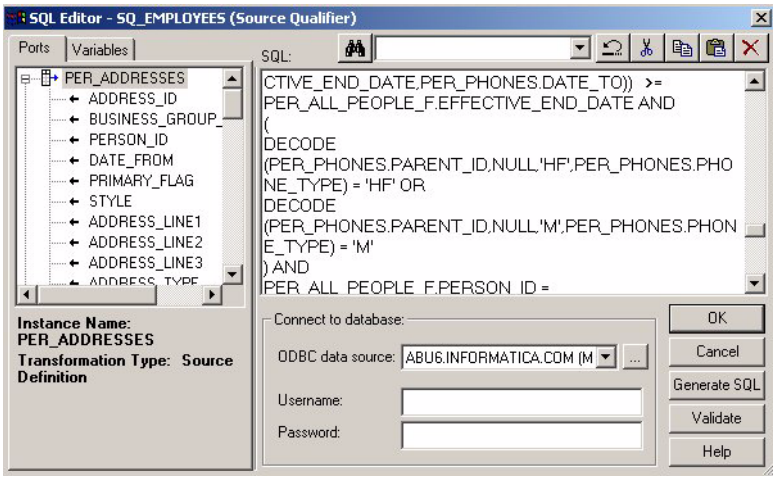
To configure Phone Type

1 In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.

2 In Mapplet Designer, open the MPLT_BCI_EMPLOYEES mapplet.

3 In the Properties tab, modify the SQL to accommodate the new phone type.

For example, the phone type has been modified to 'F' for Home Fax phone type instead of the default 'P', as shown in the following figure.



4 Validate and save changes to the repository.

Modifying the Derive Flag

The $$DERIVE_FLAG setting determines where the specific history table is populated. By default, the $$DERIVE_FLAG is set to 'N' for No. However, if you want to change this flag, you can do so by modifying the FILE_PARAMETERS_ORA11i.CSV file in the installation directory. This file contains a default value for the sessions shown in Table 49.

Table 49. Derive Flag—Default Value for the Sessions

Session	Default Value
S_M_I_EMP_HISTORY_A1_DERIVE	NO
S_M_I_EMP_HISTORY_A2_DERIVE	NO

After you have modified the file, the next time you run the session, the new Derive flag value is populated.

To modify the Default Derive flag

1 Open the FILE_PARAMETERS_ORA11i.CSV file in the installation directory, in `$pmserver\srcfiles`.

2 Replace the default Derive Flag with the new Derive flag.

Perform this action for all applicable sessions, as shown in the following figure.

S_M_I_EMP_HISTORY_A1_DERIVE:DERIVE_FLAG	0	0	0	0	S	NO
S_M_I_EMP_HISTORY_A2_DERIVE:DERIVE_FLAG	0	0	0	0	S	YES

TIP: Modify the FILE_PARAMETERS_ORA11i.CSV using Microsoft WordPad or Notepad. If you do not, the date columns in the file is truncated. Also, an alternative to modifying FILE_PARAMETERS_ORA11i.CSV is to disable the session in the PowerCenter Workflow Manager. For more information on disabling sessions, see Disabling Workflow Sessions on page 224.

3 Save and close the file.

Modifying the Snapshot Extract Date

By default, the $$EXTRACT_DATE is set to '01/01/1970' for Oracle 11i. However, if you want to modify this value, you can do so by modifying the FILE_PARAMETERS.ORA11i.CSV file in the installation directory. This file contains this default value for the sessions in Table 50.

Table 50. Sessions with '01/01/1970' as the Default Snapshot Extract Date

Session
S_M_I_EMP_SNAPSHOT_1_EXTRACT_P1
S_M_I_EMP_SNAPSHOT_2_EXTRACT_P2
S_M_I_EMP_SNAPSHOT_3_EXTRACT_P3
S_M_I_EMP_SNAPSHOT_4_EXTRACT_P4

After you have modified the file, the next time you run the session, the new snapshot extract date is populated.

To redefine the Snapshot Extract date

1 Open the FILE_PARAMETERS_ORA11i.CSV file in the installation directory, in `$pmserver\srcfiles`.

2 In the PARAM_DVALUE_1 column, enter the date for which you want the data extracted.

Perform this action for all applicable sessions. By default the date format is YYYYMMDDHH24MISS. For example, the default date of 01/01/1970 would appear as 19700101000000.

TIP: The FILE_PARAMETERS_ORA11i.CSV must be modified using Microsoft WordPad or Notepad, or else the date columns in the file is truncated.

3 Save and close the file.

16 Performing Cross-Module Configuration

This chapter provides instructions for modifying the extract, transform, and load (ETL), and the post-load processing (PLP) components of the Siebel Analytics Enterprise Data Warehouse.

This chapter contains the following topics:

Overview

As you begin performing gap analysis to determine if data is loading properly, you may find it necessary to configure the Siebel Analytics Enterprise Data Warehouse components. This chapter provides procedural information on how to configure components that are common, regardless of which module you purchased. In addition, it provides information on how to enable and disable features of the PowerCenter platform.

Configuring Extracts

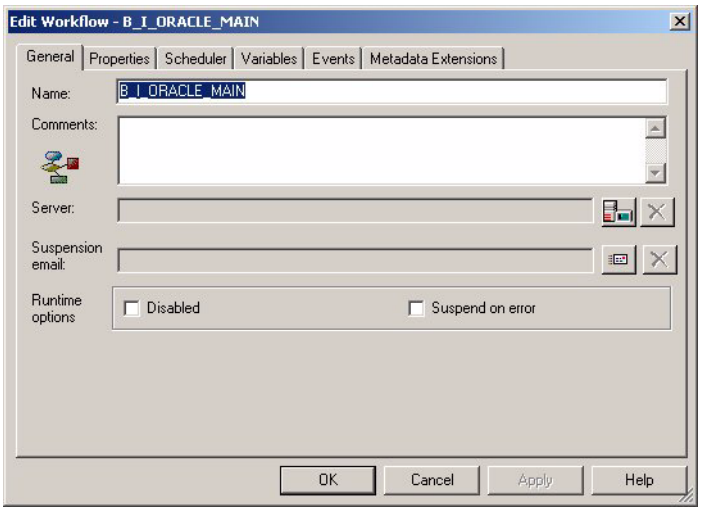
Each module has prepackaged logic to extract particular data from a particular source. This section discusses how to capture all data relevant to your reports and ad hoc queries by addressing what type of records you want and do not want to load into the data warehouse, and includes the following topics:

Disabling Workflow Sessions

After you determine that the information you want in your data warehouse, you may choose to enable or disable particular workflow sessions. If you do so, remember that most sessions must be disabled in pairs. For example, if you decide you do not want to build a particular table, you must disable both the extract and load sessions that support that object.

To disable a workflow

1. In PowerCenter Workflow Manager, open the applicable source configuration folder.
2. In the Workflow menu, select Edit to open the Edit Workflow window, as shown in the following figure.

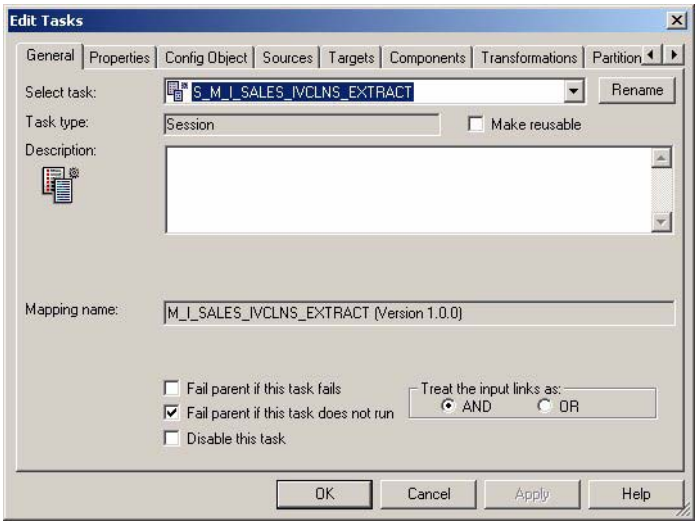


3. Select the Disabled check box to disable the workflow, and click OK.

To disable a workflow session

1. In PowerCenter Workflow Manager, open the applicable source configuration folder.

2 In the Workflow menu, select Edit to open the Edit Tasks window, as shown in the following figure.



3 Select the Disable this task check box to disable the session, and click OK.

Extracting Additional Data

Extract mappings and mapplets can be configured in the Siebel Analytics Enterprise Data Warehouse to accommodate additional source data. For example, if your business divides customer information into separate tables based on region, then you would have to set up the extract mapping to include data from both tables.

Extract mappings can be modified so that new data is loaded into extension columns that act as placeholders for additional data. Extension columns make it possible to extend any fact or dimension table without changing the schematic structure of the Siebel Analytics Enterprise Data Warehouse or making modifications to the load mapping, as the load mappings already include the extension columns. Keeping the data model intact allows you to implement upgrades without losing any customization.

Extracting New Data Using an Existing Source Table

Extract mappings generally consist of a source table or Business Component, an Expression transformation, and a staging table. If you want to extract new data using the existing mapping, you have to modify the extract mapping to include the new data by performing the following tasks:

To modify an existing mapping to include new data

1. Modify the existing Business Component to extract this information from the source, and add it to an appropriate extension column.

   TIP: You can perform calculation transformations in the Business Component mapplet of the extract mapping or in the Source Adapter mapplet of the load mapping. However, do not use performance-expensive calculations in the extract that could tie up your source transaction system. For these types of calculations, it is recommended that you perform them in the Source Adapter mapplet in the load mapping.

2. Modify the Expression transformation to perform any necessary transformations.
3. Connect all input and output ports within the extract mapping so that the data moves from the source or Business Component to the Expression transformation, and finally to the staging table's appropriate extension column.

You have to determine which type of extension column to map the data to in the staging table.

Extracting Data from a New Source Table

Business Components are packaged as mapplets, which reside in source-specific folders within the repository. Business Components are used to extract data from the source system. You can configure these mapplets to perform the following:

- Extract data from a new source table
- Set incremental extraction logic

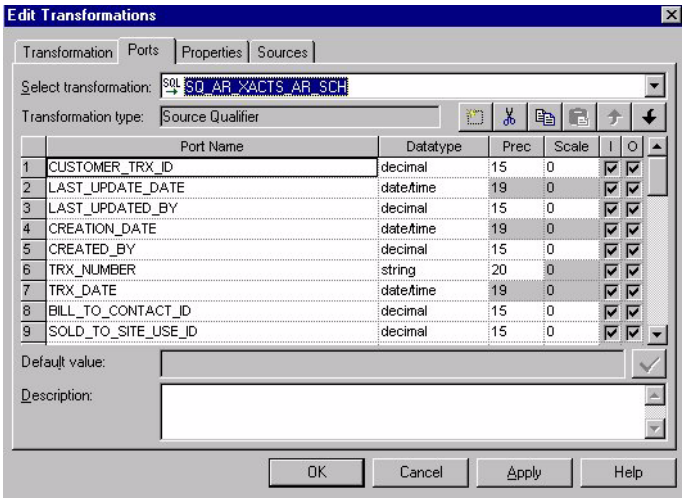
The following procedure contains instructions for adding a new table to the Business Component. The procedure includes adding a new source definition, connecting the ports to the Source Qualifier, editing the Source Qualifier, connecting the ports to the Output transformation, and editing the Output transformation.

To add a new source table to an existing Business Component mapplet

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open Mapplet Designer tool.
3. Drag the Business Component mapplet into Mapplet Designer to view the transformations that comprise the Business Component.
4. Expand the Sources folder, and copy a source table into the mapplet by dragging and dropping the table into Mapplet Designer.
5. Connect the applicable ports from the new source definition to the Source Qualifier by clicking on the port in the new source table and dragging it to the connecting port in the Source Qualifier.

6 Double-click the Source Qualifier to open the Edit Transformations box.

In the Ports tab, make any changes to the new ports for data type, precision, scale, or all these values, as necessary, as shown in the following figure.



7 Connect the applicable ports from the Source Qualifier to the Mapplet Output transformation (MAPO).

NOTE: In some cases, the Business Component contains an Expression transformation between the Source Qualifier and the MAPO.

8 In the Properties tab, make changes to the SQL query as necessary.

9 Validate and save your changes to the repository.

Filtering the Data Extract

The Siebel Analytics Enterprise Data Warehouse prepackages some filters to extract only particular types of records. For example, the Sales module, by default, extracts only booked orders. Given your unique business needs, you may need to extract nonbooked orders as well. In general, all extract filters can be managed in the same way, and the following procedure describes how to modify an extract. However, for information about module-specific extract filters, see the module-specific configuration chapters.

To modify extract filters

1 In PowerCenter Designer, open the applicable source configuration folder.

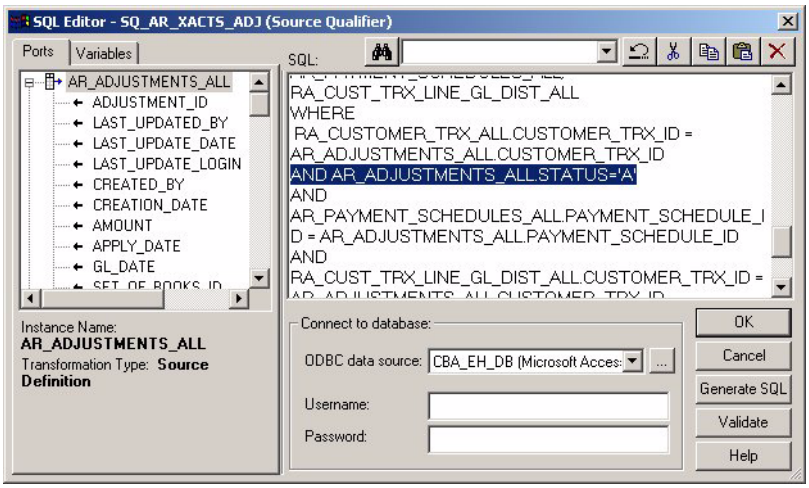
2 Open Mapplet Designer.

3 Drag the Business Component mapplet into Mapplet Designer.

4 Double-click the Source Qualifier to open the Edit Transformations window, and select the Properties tab.

5 In both the User Defined Join field and in the SQL Query field, remove or add a filter condition.

For example, if you want to change the Accounts Receivable Schedules filter so that it is not restricted to only completed schedules, you would remove the condition, as shown in the following figure.



6 Validate and save your changes to the repository.

NOTE: If a primary extract exists, you must modify both the regular mapplet and the primary extract mapplet. For information on primary extract mappings, see Understanding Primary Extract and Delete Mappings on page 233.

Configuring Loads

The Siebel Analytics Enterprise Data Warehouse prepackages load mappings for every data warehouse table. Within each load mapping, every input and output port for each extension column is already connected. Thus, if you connect new data to a staging table's extension column using the extract mapping, the default configuration pulls that data through all the load mapping's components and inserts it into the corresponding data warehouse table. However, because the load mapping does not transform data in extension columns, you may need to reconfigure the load mapping to do so. Each of the following sections describes potential configuration approaches:

- Deriving a metric or attribute from other attributes in other staging tables

  If you are going to derive a new metric or attribute from other metrics, attributes, or both in other staging tables, then you need to join all staging tables that contain the data you need. You can join these tables in the load mapping, using the Source Qualifier. For information on how to join tables in the Staging Area, see Joining Objects in the Staging Area on page 229.

- Performing calculation transformations

  It is recommended that you modify the Expression transformation to perform calculation transformations in the Source Adapter mapplet.

- Performing a lookup override at the session-level

  If you want to store an additional domained attribute, and you stored the code and code name in the IA_CODES table, then you need to modify the load session's lookup. You have to modify the SQL override to look up the code using the correct category. This lookup returns the appropriate code name from the IA_CODES table, and the ADI loads both the supplied code and the lookup code name into the target table.

- Resolving currency conversion issues

  If you are storing amount metrics, you only need to supply the amounts in one of the three currencies—document, local, or group currency. By loading one of these three values, the ADI contains logic for deriving the other two. For more information on potential configuration points with currency, see Configuring Currencies on page 249. For more information on the how each of the three types of currency is handled by the ADI, see Working with Document, Local, and Group Currencies on page 246.

Joining Objects in the Staging Area

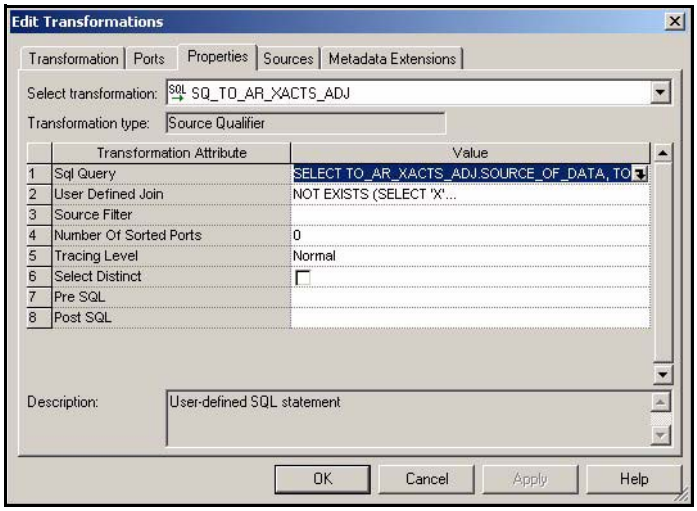
If you want to join objects from multiple tables, you can do so in the Staging Area. This section describes how to join objects in the Staging Area by configuring the load mapping.

To join staging tables in the Staging Area

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable load mapping.
3. Copy the table you want to add to the load mapping by dragging and dropping it into the mapping definition in Mapping Designer.

   Perform this step for all tables you want to join in the Staging Area.
4. Drag and drop the columns you wish to join from the source definition to the Source Qualifier.
5. Double-click the Source Qualifier transformation to open the Edit Transformations box, and select the Properties tab.

6 Select the small arrow in the Value column to open the SQL Editor, as shown in the following figure.



7 Edit the SQL statement to add the join conditions between the new table and the existing table in the mapping.

8 Drag and drop those columns from the Source Qualifier to the respective ports in the Source Adapter mapplet.

9 Validate and save your changes to the repository.

Creating a Source Adapter Mapplet

The majority of all source-specific transformations occur in the Source Adapter mapplet; source-independent transformations generally take place in the Analytic Data Interface (ADI). The Source Adapter mapplet converts source-specific data elements that come from a staging table into those data elements required by the source-independent ADI.

Figure 15 illustrates the three components of the Source Adapter mapplet that allow transformations of data to occur. The three components are Mapplet Input (MAPI), Expression transformation (EXP), and Mapplet Output (MAPO).

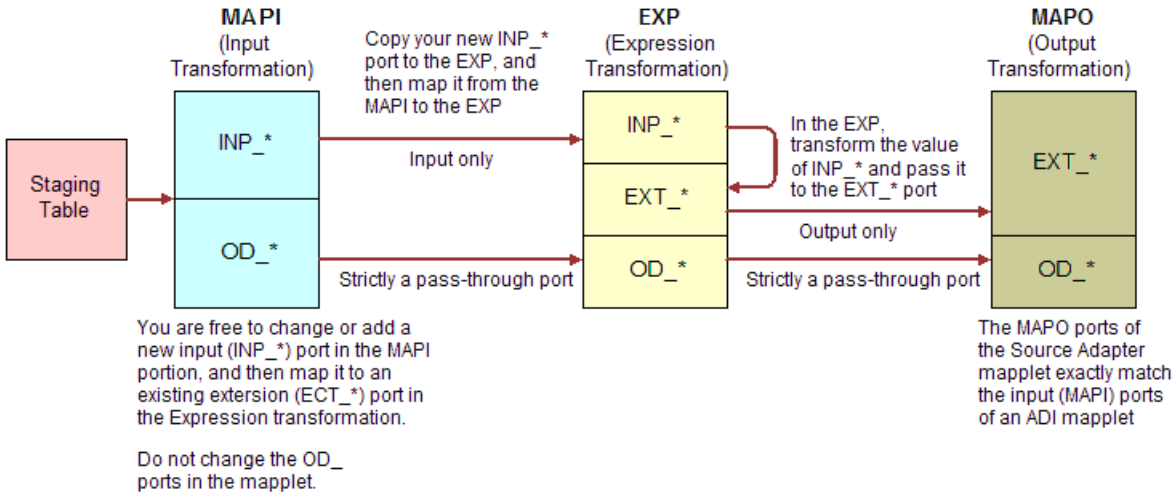


Figure 15. Input, Expression, and Output Ports of a Source Adapter Mapplet

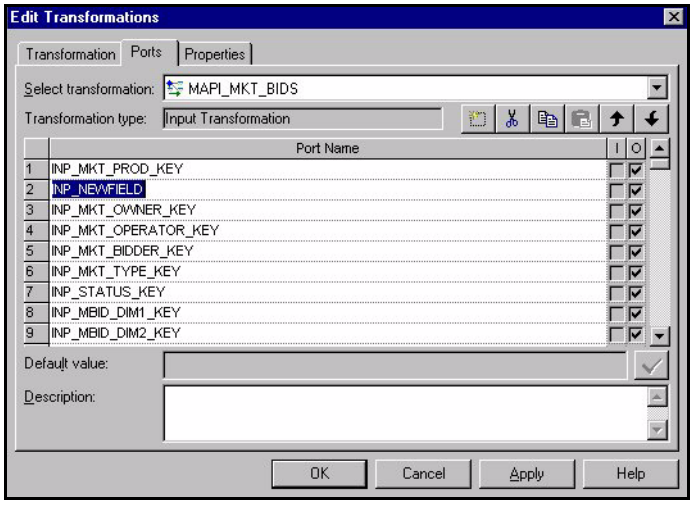
In Figure 15, notice that the MAPI receives data from the Staging table. This data is passed through ports prefixed with INP_. If the input data is transformed, the data is passed to the Expression transformation (EXP) as input only. After the data is transformed, it is output through a new port, which is prefixed with EXT_. If the data is not transformed, it comes in as input-only and leaves through an output-only port.

If you want to add a new transformation, you must add a new port to contain the expression that is used to transform the data.

To add a new port to the Source Adapter mapplet

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable Source Adapter mapplet.

3 Double-click the MAPI component of the mapplet, and add a new input port following the `INP_*` naming convention, as shown in the following figure.



4 Copy the new input port from the MAPI to the Expression transformation.

5 Connect the new port from the MAPI to the Expression transformation.

6 In the Expression transformation, uncheck the Output indicator for the new input port; you use the value from this port in an transformation expression.

7 Perform any necessary transformations within the Expression transformation.

The transformed data is passed out of an EXT_* output-only port.

8 Connect the port from the Expression transformation to the MAPO.

Figure 15 shows that the ports in the Mapplet Output match the input ports of the ADI mapplet exactly, with the exception of new ports. If you are adding a new port, you must connect it to an extension column, because you cannot add the new port to the ADI.

9 Validate and save your repository.

Filtering and Deleting Records

In a typical implementation, records that are deleted from your source system are not removed from the Siebel Analytics Enterprise Data Warehouse. If you want to delete records in the data warehouse that were removed from the source system's database and archived in a separate database, you must enable the primary extract and delete mappings.

Primary extract mappings flag records that should be deleted from the data warehouse. Delete mappings perform the deletion action. When enabled, primary extract and delete mappings by default look for any records removed from the source system's database. If these mappings find that the records no longer exist in that database, the mappings remove them from the data warehouse as well.

CAUTION: It is important to note that delete and primary extract mappings must always be disabled together; you may not disable only one type.

Stored in the [SOURCE ABBREVIATION]_[SUBJECT]_MAIN workflow, delete and primary extract sessions can be found in each application's fact-extract and fact-load subbatches.

Understanding Primary Extract and Delete Mappings

Before you decide to enable primary extract and delete sessions, it is important to understand their function within the Siebel Analytics Enterprise Data Warehouse. Primary extract and delete mappings allow your analytics system to determine which records are removed from the source system by comparing primary extract staging tables with the most current Siebel Analytics Enterprise Data Warehouse table.

The primary extract mappings perform a full extract of the primary keys from the source system. Although many rows are generated from this extract, the data only extracts the Key ID and Source ID information from the source table. The primary extract mappings load these two columns into staging tables that are marked with a *_PE suffix.

Figure 16 provides an example of the beginning of the extract process. It shows the sequence of events over a two day period during which the information in the source table has changed. On day one, the data is extracted from a source table and loaded into the Siebel Analytics Enterprise Data Warehouse table. On day two, Sales Order number three is deleted and a new sales order is received, creating a disparity between the Sales Order information in the two tables.

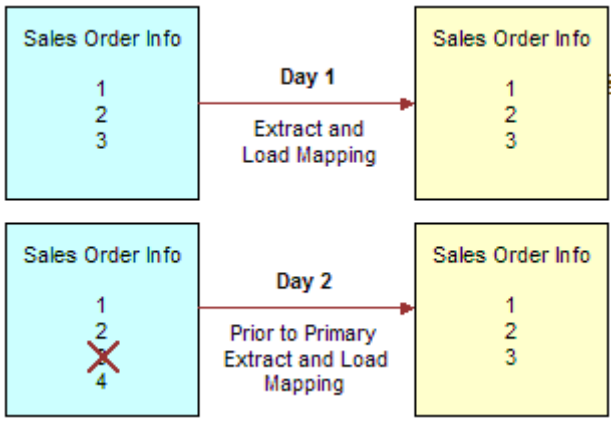


Figure 16. Changing Sales Order Information

Figure 17 shows the primary extract and delete process that occurs when day two's information is extracted and loaded into the Siebel Analytics Enterprise Data Warehouse from the source. The initial extract brings record four into the Siebel Analytics Enterprise Data Warehouse. Then, using a primary extract mapping, the system extracts the Key IDs and the Source IDs from the source table and loads them into a primary extract staging table.

The extract mapping compares the keys in the primary extract staging table with the keys in the most current the Siebel Analytics Enterprise Data Warehouse table. It looks for records that exist in the Siebel Analytics Enterprise Data Warehouse, but do not exist in the staging table (in the preceding example, record three) and sets the delete flag to Y in the Source Adapter mapplet, causing an eventual deletion in the Siebel Analytics Enterprise Data Warehouse.

The extract mapping also looks for any new records that have been added to the source, and which do not already exist in the Siebel Analytics Enterprise Data Warehouse; in this case, record four. Based on the information in the staging table, Sales Order number three is physically deleted from Siebel Analytics Enterprise Data Warehouse, as shown in Figure 17. When the extract and load mappings run, the new sales order is added to the warehouse.

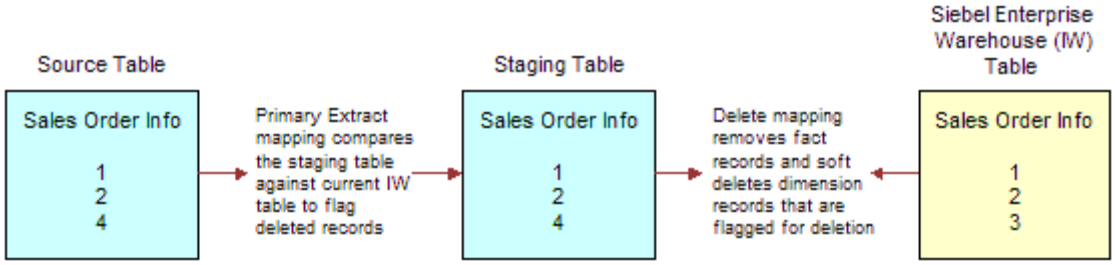


Figure 17. Primary Extract and Delete Mappings

Working with Primary Extract and Delete Mappings

The primary extract and delete mappings serve a critical role in identifying which records have been physically deleted from the source system. However, there are some instances when you can disable or remove the primary extract and delete mappings, such as when you want to retain records in the data warehouse that were removed from the source systems' database and archived in a separate database.

Because delete mappings use Source IDs and Key IDs to identify purged data, if you are using multiple source systems, you must modify the SQL query statement to verify that the proper Source ID is used in the delete mapping. In addition to the primary extract and delete mappings, the configuration of the delete flag in the ADI also determines how record deletion is handled.

You can manage the extraction and deletion of data in the following ways:

- Deleting the configuration for source-archived records
- Deleting records from a particular source
- Enabling delete and primary-extract sessions
- Configuring the Record Deletion flag

- Configuring the Record Reject flag

This topic provides procedures for these management tasks.

Deletion Configuration for Source-Archived Records

Some sources archive records in separate databases and retain only the current information in the main database. If you have enabled the delete mappings, you must reconfigure the delete mappings in the Siebel Analytics Enterprise Data Warehouse to retain the archived data.

To retain source-archived records in the Siebel Analytics Enterprise Data Warehouse, you need to perform two tasks on each delete mapping for your application. First, create a parameter for the archive date, and then edit the SQL Query field in the Source Qualifier. For a list of all delete sessions, see the discussion on disabling delete and primary extract sessions in Working with Primary Extract and Delete Mappings on page 234.

To create a parameter for the archive date

1. In PowerCenter Designer, open the applicable source configuration folder, and open a delete mapping.
2. Select Mappings > Parameters and Variables and select Add.
3. Enter `$$ARCHIVE_DK` as the name, and select Parameter for the type.
4. Select Date/Time for the data type, using the format that matches your source system.
5. Set the precision or scale required.
6. Double-click the Source Qualifier of your delete mapping.
7. Select the Variables tab, then select the parameter you just created, and click OK.

To add a clause to your Source Qualifier

1. In the delete mapping, open the Source Qualifier to edit the SQL Query field.
2. In the Properties tab, edit SQL Query field.

   Use the Archive Date as a filter in the `WHERE` clause.

   For example, if you were to create this statement for sales order lines, your clause would look like this:

   ```
   SELECT <column(s)> FROM OD_SALES_ORDLNS

   WHERE CREATED_ON_DK > $$ARCHIVE_DK AND NOT IN TO_SALES_ORDLNS_PE
   ```

3. Validate the SQL, and save your changes to the repository.

Deleting Records from a Particular Source

Delete mappings use Source IDs and Key IDs to identify the data that has been purged from the source system. Therefore, if you are using multiple source systems (where each type is identified by a Source ID), you must verify that the proper Source ID is used in the delete mapping so that only the desired records are deleted. To specify the correct source, edit the SQL query statement in the delete mapping.

To specify the source for the delete mapping

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the delete mapping.
3. Open the Source Qualifier transformation to edit the SQL statement in the SQL Query field.

   Edit the SOURCE_ID expression in the OD table by adding a source abbreviation. A sample source abbreviation is shown in the following table.

Source	Source Abbreviation for SOURCE_ID
Oracle 11i	OAP11i

4. Validate the SQL, and save your changes to the repository.

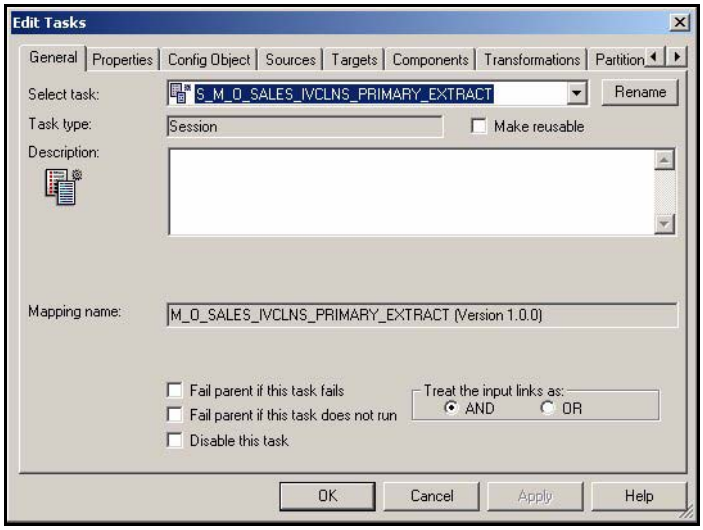
Enabling Delete and Primary Extract Sessions

If you want to remove your source-deleted records in the Siebel Analytics Enterprise Data Warehouse, you need to enable the delete and primary extract sessions for your application.

To enable primary extract and delete sessions

1. In PowerCenter Workflow Manager, open the applicable source configuration folder.

2 Double-click the session to open the Edit Tasks dialog box, as shown in the following figure.



3 Edit the session, and then clear the Disable the task check box to enable the session.

4 Repeat these steps for each applicable primary extract and delete sessions.

Configuring the Record Deletion Flag

Record deletion is performed in the ADI if there are records you want to delete regardless of whether or not they have been deleted from the source. Record deletion is also performed in the ADI when the primary extract cannot determine if records have been deleted from the source, as in the case of Web logs where there are no primary keys for the extract to recognize. The Delete Flag determines how the record deletion is handled.

The Delete Flag can be configured in the Source Adapter mapplet by modifying the transformation for the EXT_DELETE_FLAG port. To reconfigure the handling of deletions, you can modify the Delete Flag definition. There are different values that you can use when defining your Delete Flag; these values depend on whether you are dealing with a fact table or a dimension table.

When you define the Delete Flag for fact tables, it is recommended you use a conditional statement. For example, you could enter the following statement:

```
IIF(<SOURCE>.<COLUMN_NAME> = 'Y', 'Y', 'N')
```

For fact tables, there are two values for which you can set the Delete Flag—'Y' and 'N'. By setting your Delete Flag to 'Y', records that already exist in the data warehouse are purged from the fact table by the use of delete mappings. By setting your Delete Flag to 'N' or any other value besides 'Y', your records are not deleted. If the record is marked as deleted by the source and has not yet been loaded into the data warehouse, then the record is not loaded.

When defining the Delete Flag for dimension tables, it is recommended that you use a conditional statement as well. For example, you could enter the following statement:

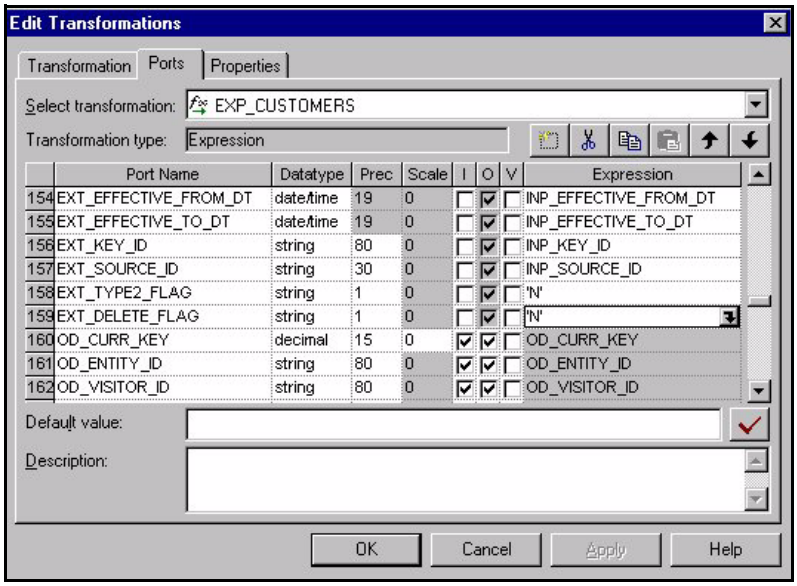
```
IIF(<SOURCE>.<SOME_COLUMN_NAME> = 'Y', 'D', 'N')
```

For dimension tables, there are two values for which you can set your Delete Flag—'D' and 'N'. By setting your Delete Flag to 'D', your records are marked for deletion, but are not be purged from the dimension table just in case you want to query these values at a later time. If you wish to analyze historical dimension records, you must enable Type II functionality, which updates records by inserting a new record and leaving the old record intact. For more information about Type II dimensions, see Type I and Type II Slowly Changing Dimensions on page 241.

NOTE: If you set the Delete flag as 'P' for a dimension record, the deletion logic behaves as if it was marked as 'D'. No dimensions are ever purged from the data warehouse.

To configure the Delete flag

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable Source Adapter mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4. In the Ports tab, edit the expression for the EXT_DELETE_FLAG port.

   For example, if your source system sets the document type to 'DEL' when a record is to be deleted, this expression should contain a statement similar to the following:

   ```
   IIF (DOCUMENT_TYPE = 'DEL', 'D', 'N')
   ```

5. Validate and save your changes to the repository.

Configuring Record Reject Flag

The purpose of rejecting records is to verify that they are not loaded into the data warehouse. You can set up the rejection logic in one of two places—the Source Qualifier in the Business Component mapplet of an extract mapping or the Source Adapter mapplet of a load mapping.

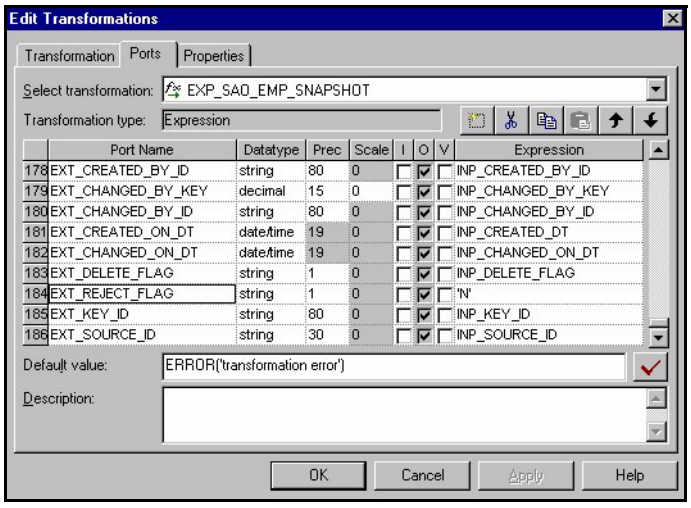
By default, the Siebel Analytics Enterprise Data Warehouse provides a reject flag in the Source Adapter mapplet that you can use to set up your record rejection logic. If the Reject Flag is set to 'Y' for any records, the ADI skips those records and does not load it into the data warehouse. However, if the Reject Flag is set to 'N' or any other value, the ADI processes the record. The reject logic must be configured in the Source Adapter mapplets according to your requirements.

The Reject Flag can be configured in the Source Adapter mapplet by modifying the transformation for the port EXT_REJECT_FLAG.

NOTE: If you want to set up the rejection logic in the Source Qualifier in the extract mapping, you can do so. The Siebel Analytics Enterprise Data Warehouse performs this in the load mapping because some extract mappings load multiple staging tables in the data warehouse. Some staging tables should not reject the records, while other staging tables should.

To configure the Reject flag

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable Source Adapter mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4. In the Ports tab, edit the expression for the EXT_REJECT_FLAG port.
5. Validate and save your changes to the repository.

Configuring Slowly Changing Dimensions

The Siebel Analytics Enterprise Data Warehouse provides Type II slowly changing dimension (SCD) functionality, which allows you to track the history of updates to dimension records. When a record in the Siebel Analytics Enterprise Data Warehouse has an update, the updated information is posted into a new row and the old information is kept for historical reporting purposes.

The Siebel Analytics Enterprise Data Warehouse identifies and applies the slowly changing dimension logic chosen by the user after data has been extracted and transformed to be source independent, as shown in Figure 18. Users may configure the Source Adapter mapplet to support both Type I SCDs, in which data is overwritten with updates, and Type II SCDs, in which the original records are maintained while a new record stores the updated data. Choosing Type I or Type II SCDs depends on identifying your historically significant attributes.

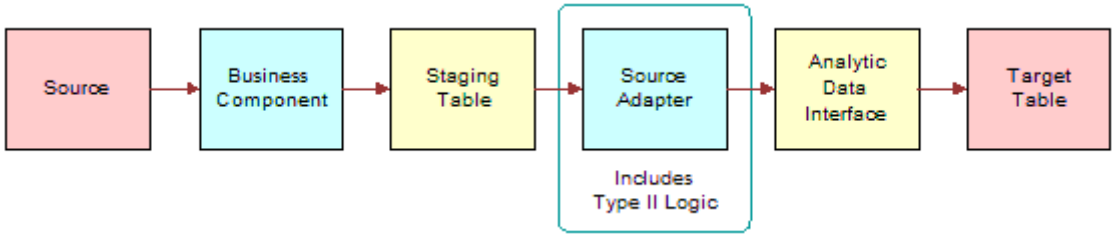


Figure 18. Slowly Changing Dimension Logic Flow

Identifying Historically Significant Attributes

You may want to retain a history of all the updates to a particular dimension so that you can use them in reports. These dimensions are known as *historically significant* attributes. For example, if a customer moves to a different region and you assign that customer a new regional salesperson and territory ID, you may want to keep records of that customer's account history with the original salesperson and territory ID. In this case, the salesperson and territory IDs are *historically significant* attributes. In contrast, you may have a load that populates the telephone number field. If your business does not perform data analysis on phone number history, then this information may be considered a *historically insignificant* attribute.

Identifying attributes as significant or insignificant allows you to determine the type of SCD you require. However, before you can select the appropriate type of SCD, you must understand their differences.

Understanding the Extract View

The extract view of any given table in the Staging Area consists of four types of records:

- New records
- Changed records with data that is historically insignificant

- Changed records having historical significance
- Changed records whose changes have no significance of any kind and are ignored altogether

Of the four kinds of records, only the first three are of interest for the data mart. Of those three, brand new records and records whose changes are tracked as SCDs are both treated as new and become inserts into the data warehouse. Records with changes that are important but not historically tracked are overwritten in the data warehouse, based on the primary key.

Type I and Type II Slowly Changing Dimensions

After you have correctly identified your significant and insignificant attributes, you can configure the Siebel Analytics Enterprise Data Warehouse based on the type of slowly changing dimension (SCD) that best fits your needs—Type I or Type II.

Type I Slowly Changing Dimension

A Type I SCD overwrites the column's value and is the default SCD for the Siebel Analytics Enterprise Data Warehouse. Although a Type I does not maintain history, it is the simplest and fastest way to load dimension data. Type I is used when the old value of the changed dimension is not deemed important for tracking or is an historically insignificant attribute. For example, you may want to use Type I when changing incorrect values in a column.

In Figure 19, the State Name column for the supplier KMT is changed in the source table Suppliers, because it was incorrectly entered as California. When the data is loaded into the data warehouse table, no historical data is retained and the value is overwritten. If you look up supplier values for California, records for KMT do not appear; they only appear for Michigan as they should have from the beginning.

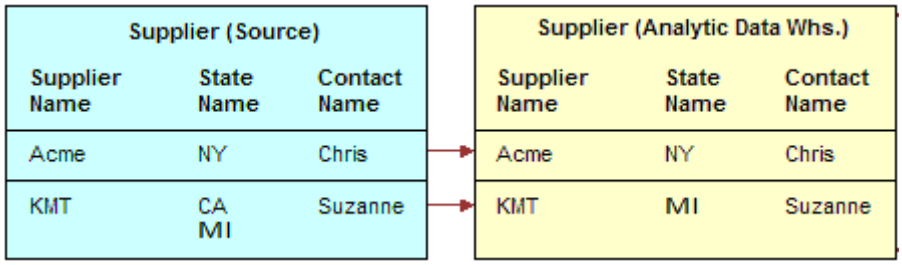


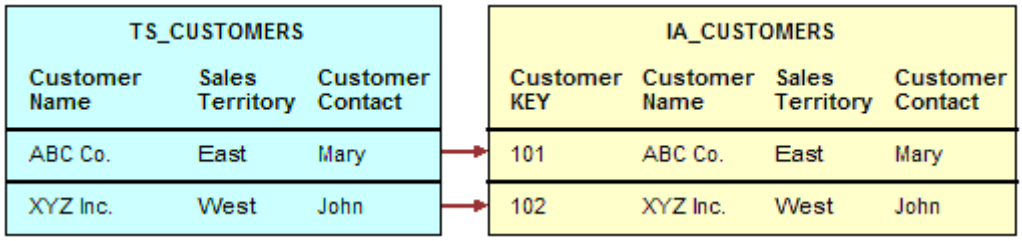
Figure 19. Type I Slowly Changing Dimension Example

Type II Slowly Changing Dimension

A Type II SCD creates another record and leaves the old record intact. Type II is the most common SCD because it allows you to track historically significant attributes. The old records point to all history prior to the latest change, and the new record maintains the most current information.

Slowly changing dimensions work in different parts of a star schema (the fact table and the dimension table). In Figure 20, shows how an extract table (TS_CUSTOMERS) becomes a data warehouse dimension table (IA_CUSTOMERS). Although there are other attributes that are tracked, such as Customer Contact, in this example there is only one *historically tracked attribute,* Sales Territory. This attribute is of historical importance because businesses frequently compare territory statistics to determine performance and compensation. Then, if a customer changes region, the sales activity should be recorded with the region that earned it.

This example deals specifically with a single day's extract, which brings in a new record for each customer. The extracted data from TS_CUSTOMERS is loaded into the target table IA_CUSTOMERS, and each record is assigned a unique primary key (Customer Key).



TS_CUSTOMERS

Customer Name	Sales Territory	Customer Contact
ABC Co.	East	Mary
XYZ Inc.	West	John

IA_CUSTOMERS

Customer KEY	Customer Name	Sales Territory	Customer Contact
101	ABC Co.	East	Mary
102	XYZ Inc.	West	John

Figure 20. Day One: The CUSTOMERS Extract and Data Warehouse Tables

However, this data is not static; the next time a data extract shows a change for your customers in IA_CUSTOMERS, the records must change. This situation occurs when slowly changing dimensions are invoked. Figure 20 shows that records for the two customers, ABC Co., and XYZ inc. have changed when compared with Figure 19. Notice that ABC's Customer Contact has changed from Mary to Jane, and XYZ's Sales Territory has changed from West to North.

As discussed earlier in this example, the Customer Contact column is historically insignificant; therefore a Type I SCD is applied and Mary is overwritten with Jane. Because the change in ABC's record was a Type I SCD, there was no reason to create a new customer record. In contrast, the change in XYZ's record shows a change of sales territory, an attribute that is historically significant. In this example, the Type II slowly changing dimension is required.

As shown in Figure 21, instead of overwriting the Sales Territory column in the XYZ's record, a new record is added, assigning a new Customer Key, 172, to XYZ in IA_CUSTOMERS. XYZ's original record, 102, remains and is linked to all the sales that occurred when XYZ was located in the West sales territory. However, new sales records coming in are now attributed to Customer Key 172 in the North sales territory.

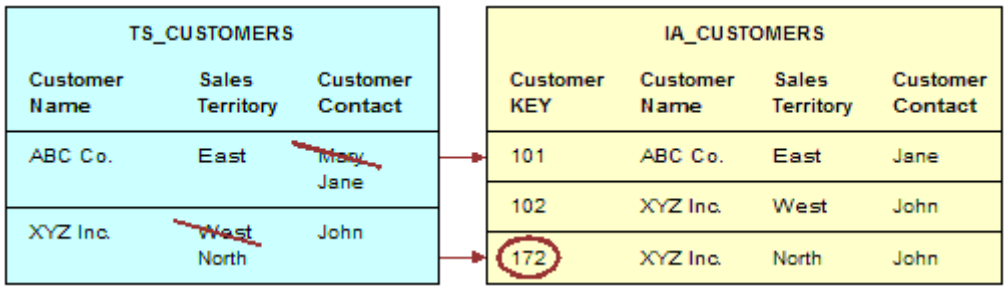


Figure 21. Changes in the Customer Extract Table

Effective Dates

Effective dates specify when a record was effective. For example, if you load a new customer's address on January 10, 2003 and that customer moves locations on January 20, 2003, the address is only effective between these dates. Effective Dates are handled in the following manner:

- If the source supplies both effective dates, these dates are used in the warehouse table.
- If the source does not supply both the effective to and effective from dates, then the Type II logic creates effective dates.
- If the source supplies one of the two effective dates, then the Siebel Analytics Enterprise Data Warehouse can be set up to populate the missing effective dates using a wrapper mapping. This situation is discussed in this section. By default, these wrapper sessions are disabled and need to be enabled in order to be executed.

For example, in the IA_CUSTOMERS table previously discussed, XYZ moved to a new sales territory.

If your source system supplied historical data on the location changes, your table may contain a record for XYZ in the West sales territory with an effective from date of January 1, 2001 and an effective to date of January 1, 3714. If the next year your source indicates XYZ has moved to the North sales territory, then a second record is inserted with an effective from date of January 1, 2002, and an effective to date of January 1, 3714, as shown in Table 51.

Table 51. Records Before a Wrapper Session

IA_CUSTOMER					
Customer Name	**Sales Territory**	**Customer Contact**	**Effective From**	**Effective To**	**Current**
ABC	East	Jane	1/1/2001	1/1/3714	Y

Table 51. Records Before a Wrapper Session

IA_CUSTOMER					
Customer Name	**Sales Territory**	**Customer Contact**	**Effective From**	**Effective To**	**Current**
XYZ	West	John	1/1/2001	1/1/3714	Y
XYZ	North	John	1/1/2002	1/1/3714	Y

Note your first record for XYZ still shows as effective from January 1, 2001 to January 1, 3714, while a second record has been added for XYZ in the North territory with the new effective from date of January 1, 2002. In this second record the effective to date remains the same, January 1, 3714.

When you schedule a wrapper session to execute, the effective dates for the first XYZ are corrected (January 1, 2001-January 1, 2002), and the Current Flag is adjusted in the Analytic Data Interface (ADI) so that only the second record (January 1, 2002-January 1, 3714) is set to Y. After the wrapper session completes its work, you have Type II information for XYZ in your data warehouse rather than two disparate records, as shown in Table 52.

Table 52. Records After a Wrapper Session

IA_CUSTOMER					
Customer Name	**Sales Territory**	**Customer Contact**	**Effective From**	**Effective To**	**Current**
ABC	East	Jane	1/1/2001	1/1/3714	Y
XYZ	West	John	1/1/2001	1/1/2002	N
XYZ	North	John	1/1/2002	1/1/3714	Y

In the previous paragraph, the wrapper session corrected the effective to dates and current flag. However, if the record's dates had been correct, the wrapper mapping would simply have set the current flag as needed, because its logic is set to check dates and flags and only adjust columns that contain discrepancies. Finally, if your source system does not supply any Type II information, you may disable the wrapper session completely; in this case all Type II work is handled by the Analytics Data Interface mapplet.

Enabling Type II Slowly Changing Dimensions (SCDs)

You can configure the Siebel Analytics Enterprise Data Warehouse to maintain history down to the column level. For example, if you are loading the IA_ACTIVITY_COSTS table and you only want to maintain history for the AVG_ACTV_DURATION column, you can set the Type II flag to 'Y' for this column and 'N' for all other columns. A brief summary is provided in Table 53.

Table 53. Types of Slowly Changing Dimensions Summary

Slowly Changing Dimensions Type	Type II Flag	Description
Type I Slowly Changing Dimension	N	Overwrites the data with the latest value.
Type II Slowly Changing Dimension	Y	Creates a new record for the updated records and, if applicable, updates the effective dates and current flag for any existing record.

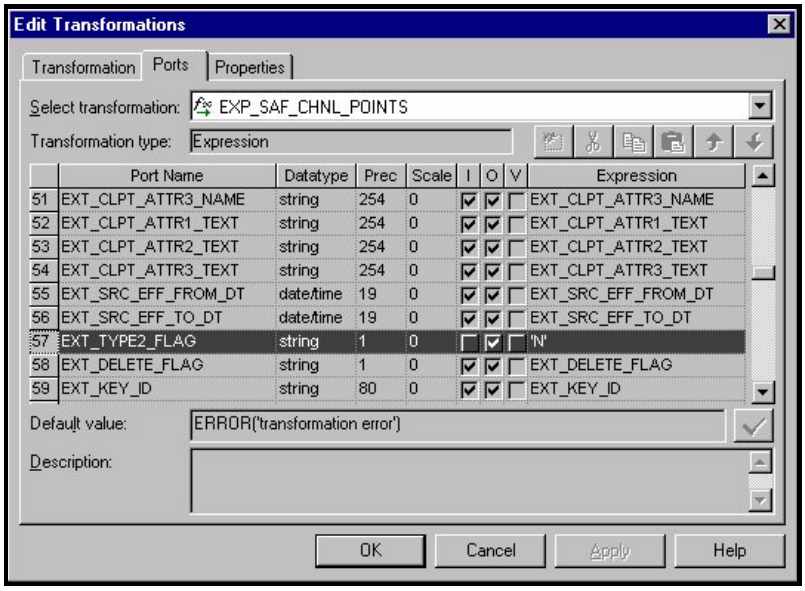
If you want to use Type II SCDs, you need to set the value of the Type II Flag. By default it is set to 'N', but you can enter a conditional statement that sets the flag to 'Y'. For example, you may only want to create new records if particular columns change values. In this case, you should set this up in your conditional statement.

The Type II Flag can be configured in the Source Adapter mapplet by modifying the port EXT_TYPE2_FLAG in the Expression transformation. Use the following procedure.

To configure the Type II flag

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable Source Adapter mapplet.

3 Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4 In the Ports tab, edit the expression for the EXT_TYPE2_FLAG port.

The default for the Type II Flag is 'N'. Here you can enter 'Y' to enable Type II functionality for all columns in the table. You can also enable only certain columns to maintain history. To enable some columns, but not others, you need to insert a conditional statement. For example, if you want to maintain history only for the Channel Point Name column, then you could write the following conditional statement:

```
IIF(EXT_CHNL_POINT_NAME!= OD_CHNL_POINT_NAME, 'Y', 'N')
```

In this case, only the Channel Point Name column has historical values—all other columns do not.

5 Validate and save your changes to the repository.

Working with Document, Local, and Group Currencies

Currency lookups are required because your business may have transactions involving multiple currencies. To create a meaningful report, you have to use a common currency. The Siebel Analytics Enterprise Data Warehouse provides a means for converting a variety of currencies, as well as prepackaging the following three currency types for each amount stored in the Siebel Analytics Enterprise Data Warehouse:

- **Document currency.** The currency of the transaction. For example, if you purchase a chair from a supplier in Mexico, the document currency is probably the Mexican peso.

- **Local currency.** The currency in which the financial books, including the transaction, are closed. For example, if your business organization is located in France and orders a part from a supplier in Britain, it may pay in British pounds, but it closes its books in French francs. In this case the local currency for the transaction is French francs and the document currency for the transaction is British pounds. The local currency is useful when each business unit of the enterprise creates its own internal reports. For example, your Japanese site may produce internal reports using Japanese yen, while your United States site may produce internal reports using United States dollars.
- **Group currency.** The standard currency used by your entire enterprise. For example, if a multinational enterprise has its headquarters in the United States, its group currency is probably U.S. dollars. The group currency is useful when creating enterprise-wide reports.

For every monetary amount extracted from the source, the ADI loads the document, local and group currency amounts into the target table. The method that the ADI uses to load the three different currency values depends on what the source provides. There are three possible situations that can occur:

- Source Provides Three Currency Amounts on page 248

  In this situation, the source system provides all three currency amounts. All three amounts are extracted and loaded into the corresponding Siebel Analytics Enterprise Data Warehouse table; the system does not need to do any currency conversions.

- Source Provides Document Amount, Codes, and Exchange Rates for Local and Group Currencies on page 248

  In this situation, the source system provides the document currency amount and the exchange rates for finding the local and group currency amounts.

- Source Provides the Document Currency Amount on page 249

  In this situation, the source system provides the document currency amount, but it does not provide the local and group currency amounts or exchange rates used for currency conversion.

The Siebel Analytics Enterprise Data Warehouse has a predefined logic for loading the document, local, and group currency values into an IA table. The logic is shown in Figure 22.

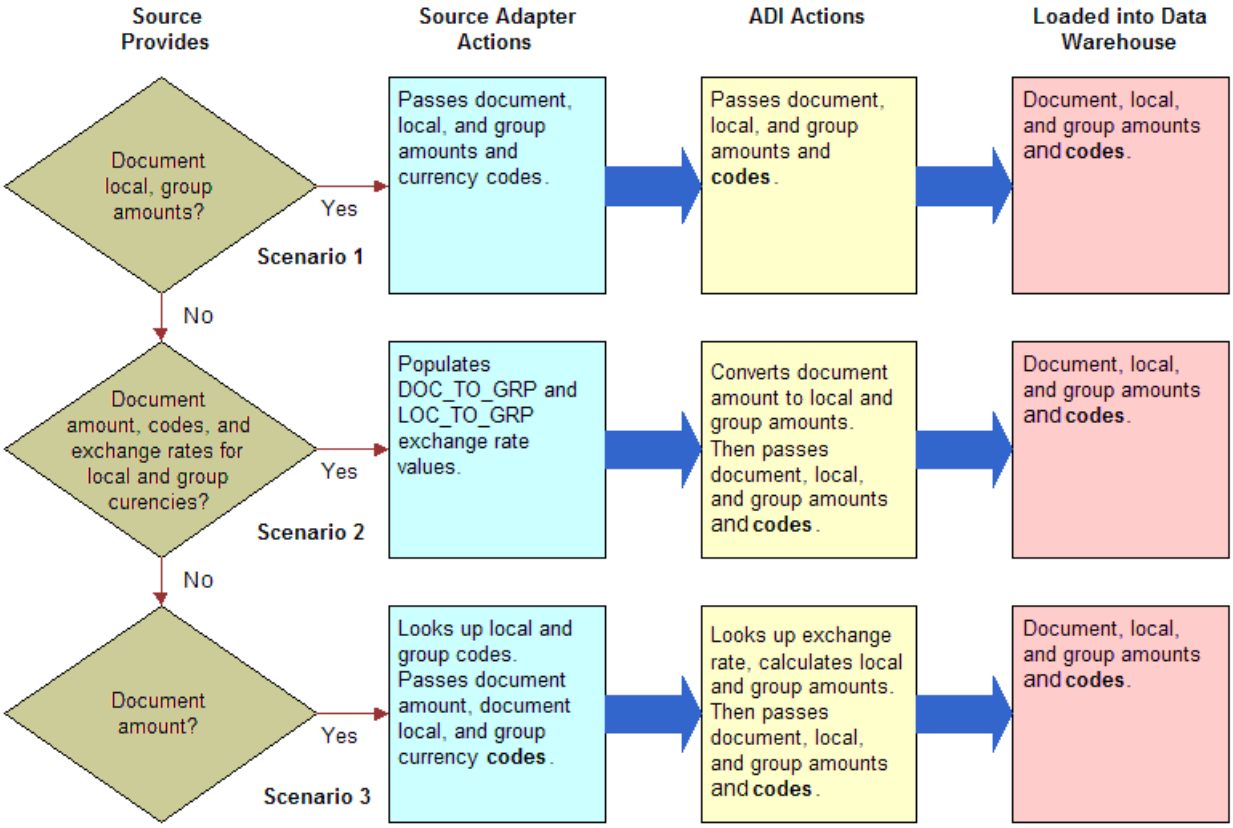


Figure 22. Currency Conversion Logic

Source Provides Three Currency Amounts

If the source system provides the amounts for the three currencies—document, local, and group. Each of these amounts are loaded into the data warehouse as provided.

Source Provides Document Amount, Codes, and Exchange Rates for Local and Group Currencies

The source system provides the following:

- Amount in document currency
- Local currency and group currency codes
- Exchange rates for local and group currencies (DOCUMENT_TO_GROUP and LOCAL_TO_GROUP exchange rates).

All of this information is fed to the ADI. The ADI uses the DOCUMENT_TO_GROUP rate to convert the document currency amount to the group currency amount and uses the ratio of DOCUMENT_TO_GROUP and LOCAL_TO_GROUP to convert the document currency amount to the local currency amount. The ADI then passes the three currency amounts and three currency codes to the corresponding IA table.

Source Provides the Document Currency Amount

This is the most common situation, and thus is the Siebel Analytics Enterprise Data Warehouse's default for handling currency. If the source system provides only the document currency amount, the Source Adapter performs lookups to identify the local and group currency codes based on the source system. Based on the source system the appropriate currencies are assigned. After the lookups occur, the Source Adapter mapplet provides the ADI with the document currency amount and the document, local, and group currency codes.

Because the ADI does not have the local and group amounts, or document-to-group (DOC_TO_GRP) and local-to-group (LOC_TO_GRP) currency conversion rates to derive the amounts, it looks up the exchange rates. By default, the ADI looks up the exchange rates in the IA_XRATES table. However, if you have custom tables that maintain exchange rate values, you can reconfigure the system to do the extraction from that source.

The ADI then populates the exchange rate values in the DOC_TO_GRP and LOC_TO_GRP variables. After this population is complete, the ADI uses the DOC_TO_GRP rate to convert the document currency amount to the group currency amount. The ADI then uses the ratio of DOC_TO_GRP and LOC_TO_GRP to convert the document currency amount to the local currency amount. Finally, the ADI passes the three currency amounts and currency codes to the corresponding IA table.

NOTE: For currency code information to be resolved, you must verify that the currency codes extracted map to the currency codes in the IA_XRATES table.

Configuring Currencies

Depending on how your organization handles currency, some of the following currency configuration points may apply to you. For example, you may find it necessary to reconfigure the group currency type, or you may need to reconfigure the way exchange rates are handled. The following sections provide procedures for configuration points related to these currency types.

Supplying Document-to-Group and Local-to-Group Currency Exchange Rates

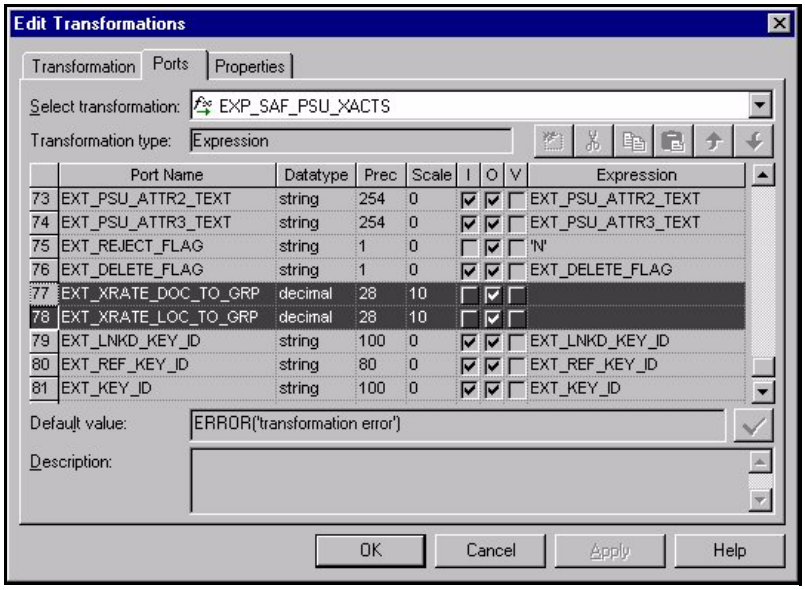
As mentioned in the previous section, the Siebel Analytics Enterprise Data Warehouse can present all amount values in three different currencies—document, local, and group. If all three amounts are not supplied by the source system, then the ADI can calculate the amounts using prepackaged exchange rate logic.

Usually the EXT_DOC_TO_GRP and EXT_LOC_TO_GRP values are null at the input of the ADI. Therefore, the ADI has to perform a lookup to retrieve the exchange rates. However, if you want to supply an exchange rate directly, you can do so by using the appropriate column. The EXT_XRATE_DOC_TO_GRP column should be supplied with the exchange rate that converts the document currency to group currency, and the EXT_XRATE_LOC_TO_GRP column should be supplied with the exchange rate that converts the local currency amount to the group currency amount. In this topic, you can find procedures for providing exchange rates.

To supply exchange rate values directly

1. In PowerCenter Designer, open the applicable source configuration folder.
2. In the MPLT_SA[Source Abbreviation]_XRATES Source Adapter mapplet, edit the expression for the applicable port.

   You can use the EXT_DOC_TO_GRP column to convert the document currency to the group currency. You can use the EXT_LOC_TO_GRP column to convert the local currency to the group currency, as shown in the following figure.



3. Validate and save your changes to the repository.

Extracting Exchange Rates from a Custom Table

If you do not maintain exchange rates in your ERP tables, but instead, you maintain them in your custom tables, then you can load them into the IA_XRATES table of the Siebel Analytics Enterprise Data Warehouse by creating a new mapping. The easiest way to create the mapping is to copy an existing exchange rate extraction and load mapping. After you copy it, you can modify the mapping to work with your custom tables.

To create new exchange rate extraction and load mappings

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Locate an existing exchange rate extract mapping.

   Copy the entire exchange rate mapping. Be sure to copy the entire set of objects—Business Components, and staging tables.
3. Reconfigure the Business Component to select the columns from your custom tables, instead of selecting them from the prepackaged tables.

   You can name and save this modified version of the Business Component as MPLT_BC1_XRATES.
4. Reconfigure the staging table to hold all the source related columns.

   You can name and save this staging table as T1_XRATES.
5. Reconfigure the extract mapping that uses the MPLT_BC1_XRATES as the source, and T1_XRATES as the target, and form the KEY_ID in the extract mapping.

   This mapping should be the unique record identifier from the source. However, it can be a combination of multiple columns from the source.
6. Locate an existing exchange rate load mapping and copy the entire exchange rate mapping.

   Be sure to copy the entire set of objects—Business Components, and staging tables.
7. Reconfigure the Source Adapter MPLT_SA1_XRATES mapplet to map the data in the staging table to the ADI format.
8. Reconfigure the load mapping M_1_XRATES_LOAD with the new Source Adapter mapplet.
9. Validate any expressions or SQL, and then save the changes to your repository.

Configuring the Exchange Rate Type

The column XRATE_TYPE_CODE in the table IA_XRATES identifies a specific type of exchange rate. Data loaded from various sources have different types of exchange rates, thus, there can be different values for the XRATE_TYPE_CODE column.

If an exchange rate is not supplied to convert an amount to a different currency, then the ADI performs a lookup to retrieve an exchange rate from the IA_XRATES table. The lookup that is retrieved is based on the specified exchange rate type. Each load mapping has the default exchange rate type 'M'. However, depending on the source system, the Siebel Analytics Enterprise Data Warehouse has a prepackaged SQL override in the session to extract a default exchange rate type that is applicable to the corresponding source. Thus, for Oracle 11i load sessions, there is a SQL override that requests the exchange rate type 'Corporate' instead of 'M'.

If you would like to use an exchange rate type other than the prepackaged override, then you must edit the SQL override in the applicable session.

To override the default exchange rate type

1. In PowerCenter Workflow Manager, open the source configuration folder.
2. Open the applicable session to open the Edit Tasks box.

3 In the Transformations tab, edit the SQL for the exchange rate lookups.

4 Change the exchange rate type by editing the following SQL statement fragment:

```
WHERE XRATE_TYPE = 'Corporate'
```

For a list of exchange rate type codes, see your source system's options.

5 Click OK.

NOTE: You need to perform these steps for each fact load for which you wish to configure the exchange rate information.

Configuring Currency Code for Oracle 11i

Oracle 11i provides document and local currencies, but not a group currency. As a result, you need to supply the currency code by means of a text file. For more information, see the discussion on configuring the group currency code in Configuration for Oracle 11i on page 279.

Handling European Monetary Union (EMU) Currencies

For transactions that take place after the Euro effective date (January 01, 1999), the published exchange rates involving *individual* EMU member currencies may not be available. To address the need for conversions between the individual currency of an EMU member and any other currency, you may need to set up a mechanism for retrieving these rates. The Siebel Analytics Enterprise Data Warehouse prepackages two methods for handling European Monetary Union currencies:

- Flat file (file_xrates_emu.csv)
- Expression transformation (EXP_CURR_CONVERSION_TRANSFORM).

Each of the two methods is discussed in the following sections.

Using the Flat File to Supply EMU Exchange Rates

The flat file contains currency conversion rates for all combinations of EMU currencies (including individual EMU member currencies as well as the joint EMU currency, the Euro). This file includes the following combinations:

- **EMU to EMU.** This is the conversion from the individual currency of one EMU member country to another.
- **EUR to EMU.** This is the conversion from the joint EMU currency, the Euro, to the individual currency of an EMU member.
- **EMU to EUR.** This is the conversion from the individual currency of an EMU member to the joint EMU currency, based on the Euro.

If you use this method, the Siebel Analytics Enterprise Data Warehouse assumes that the source supplies currency conversion rates between individual EMU member currencies and national currencies for countries outside of the EMU. In addition, this method does not conform to the rounding rules set by the European Monetary Union.

To use this method, you must upload the exchange rates from this flat file to the IA_XRATES table. After the exchanges rates are available in the IA_XRATES table, the ADI can look up these exchange rates from the IA_XRATES table and convert currency as required.

To load the individual EMU member currency exchange rates using the flat file

1. Before you load the exchange rates from the flat file, if necessary, change the default values in the flat file for the parameters SOURCE_ID, XRATE_TYPE_CODE, XRATES_TYPE_DESC, and KEY_ID.
   - The SOURCE_ID default is 'OAP11'. If your source is not Oracle, then you can reconfigure this ID.
   - The XRATE_TYPE_CODE and XRATE_TYPE_DESC default is 'Corporate'.
   - The KEY_ID uses the XRATE_TYPE_CODE as part of it definition. The Key ID is defined as the concatenation of the From Currency Code, To Currency Code, Exchange Rate Type Code, and the Effective From Date. Therefore, if you change the Exchange Rate Type Code, then you must also modify the Key ID accordingly.
2. Create the session for the M_F_XRATES_EXTRACT mapping, using the file_xrates_emu.csv flat file as the source file.
3. Run the extract session to load the flat file exchange rates into the TF_XRATES staging table.
4. Create a session for the M_F_XRATES_LOAD mapping.
5. Run the load session to load the exchange rates into the IA_XRATES table.

Using the EXP_CURR_CONVERSION_TRANSFORM Expression Transformation to Supply EMU Exchange Rates

The Expression transformation contains currency conversion logic for converting any combination of EMU currency (including individual EMU member currencies as well as the Euro) and other currencies (OTH) outside of the EMU. This logic applies to the following combinations:

- **EMU to EMU.** This is the conversion from the individual currency of one EMU member country to another.
- **EUR to EMU.** This is the conversion from the joint EMU currency, the Euro, to the individual currency of an EMU member.
- **EMU to EUR.** This is the conversion from the individual currency of an EMU member to the joint EMU currency, the Euro.
- **OTH to EUR.** This is the conversion from any other currency outside the EMU to the Euro.
- **EUR to OTH.** This is the conversion from the Euro to any other currency outside the EMU.
- **EMU to OTH.** This is the conversion from the individual currency of an EMU member country to another currency outside the EMU.
- **OTH to EMU.** This is the conversion from any other currency outside the EMU to the individual currency of an EMU member.

Unlike the flat file method, the transformation handles conversions between EMU currencies and currencies for countries outside the EMU. In addition, this method conforms to the rounding rules set by the European Monetary Union.

The transformation performs the conversion in different ways, depending on the types of input and output currencies. In the transformation, there are ten sets of amount input fields; each set consists of a document amount, local amount, and group amount field. In this discussion, document currency is referred to as the *From* currency and group and local currencies as the *To* currencies. Source-supplied document amounts are output as the same value. However, if the local and group amounts are not supplied, then a particular currency conversion process occurs. There are six different conversion processes that could occur. They are described in the list that follows:

- **Non-EMU to Non-EMU.** If the From Currency and To Currency are not EMU currencies, the following logic is used:
  - The supplied exchange rate is used to compute the new amount.

  OR

  - If the exchange rate is not supplied as an input to the transformation, the exchange rate is retrieved using a lookup to the IA_XRATES table. The new amount is then calculated using the looked up exchange rate and the From Currency amount.
- **EMU to EMU.** If the From Currency and To Currency are EMU currencies, the following logic is used:
  - Euro-triangulation is used to derive the new amount. Please note that the Euro-triangulation logic only applies if the exchange rate date is later than or equal to the Euro effective date.

  OR

  - If the exchange rate date is earlier than the Euro effective date, this case is treated as a Non EMU to Non EMU currency conversion case.

  The Euro-triangulation logic is used to convert one EMU currency to another EMU currency (EMU to EMU). Again, this logic is only applicable if the exchange rate date is equal to or later than the Euro effective date. The Euro-triangulation logic is as follows:
  - Convert from one national denomination (EMU_DOC) to its Euro equivalent (EUR). For example, EUR = EMU_DOC/(EUR-TO-EMU_DOC conversion rate).
  - Round the previous step's result to the nearest three decimal points. The default rounding is to three decimal points, but this is configurable.
  - Convert the Euro equivalent into the resulting national denomination (EMU_LOC). For example:

    EMU_LOC=(*Result from previous step*) * (EUR-TO-EMU_LOC conversion rate)
- **EMU to OTH.** If the From Currency is EMU and To Currency is OTH, the following logic is used in the given order:
  - The supplied exchange rate is used to compute the amount.

  OR

- If the exchange rate is not supplied and the NONEMU_EMU_TRI_FLAG = 'Y', then the two step conversion method is used to derive the amount. The two-step conversion method is defined as follows:
  - First, the From Currency amount is converted to the Euro Currency amount using the fixed EMU conversion rates supplied by the transformation's logic.
  - Second, the Euro Currency amount is converted to the To Currency amount using the appropriate exchange rate.

NOTE: The two-step conversion method is only used if the exchange rate date is greater than or equal to the Euro effective date and the NONEMU_EMU_TRI_FLAG is set to 'Y'.

OR

- If the NONEMU_EMU_TRI_FLAG is set to 'N', this is treated as a Non EMU to Non EMU currency conversion case.

- **OTH to EMU.** If the From Currency is OTH and the To Currency is EMU, then the following logic is used:
  - The supplied exchange rate is used to compute the To Currency amount.

  OR

  - If the exchange rate is not supplied and the NONEMU_EMU_TRI_FLAG = 'Y', then the two-step conversion method is used to derive the amount. The two-step conversion consists of the following two steps.
    - First, the From Currency amount (OTH) is converted to the Euro Currency amount using the exchange rate from the IA_XRATES table.
    - Second, the Euro Currency amount is converted to the EMU Currency amount using the fixed Euro conversion rate available in the transformation. Please note that the two-step conversion method only applies if the NONEMU_EMU_TRI_FLAG = 'Y'.

  OR

  - If the NONEMU_EMU_TRI_FLAG = 'N', this is treated as a Non EMU to Non EMU currency conversion case.
- **EUR to EMU.** If the From Currency is EUR and the To Currency is EMU, the following logic is used:
  - If the exchange rate date is greater than or equal to the Euro effective date, then the From Currency amount is converted to the To Currency amount using the fixed Euro conversion rates supplied by the transformation's logic.

  OR

  - In all other cases, this is treated as a Non-EMU to Non-EMU currency conversion case.
- **EMU to EUR.** If the From Currency is EMU and the To Currency is EUR, the following logic is used:
  - If the exchange rate date is greater than or equal to the Euro effective date, the From Currency amount is converted to the To Currency amount using the fixed Euro conversion rates supplied by the transformation's logic.

  OR

- If the exchange rate date is less than the Euro effective date, this is treated as a Non EMU to Non EMU currency conversion case.

To use an Expression transformation, you must add the transformation to every load mapping that requires any of the previously listed types of currency conversions. See the M_F_EURO_TRIANG_EXP_USAGE_EXAMPLE mapping for an example of how to incorporate an Expression transformation. This mapping is located in the Configuration for Universal Source folder.

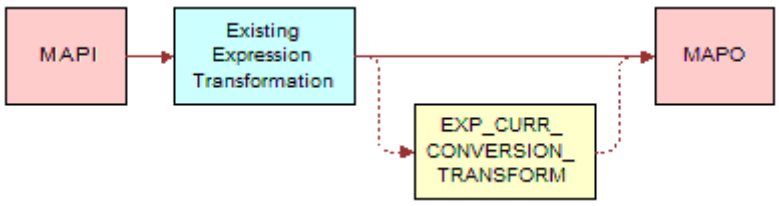
To add the transformation to a load mapping

1. Open the applicable source configuration folder.
2. Select Tools > Transformation Developer.
3. Open the Siebel Applications folder and drag and drop the EXP_CURR_CONVERSION_TRANSFORM Expression transformation to create a shortcut.

   You can place the shortcut in the Transformations folder contained in the applicable source configuration folder.
4. Select Mapplet Designer.
5. Open the applicable Source Adapter mapplet.
6. Add LKP_XRATES from the Transformations folder.

   NOTE: LKP_XRATES is a lookup to the IA_XRATES table. If the Transformations folder does not have a lookup to IA_XRATES, you must make a shortcut to LKP_XRATE in the Siebel Analytics Enterprise Applications folder.
7. Add the shortcut to the EXP_CURR_CONVERSION_TRANSFORM Expression transformation to the Source Adapter mapplet.

   Place the transformation between the load mapping's existing Expression transformation and MAPO, as shown in the following figure.



8. Reconnect the ports as necessary so that the data flows through the transformation and into the MAPO.

Table 54 lists all of the prepackaged exchange rates used by the transformation. Table 54 only lists exchange rates from EMU currency to Euro. Using these exchange rates, the transform can calculate any of the five scenarios (EMU to EMU, EUR to EMU, EMU to EUR, EMU to OTH, and OTH to EMU). For EMU to OTH, OTH to EMU, and EMU to EMU, the transformation converts the From Currency to the Euro and then from the Euro to the To Currency. For that reason, Table 54 only has EMU currency to the Euro conversion rates.

Table 54. Prepackaged Exchange Rates in Expression Transformation

Currency Code	Currency Name	Effective From Date	Conversion Rate for One Euro
BEF	Belgian Francs	01/01/1999	40.3399
DEM	German Marks	01/01/1999	1.95583
ESP	Spanish Pesetas	01/01/1999	166.386
FRF	French Francs	01/01/1999	6.55957
IEP	Irish Púnt	01/01/1999	.787564
ITL	Italian Lira	01/01/1999	1936.27
LUF	Luxembourg Francs	01/01/1999	40.3399
NLG	Dutch Guilders	01/01/1999	2.20371
ATS	Austrian Schillings	01/01/1999	13.7603
PTE	Portuguese Escudo	01/01/1999	200.482
FIM	Finnish Markka	01/01/1999	5.94573

You can incorporate additional EMU currency conversion rates into the transformation. When doing so, you must consider whether or not the conversion rate is effective before January 1, 1999 or on/after January 1, 1999. (January 1, 1999 is the date the Euro was enacted.)

To add a new EMU currency with the same effectivity date (January 1, 1999)

- Edit the decode statement for the columns EMU_TO_EURO_CONV_FACT_DOC, EMU_TO_EURO_CONV_FACT_LOC, and EMU_TO_EURO_CONV_FACT_GRP, to add the new currency and its conversion rate.

  For example, if you are adding EMU1, where the Euro-to-EMU1 conversion rate is 2.34567, then you would modify the EMU_TO_EURO_CONV_FACT_DOC column definition as follows:

```
EMU_TO_EURO_CONV_FACT_DOC = IIF (ISNULL (EXT_XRATE_LKP_DT) OR
TO_NUMBER(TO_CHAR(EXT_XRATE_LKP_DT,'J')) >= TO_NUMBER(TO_CHAR(TO_DATE('01/01/
1999', 'MM/DD/YYYY'),'J')), DECODE(EXT_DOC_CURR_CODE, 'BEF', 40.3399000000,
'NLG', 2.2037100000, 'ATS', 13.7603000000, 'PTE', 200.4820000000, 'FIM',
5.9457300000, 'IEP', 0.7875640000, 'LUF', 40.3399000000, 'FRF', 6.5595700000,
'ITL',1936.2700000000, 'DEM', 1.9558300000, 'ESP', 166.3860000000,
‘EMU1’,2.3456700000, 'EUR', 1.0000000000, NULL), NULL)
```

To add a new EMU currency with a different effectivity date than January 1, 1999

- Edit the decode statement for the columns EMU_TO_EURO_CONV_FACT_DOC, EMU_TO_EURO_CONV_FACT_LOC, and EMU_TO_EURO_CONV_FACT_GRP, to add the new currency, its conversion rate, and the effectivity date logic.

  For example, if you are adding EMU2, where the Euro-to-EMU2 conversion rate is 3.45678, and the exchange rate effectivity date is 01/01/2001, then you would modify the EMU_TO_EURO_CONV_FACT_DOC column definition as follows:

```
EMU_TO_EURO_CONV_FACT_DOC = IIF (ISNULL (EXT_XRATE_LKP_DT)
OR(TO_NUMBER(TO_CHAR(EXT_XRATE_LKP_DT,'J')) >= TO_NUMBER(TO_CHAR(TO_DATE('01/01/
1999', 'MM/DD/YYYY'),'J')) AND EXT_DOC_CURR_CODE!= 'EMU2') OR
TO_NUMBER(TO_CHAR(EXT_XRATE_LKP_DT,'J')) >= TO_NUMBER(TO_CHAR(TO_DATE('01/01/
2001', 'MM/DD/YYYY'),'J')),DECODE(EXT_DOC_CURR_CODE, 'BEF', 40.3399000000,
'NLG', 2.2037100000, 'ATS', 13.7603000000, 'PTE',200.4820000000, 'FIM',
5.9457300000, 'IEP', 0.7875640000, 'LUF', 40.3399000000, 'FRF', 6.5595700000,
'ITL',1936.2700000000, 'DEM', 1.9558300000, 'ESP',166.3860000000, 'EMU2',
3.4567800000, 'EUR', 1.0000000000, NULL), NULL)
```

Configuring Stored Lookups

A lookup transformation allows you to specify a reference table, and then retrieve information such as code descriptions, exchange rates, and currency codes. The main types of preconfigured lookups within the Siebel Analytics Enterprise Data Warehouse are:

- Codes Lookup on page 258.
- Dimension Key Lookups. For more information, see Resolving Dimension Keys on page 261.

Codes Lookup

Some source systems use intelligent codes that are intuitively descriptive, such as HD for hard disks, while other systems use nonintelligent codes (like numbers, or other vague descriptors), such as '16' for hard disks. While codes are an important tool with which to analyze information, the variety of codes and code descriptions used poses a problem when performing an analysis across source systems. The lack of uniformity in source system codes must be resolved to integrate data for the Siebel Analytics Enterprise Data Warehouse.

The code lookup in the ADI integrates both intelligent and nonintelligent codes by performing a separate extract for codes, and inserting the codes and their description into a codes table. The codes table provides the ADI with a resource from which it can automatically perform a lookup for code descriptions.

The Analytic Data Interface's architecture uses components, as well as both fact and dimension tables, to facilitate lookup functionality. The following components and process are used in a lookup:

IA_CODES Table

The load control table IA_CODES consolidates all codes for future reference and assigns them a category and a single language for efficient lookup capability.

Codes Mappings

The Siebel Analytics Enterprise Data Warehouse uses mappings designed to extract codes from source systems and populate the IA_CODES table in preparation for use by the ADI.

To understand how codes mappings function, it is helpful to first understand the columns within IA_CODES. Table 55 describes these columns.

Table 55. Columns in Code Mapplet

Column	Description
SOURCE_ID	Unique identifier of the source system from which data was extracted
SOURCE_CODE1	The first code in the hierarchy of the various source codes used to identify a particular code and description combinations
SOURCE_CODE2	The second code in the hierarchy of the various source codes used to identify a particular code and description combinations
SOURCE_CODE3	The third code in the hierarchy of the various source codes used to identify a particular code and description combinations
SOURCE_DESC_1	Short description of the Source code
SOURCE_DESC_2	Long description for code

The naming convention for mappings designed for codes lookup is M_[SOURCE]_CODES_[CATEGORY]. Figure 23 shows an example of a code mapping in PowerCenter Mapping Designer.

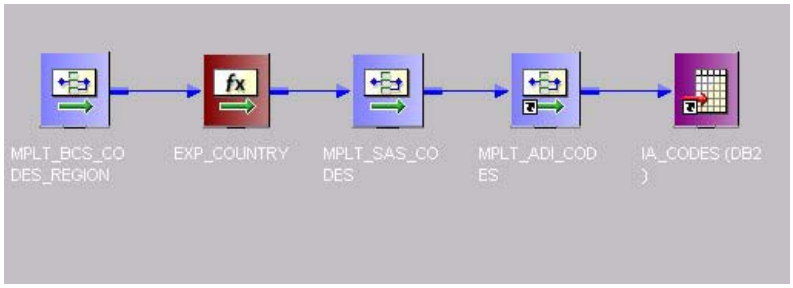


Figure 23. Sample Codes Mapping: M_S_CODES_REGION

Codes Mapplets

There are several mapplets that support the codes mappings in preparation for the source-independent ADI. They are as follows:

- **Source Adapter mapplets.** The Source Adapter mapplet connects the source-specific input attributes of CODES and the attributes from control or warehouse tables to the expression transform for mapping them. The naming convention for the Source Adapter codes mapplet is MPLT_SA[Source Abbreviation]_CODES.
- **Business Component mapplets.** The Business Component mapplet makes the source attributes of CODES_CUST_CLASS available to the extract mapping. The naming convention for the Business Component codes mapplet is MPLT_BC[Source Abbreviation]_CODES_[CATEGORY].

- **ADI Mapplet.** The Analytic Data Interface (ADI) mapplet is source independent and resolves the codes for the target table. The naming convention for the ADI codes mapplet is `MPLT_ADI_CODES`.

The ADI integrates multiple source codes by designating one source instance as a master in a mapping. All other source codes are then mapped to the master. When the ADI encounters a code that requires definition, it references the load control lookup table to match the source code to a Siebel Analytics Enterprise Data Warehouse source-independent code, which retains all the source codes' original functionality.

The following columns are used to designate a source instance as the master source:

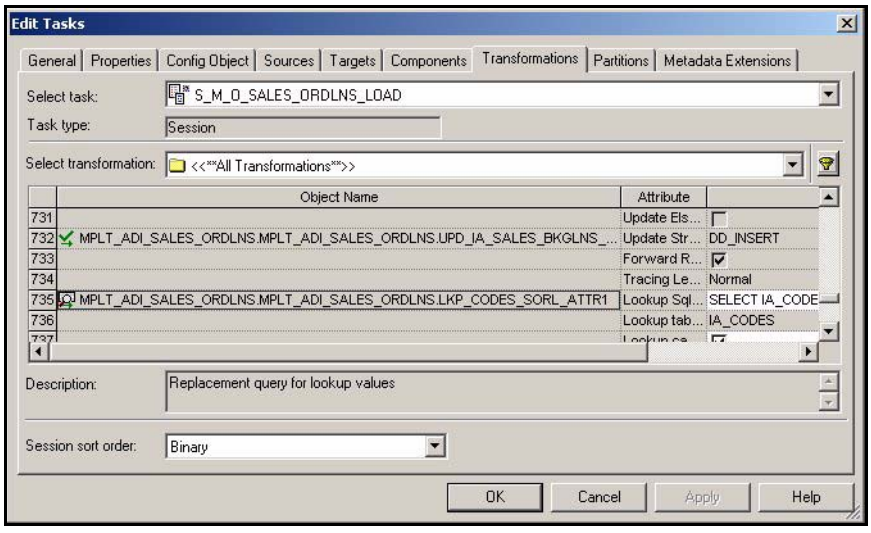
- **`MASTER_ID`.** Code for the source system designated as master.
- **`SOURCE_ID`.** Unique identifier for the source system.

Configuring Extension Column Code Description Lookups

You can configure dimension and fact load sessions to perform specific lookups by editing the category of the data to be extracted from the `IA_CODES` table and loading the code information into a target table. If the code and code name do not exist in the `IA_CODES` table, then you must add them to the table. To configure the lookup, create a session override; do not modify the ADI in the load mapping.

To configure sessions for lookups

1. In PowerCenter Workflow Manager, open the applicable source configuration folder.
2. Open the Edit Tasks box, as shown in the following figure.



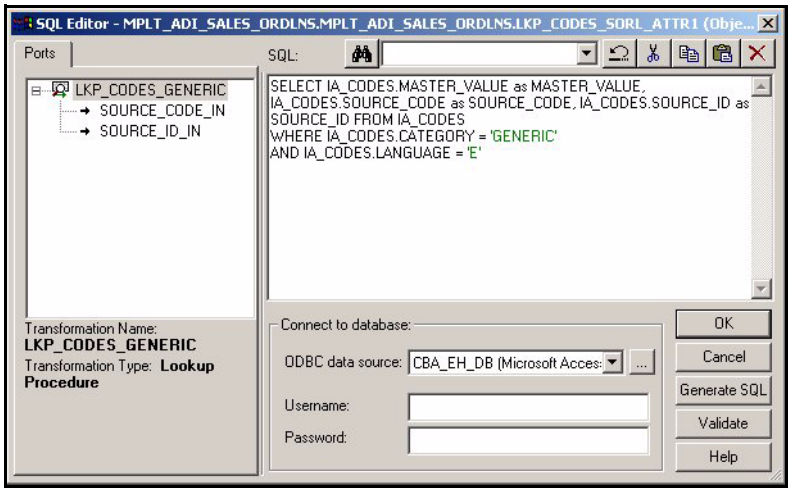
3. In the Transformations tab, edit the SQL for the lookup.

   For example, you may wish to edit the following lookup:

MPLT_ADI_SUPPLIERS.LKP_SPLR_ATTR1

4 Edit the SQL statement to use the desired code category.

5 Edit the SQL statement from 'GENERIC' to the category you wish to use for the lookup, as shown in the following figure.



Resolving Dimension Keys

By default, dimension key resolution is performed by the Siebel Analytics Enterprise Data Warehouse in the ADI. The ADI uses prepackaged, reusable lookup transforms to provide prepackaged dimension key resolution. This section describes how dimension keys are looked up and resolved.

There are two commonly used methods for resolving dimension keys. The first method, which is the primary method used, is to perform a lookup for the dimension key. The second method is to supply the dimension key directly into the fact load mapping.

Dimension Key Resolution Using a Lookup

If the dimension key is not provided to the ADI through database joins, the ADI performs the lookup in the dimension table. The ADI does this using prepackaged lookup transformations.

The ADI uses the Dimension Key ID, the Source ID and Lookup date in looking up the dimension key. All these columns are necessary for the ADI to return the dimension key. The ports are described in Table 56.

Table 56. Columns Used in the ADI Dimension Key Lookup

Port	Description
Key ID	Uniquely identifies the dimension entity within its source. Formed from the transaction in the Source Adapter of the fact table.
Source ID	Unique identifier of the source instance.
Lookup Date	The primary date of the transaction; for example, receipt date, sales date, and so on.

In Figure 24, the Supplier Products Key Lookup transformation illustrates the three input columns needed for the ADI lookup—the Key ID, Source ID, and Date (lookup date). The transformation then outputs the Supplier Product key (the dimension key) to the data warehouse table IA_SPLR_PRODS.

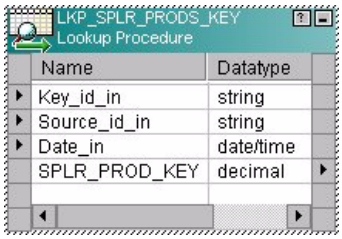


Figure 24. Example of Input Columns Required for the ADI

If Type II slowly changing dimensions are enabled, the ADI uses the unique effective dates for each update of the dimension records. When a dimension key is looked up, it uses the fact's primary date to resolve the appropriate dimension key.

The effective date range gives the effective period for the dimension record. The same entity can have multiple records in the dimension table with different effective periods due to Type II slowly changing dimensions. This effective date range is used to exactly identify a record in its dimension, representing the information in a historically accurate manner. In the lookup for Employee Contract Data shown in Figure 25, you can see the effective dates used to provide the effective period of employee contracts.

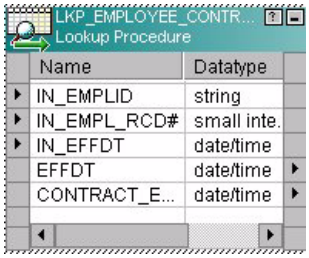


Figure 25. Effective Dates Used in Slowly Changing Dimension

Configuring the Dimension Key ID

There are many ways in which you can configure Dimension Key IDs. This section provides procedures on configuring dimension keys so that your dimension tables load data at an appropriate granular level and, also so that your keys get resolved.

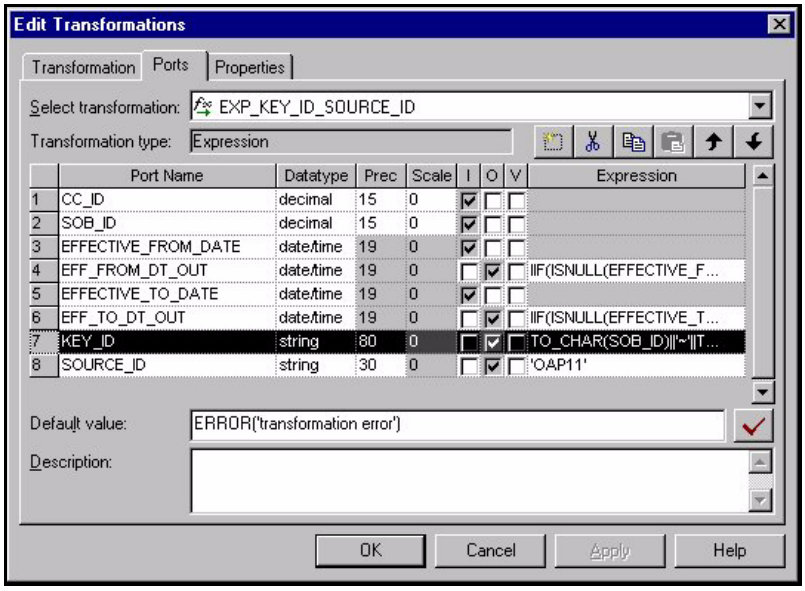
Each Dimension Key ID has a default value, which can be configured. If you want to reset the value of a Dimension Key ID, you must modify the Key ID definition in the dimension extract mapping, in the Expression transformation, as well as in every fact load that uses the key. For example, if you want to modify the Key ID for the IA_GL_ACCOUNTS dimension table, then you must modify the Key ID's definition in the IA_GL_ACCOUNTS extract mapping's Expression transformation.

In addition, you have to modify any fact table load mapping that uses the key. For example, because the IA_SALES_IVCLNS fact table uses the Key ID of the IA_GL_ACCOUNTS dimension table, you must modify the Key ID definition in the IA_SALES_IVCLNS load mapping's Source Adapter mapplet. The following two procedures tell you how to accomplish both of these tasks.

To configure the Key ID in the dimension extract mapping

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable extract mapping.

3 Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4 Edit the expression for the KEY_ID column.

For example, if you redefine the KEY_ID column for the IA_GL_ACCOUNTS table, modify the default Key ID port in the M_I_GL_ACCOUNTS_EXTRACT mapping, which, by default, is set to the Set-of-Books ID ~ Code Combination ID (TO_CHAR(SOB_ID)||'~'||TO_CHAR(CC_ID)). You can reset the grain of this dimension table by setting the Key ID to something else, like Set-of-Books ID ~ GL account number.

NOTE: Verify that any modified Key ID continues to uniquely identify each record in the table.

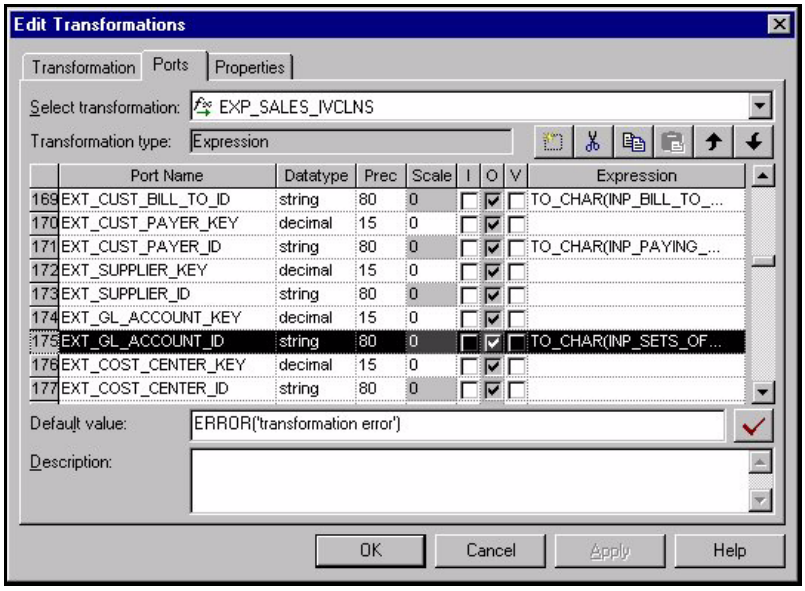
5 Validate and save your changes to the repository.

To configure the Key ID in the fact load mapping

1 In PowerCenter Designer, open the applicable source configuration folder.

2 Open the applicable fact mapping's Source Adapter mapplet.

3 Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4 Edit the expression for the *_ID column.

For example, if you want to redefine the IA_GL_ACCOUNTS Key ID in the IA_SALES_IVCLNS table, modify the default Key ID as shown in the following:

```
TO_CHAR(INP_SETS_OF_BOOK_ID)||'~'||

TO_CHAR(INP_CODE_COMBINATION_ID)
```

5 Validate and save your changes to the repository.

6 Repeat these steps for every fact table that is joined to the dimension in question.

Configuring Lookup to Additional or Custom Tables

The predefined extension dimension keys in the fact tables point to the IA_DIMENSION table. If you have custom dimension tables to add, or want to join additional dimension tables to a Siebel Analytics Enterprise Data Warehouse fact table, you can link the dimension tables to the fact tables by modifying the SQL statement for the extension dimension keys in the fact tables.

Each fact table has at least three extension dimension keys, allowing you to store additional dimension tables. To join a new dimension table to a fact table, you need to modify the fact table's load mapping or session, which involves two tasks:

- Define the Key ID in the Source Adapter mapplet of the fact table load.
- Modify the session to perform a SQL override for the lookup that is used to resolve the Key ID and redirect the lookup to the dimension table of your choice.

To reset the Dimension Key ID in the Source Adapter mapplet

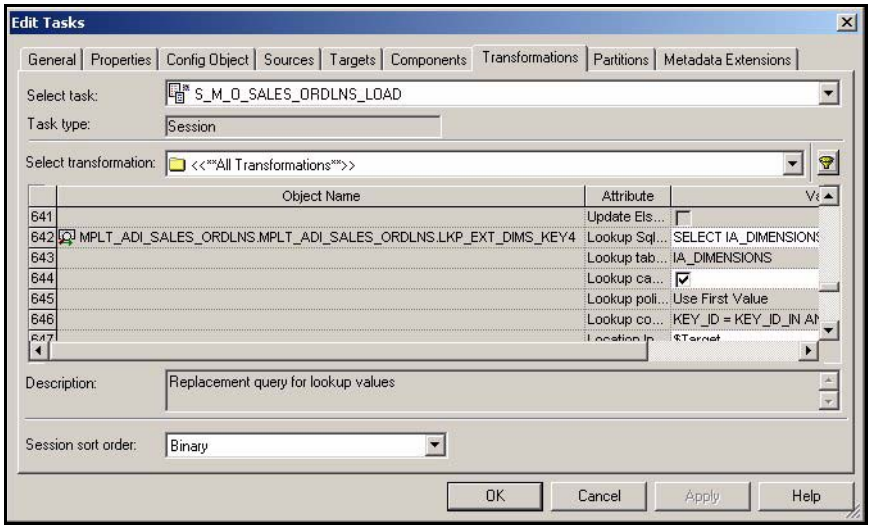
1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable fact load mapping's Source Adapter mapplet.
3. Double-click the Expression transformation to open the Edit Transformations box.
4. Edit the EXT_[SUBJECT]_DIM*X*_KEY port.

   For example, if you want to join the IA_GL_ACCOUNTS dimension table to the IA_SALES_IVCLNS fact table, you could join it to the EXT_SIVL_DIM1_ID.

   NOTE: Make sure that the level at which you define the Dimension Key ID in the fact mapping is the same grain at which the Key ID is defined in the dimension table's extract mapping.
5. Validate and save your changes to the repository.

To resolve the extension column Dimension Key ID in the fact load

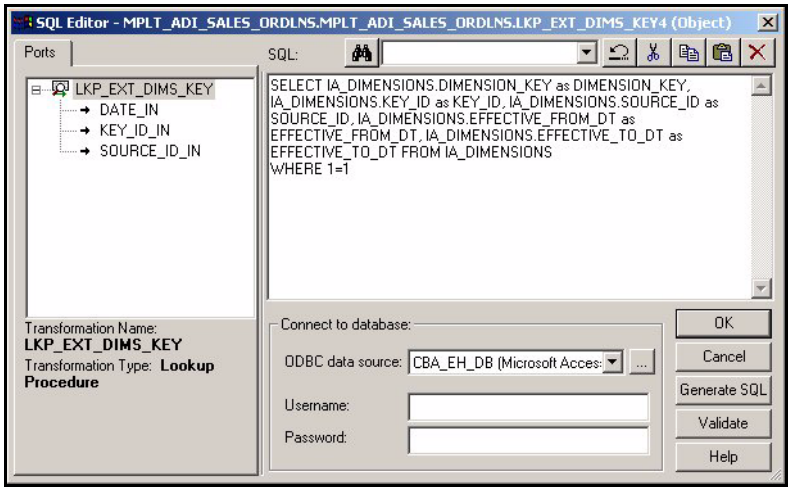
1. In PowerCenter Workflow Manager, open the applicable source configuration folder.
2. Double-click the applicable session for the fact load mapping to open the Edit Tasks box, as shown in the following figure.



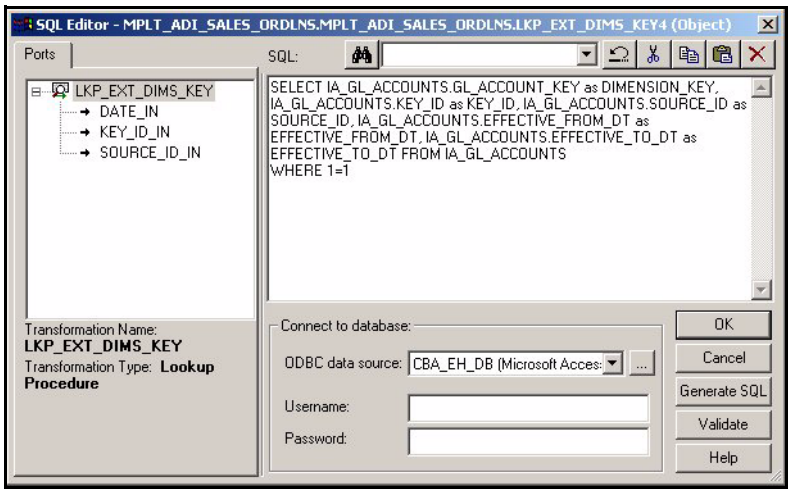
3. In the Transformations tab, edit the Lookup SQL Override field for the dimension key lookup in the ADI mapplet.

4 By default, the lookup points to the IA_DIMENSIONS table.

Point the lookup to the new dimension table, as shown in the following figure.



If you are joining the IA_GL_ACCOUNTS table, then you would change the references from IA_DIMENSIONS to IA_GL_ACCOUNTS, as shown in the following figure.



5 Click OK, and then click OK to exit the Edit Tasks dialog box.

Dimension Key Resolution Using Database Joins

The ADI provides lookups for dimension keys by default. However, if the dimension table is large, you can provide the dimension key to the ADI by joining the dimension table and the fact table in the database.

If you supply the key, then the ADI does not perform the lookup and instead resolves the dimension key within the load mapping itself. In this case, you modify the SQL in PowerCenter to join the tables.

To load a dimension key using a database join

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Open the applicable load mapping.
3. Add the dimension table (the table that contains the dimension key) as a source to your load mapping.
4. Drag and drop the surrogate key column from the dimension source definition to the Source Qualifier.
5. Double-click the Source Qualifier transformation to open the Edit Transformations box.
6. In the Properties tab, edit the SQL statement to put in the join conditions between the dimension table and the fact extract table.
7. Drag and drop the surrogate key column from the Source Qualifier to an available EXT_*_KEY port in the Source Adapter mapplet.
8. Validate and save your changes to the repository.

In Figure 26, the dimension key resolution is performed in the Customer Dimension table in the database by joining the Customer Dimension table to the Sales Orders extract table. The dimension key is then passed to the ADI, and is then loaded into the Sales Orders Fact table.

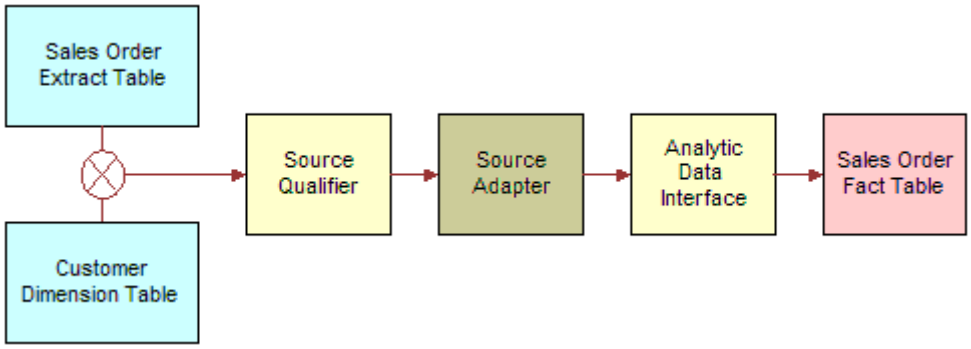


Figure 26. Dimension Key Resolution

Working with Domain Values

The Siebel Analytics Enterprise Data Warehouse foundation comprises a data model that accommodates data from disparate source systems. Data is sourced from operational systems and systematically molded into a source-independent format. After the data is source independent, it can then be used to create key metrics for analytic reporting, so that metric calculations are not source dependent. This clear separation allows you to swap source systems or integrate additional source systems without having to reconfigure the metric calculations to accommodate each source system's requirements.

One method for transforming source data into a source-independent format is to convert the source-supplied values to domain values. Domain values are a set of distinct values used to calculate prepackaged metrics. These values are provided by the Siebel Analytics Enterprise Data Warehouse to allow you to create metric calculations independent of source system values.

Understanding the Domain Value Conversion Process

To best understand the domain value conversion process, consider an example of two source systems—Source System A and Source System B. Each source system stores two types of employee events—hire and rehire. Source system A uses 'H' to denote a hire event and 'R' to denote a rehire event, whereas source system B uses '1' to denote a hire event and '2' to denote a rehire event. When the Siebel Analytics Enterprise Data Warehouse extracts data from both systems, it ports those source values through the extract mapping until the data reaches the IA_EVENT_GRP_CODE column in the TF_EVENT_TYPES Staging table.

The load mapping then ports the extracted source values ('H' and 'R' from source system A, and '1' and '2' from source system B) into the Source Adapter mapplet. Within the Source Adapter, source values are translated into domain values ('HIR' and 'REH') based on a set of rules that are particular to your business practices.

You must define the rules so that the Source Adapter knows how to map your specific source values to the given set of domain values. Before you set up the rules, you must first analyze all of your source values and how they should map to the prepackaged domain values. You may find that you need to create additional domain values for particular columns. The result of this preparation work should be a list of each source value and how it should be mapped to a domain value. After you have this information, you can implement this logic in the applicable Source Adapter mapplet. To set up the logic, modify the Expression transformation in the Source Adapter mapplet for each affected column. For information on setting up the rules for domain values, see Configuring the Domain Value Set on page 273. Figure 27 illustrates how the source values are converted to the domain values—'HIR' and 'REH'.

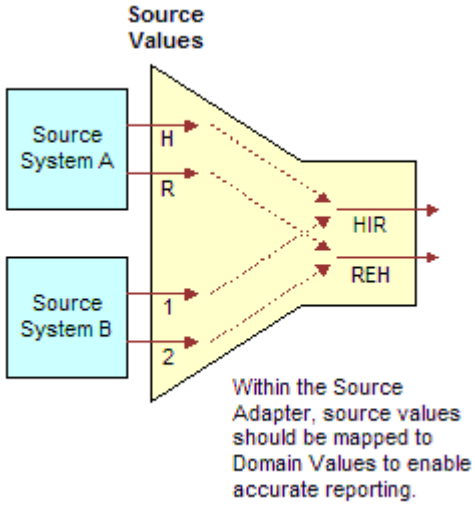


Figure 27. Source Values Translated to Domain Values

Figure 28 illustrates a different situation where the records may not contain a source value that flags the record as Hire or Rehire. In this case, the source system stores hires in one table and rehires in another table. To make this work, one possible solution is to modify the extract mappings to populate the IA_EVENT_GRP_CODE column with 'HIR' or 'REH'. If the field is populated in the extract mapping, you can then carry those same values through the Source Adapter mapplet.

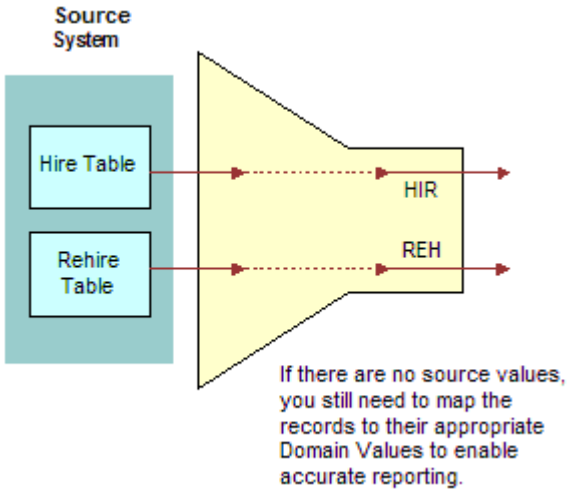


Figure 28. Records Mapped to Domain Values

After the Source Adapter mapplet converts the source-specific values to domain values, the domain values are inserted into a Siebel Analytics Enterprise Data Warehouse table. In this example, the 'HIR' and 'REH' values populate the IA_EVENT_TYPES table, as illustrated in Figure 29.

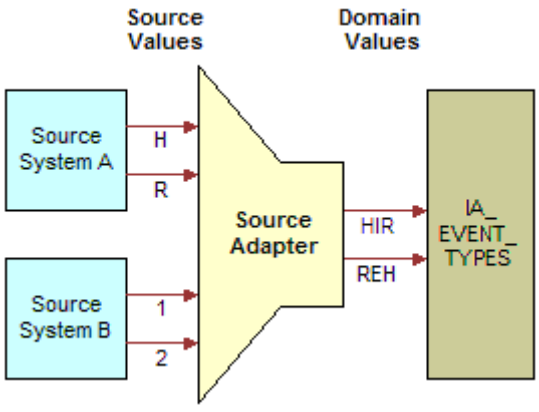


Figure 29. Domain Value Loading Siebel Analytics Enterprise Data Warehouse Table

Understanding the Importance of Domain Values

Values in the IA_EVENT_TYPES table are used to create metrics in the front end. Some metrics are defined using domain values. For example, seven metrics use the 'HIR' and 'REH' event group code in their calculation. The following lists these seven metrics, along with their descriptions and calculations.

Hire Count

This metric counts all hires for a specified period. The calculation is:

```
SUM(CASE WHEN (CMMNEVTP.IA_EVENT_GRP_CODE IN ('HIR','REH')) THEN EVNT.EVENT_CNT ELSE
0 END)
```

Rehires Ratio

This metric determines the ratio of rehires to all employees hired during a specified period. The calculation is:

```
CASE WHEN SUM(CASE WHEN CMMNEVTP.IA_EVENT_GRP_CODE IN ('REH','HIR') THEN
EVNT.EVENT_CNT ELSE 0 END) = 0 THEN 0 ELSE SUM(CASE WHEN CMMNEVTP.IA_EVENT_GRP_CODE
IN ('REH') THEN EVNT.EVENT_CNT ELSE 0 END)/SUM(CASE WHEN CMMNEVTP.IA_EVENT_GRP_CODE
IN ('REH','HIR') THEN EVNT.EVENT_CNT ELSE 0 END) END
```

New Hire Count

This metric counts the headcount hired for regular full-time positions. The calculation is:

```
SUM(CASE WHEN CMMNEMPT.FULL_TIME_FLAG = 'Y' AND CMMNEMPT.EMP_CAT_CODE = 'R' AND
(CMMNEVTP.IA_EVENT_GRP_CODE = 'HIR' OR CMMNEVTP.IA_EVENT_GRP_CODE = 'REH') AND
EVNT.EVENT_DK >= (CMMNDATE.DATE_KEY - 365) AND EVNT.EVENT_DK <= CMMNDATE.DATE_KEY
THEN EVNT.EVENT_CNT ELSE 0 END)
```

Newly Separated Veterans - New Hires

This metric counts the regular full-time and part-time employees who belong to this category of veterans and were hired during the previous 12 months. The calculation is:

```
SUM(CASE WHEN CMMNEMPD.VETERAN_STAT_CODE = '4' AND CMMNEMPT.EMP_CAT_CODE = 'R' AND
(CMMNEVTP.IA_EVENT_GRP_CODE = 'HIR' OR CMMNEVTP.IA_EVENT_GRP_CODE = 'REH') AND
EVNT.EVENT_DK >= (CMMNDATE.DATE_KEY - 365) AND EVNT.EVENT_DK <= CMMNDATE.DATE_KEY
THEN EVNT.EVENT_CNT ELSE 0 END)
```

Other Protected Veterans - New Hires

This metric counts regular full-time and part-time employees who belong to this category of veterans. The calculation is:

```
SUM(CASE WHEN CMMNEMPD.VETERAN_STAT_CODE = '3' AND CMMNEMPT.EMP_CAT_CODE = 'R' AND
(CMMNEVTP.IA_EVENT_GRP_CODE = 'HIR' OR CMMNEVTP.IA_EVENT_GRP_CODE = 'REH') AND
EVNT.EVENT_DK >= (CMMNDATE.DATE_KEY - 365) AND EVNT.EVENT_DK <= CMMNDATE.DATE_KEY
THEN EVNT.EVENT_CNT ELSE 0 END)
```

Special Disabled Veteran Headcount - New Hires

This metric counts regular full-time and part-time employees who belong to this category of veterans and were hired during the previous 12 months. The calculation is:

```
SUM(CASE WHEN CMMNEMPD.VETERAN_STAT_CODE = '1' AND CMMNEMPT.EMP_CAT_CODE = 'R' AND
(CMMNEVTP.IA_EVENT_GRP_CODE = 'HIR' OR CMMNEVTP.IA_EVENT_GRP_CODE = 'REH') AND
EVNT.EVENT_DK >= (CMMNDATE.DATE_KEY - 365) AND EVNT.EVENT_DK <= CMMNDATE.DATE_KEY
THEN EVNT.EVENT_CNT ELSE 0 END)
```

Vietnam Era Veteran Headcount - New Hires

This metric counts regular full-time and part-time employees who belong to this category of veterans and were hired during the previous 12 months. the calculation is:

```
SUM(CASE WHEN CMMNEMPD.VETERAN_STAT_CODE = '2' AND CMMNEMPT.EMP_CAT_CODE = 'R' AND
(CMMNEVTP.IA_EVENT_GRP_CODE = 'HIR' OR CMMNEVTP.IA_EVENT_GRP_CODE = 'REH') AND
EVNT.EVENT_DK >= (CMMNDATE.DATE_KEY - 365) AND EVNT.EVENT_DK <= CMMNDATE.DATE_KEY
THEN EVNT.EVENT_CNT ELSE 0 END)
```

Each of these metric calculations is based on the domain values 'HIR' and 'REH'. All records whose source values are converted to one of these domain values are included in the metric calculations, as shown in Figure 30.

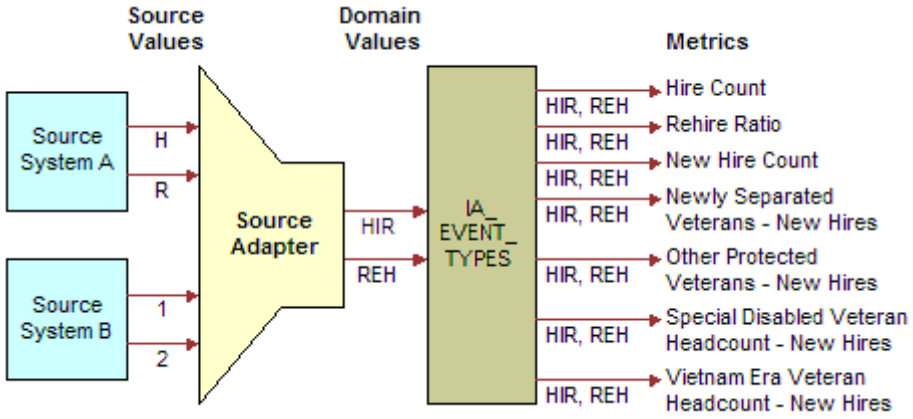


Figure 30. Domain Values Used in Metric Definitions

Extending the Domain Value Set

The Siebel Analytics Enterprise Data Warehouse is also extensible in that you can create additional domain values for those columns that do not fit into the existing domain value definitions. To see which domain value sets can be modified, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*. However, before you modify the domain value set for a particular column, you should first perform impact analysis on existing metrics. For example, the Siebel Analytics Enterprise Data Warehouse prepackages the following two events:

- **New Hire.** This event occurs when a new person is hired.

- **New Position.** This event occurs when a position is created, but an existing employee may be hired internally.

If you have an event that represents both a New Hire and a New Position, you may have to create a third event that depicts both. If you create this new event type domain value, you should include it in the applicable metric definitions so as to account for all hires and positions.

Configuring the Domain Value Set

Configuring the domain value set for a particular column entails one or both of the following activities:

- Mapping source-specific values to domain values
- Adding more domain values to the prepackaged set of values

Regardless of which activity you choose, the configuration occurs in the Expression transformation of the applicable Source Adapter mapplet. The following procedure shows how to configure the Expression transformation to change the domain values.

To map source values to domain values

1. Identify all the Siebel Analytics Enterprise Data Warehouse table columns that use domain values.

   For a list of columns that use domain values, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

2. List all of your source values that qualify for conversion to one of the domain values.
3. Map each source value to a domain value.

   If any of your source values do not map to a prepackaged domain value, and the list of domain values can be modified, then create a list of new domain values and map your orphaned source values to your newly created domain values.

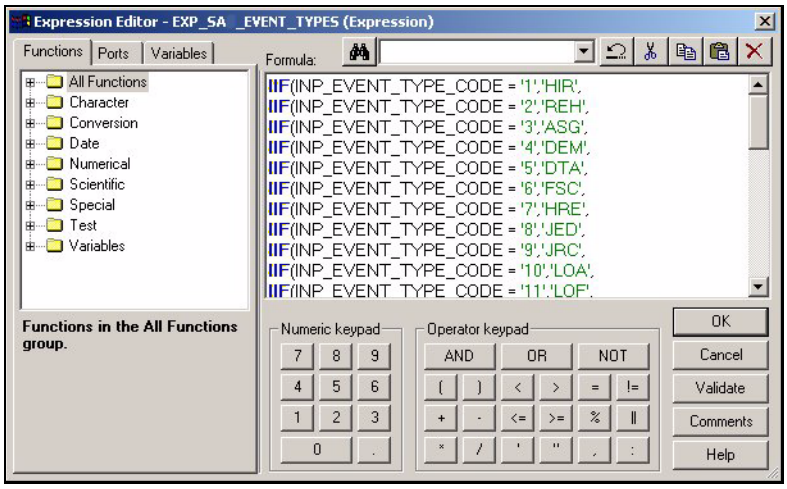
   Not all domain value sets can be modified. Also, you must check which metrics are affected by the modified domain value set. For more information, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

4. In PowerCenter Designer, open the applicable Source Adapter mapplet.
5. Open the Expression transformation.
6. Locate the applicable port's expression so that you can modify it.

7 Edit the port's expression to map your source values to the existing domain values.

Alternately, if you want to add additional domain values, add them in this same expression.

For example, if you wanted to map '1' and '2' to domain values 'HIR' and 'REH' for the IA_EVENT_GRP_CODE column of the IA_EVENT_TYPES table, then you would modify or create the highlighted expression, as shown in the following figure.



8 Save and validate your changes to the repository.

Configuring Conformed Dimensions

This section provides procedures on configuring objects that apply to more than one module.

Configuring Conformed Dimensions for Universal Source

This section provides configuration procedures for modifying dimensions that are loaded using a universal business adapter.

Product Effective Dates in the Products Dimension

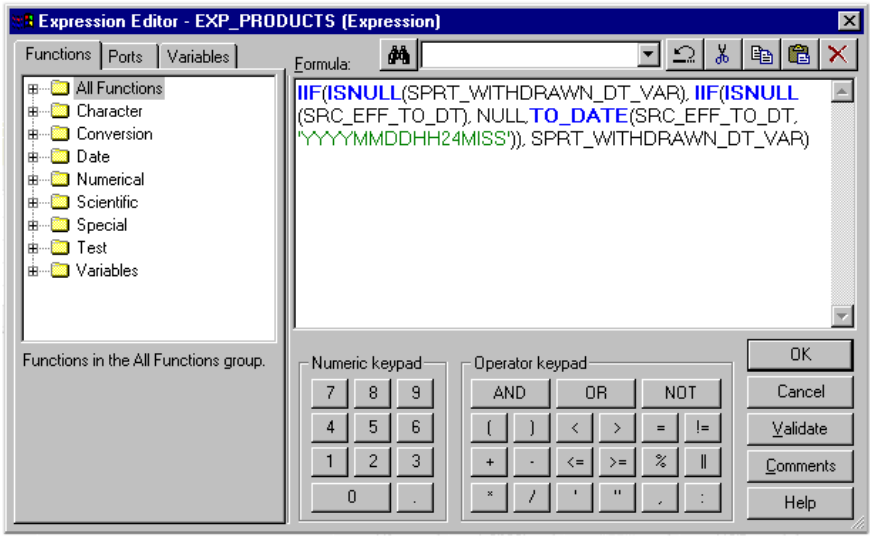
The Siebel Analytics Enterprise Data Warehouse stores product Effective To (SRC_EFF_TO_DT) and From (SRC_EFF_FROM_DT) dates in the Products dimension table, IA_PRODUCTS. In addition, the Products dimension stores a Support Withdrawn Date column, SPRT_WITHDRAWN_DT.

By default, the Support Withdrawn Date takes precedence over the product Effective To Date. This prioritization means that if you supply a value for the Support Withdrawn Date column in your flat file upload, the Siebel Analytics Enterprise Data Warehouse uses that value as the product Effective To value as well, overwriting anything in the SRC_EFF_TO_DT column. You can change this default behavior by modifying the Products Expression in the Universal Source Products Extract mapping.

To modify the product Effective To Date logic for a flat file extract

1. In PowerCenter Designer, open the Configuration for Universal Source folder.
2. In the M_F_PRODUCTS_EXTRACT mapping, open the EXP_PRODUCTS expression.
3. Edit the logic for the SRC_EFF_TO_DT_OUT port.

   The default logic is shown in the following figure.



4. Validate and save your changes.

Defining Services at the Package or Component Level

The Service Provisions dimension, IA_SRVC_PRVSNS, stores information about service or account packages, as well as the components that comprise these packages. When the Siebel Analytics Enterprise Data Warehouse reports your product or service count, by default, they report it at the lower level of granularity—the individual service level. Therefore, if you have two service packages, Package1 and Package2, each consisting of ten components, your product count is twenty (at the component level), rather than only two (at the package level). You can change this logic to the higher level of granularity—the service package level.

To configure the granularity of the your services definition, use the Package Product ID instead of the Component Product ID to populate the Product ID column of your Service Provisions dimension.

To define the granularity of the Service Provisions dimension

1. Configure the contents of the FILE_SRVC_PRVSNS.csv flat file that is used in the M_F_SRVC_PRVSNS_EXTRACT mapping.

   The M_F_SRVC_PRVSNS_EXTRACT mapping can be found in the Configuration for Universal Source folder in PowerCenter Designer.

2 If you want the Service Provisions dimension to be configured at the package level of granularity, use the value of the PACKAGE_PROD_ID to populate the PRODUCT_ID column.

Cross-Referencing Entities in the Siebel Analytics Enterprise Data Warehouse

The Siebel Analytics Enterprise Data Warehouse contains several dimension tables that store information about *entities,* which are (companies and individuals who come in contact with your business). Because one entity can play multiple roles in your business, the Siebel Analytics Enterprise Data Warehouse includes a cross-reference table, IA_XRF_ENTITIES, to maintain cross-reference information.

For example, assume that a vendor, who is defined in your Suppliers dimension table, IA_SUPPLIERS, also plays the role of a customer (and is therefore also defined in your Customers dimension table, IA_CUSTOMERS). The multiple roles of this customer or vendor would be tracked in the Cross Reference Entities table, IA_XRF_ENTITIES.

Table 57 provides the columns in IA_XRF_ENTITIES that are updated by means of a flat files. As new references come in from a flat file, the ENTITY_KEY column is updated, while the ORIG_ENTITY_KEY column retains its original value.

Table 57. Cross Reference Entities Table: IA_XRF_ENTITIES

IA_XRF_ENTITIES		
DIM_KEY_ID	**ENTITY_KEY**	**ORIG_ENTITY_KEY**
Customer1	Entity1	Entity1
Supplier1	Entity2	Entity2
Employee1	Entity3	Entity3
WebVisitor1	Entity4	Entity4

Table 58 depicts the flat file columns that are used to update the IA_XRF_ENTITIES table. The sample shows that Supplier1 is the same entity as Customer1, and that the same is true for Employee1, indicating three separate roles for one individual. The sample flat file is structured correctly because the Reference Dimension ID column uses Customer as its reference all three times. If there were an additional record with EmployeeX as the Base Dimension and SupplierY as the Reference Dimension, that record would go into a separate file for upload so that every uploaded file uses only one dimension in its Reference Dimension ID column. This consistency allows the IA_XRF_ENTITIES table to update correctly.

Table 58. Flat File to Update the Cross-Reference Entities Table IA_XRF_ENTITIES

Flat File Source Base Dimension ID	Reference Dimension ID
Supplier1	Customer1

Table 58. Flat File to Update the Cross-Reference Entities Table IA_XRF_ENTITIES

Flat File Source Base Dimension ID	Reference Dimension ID
Employee1	Customer1
WebVisitor7	Customer23

Limiting the Entities Cross-Referenced in IA_XRF_ENTITIES

By default, the cross-reference entities load mappings load every entity from the following four dimension tables:

- Customers dimension: IA_CUSTOMERS
- Suppliers dimension: IA_SUPPLIERS
- Employees dimension: IA_EMPLOYEES
- Web Visitors dimension: IA_WEB_VISITORS

To limit which entities are loaded into the cross-reference table, edit the SQL override in the Source Qualifier of the appropriate PLP cross-reference entities load mapping.

To limit the records loaded into IA_XRF_ENTITIES

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the appropriate PLP cross-reference entities load mapping and edit the Source Qualifier for the type of record you want to limit on the cross-reference entities table:
   - **Customers.** Open the M_PLP_XRF_ENTITIES_CUSTOMERS_LOAD mapping, and edit the SQ_XRF_ENTITIES_CUSTOMERS_LOAD Source Qualifier.
   - S**uppliers.** Open the M_PLP_XRF_ENTITIES_SUPPLIERS_LOAD mapping, and edit the SQ_XRF_ENTITIES_SUPPLIERS_LOAD Source Qualifier.
   - **Employees.** Open the M_PLP_XRF_ENTITIES_EMPLOYEES_LOAD mapping, and edit the SQ_XRF_ENTITIES_EMPLOYEES_LOAD Source Qualifier.
   - **Web visitors.** Open the M_PLP_XRF_ENTITIES_WEB_VISITORS_LOAD mapping, and edit the SQ_XRF_ENTITIES_WEB_VISITORS_LOAD Source Qualifier.
3. Edit the SQL override in the Source Qualifier to select only the desired set of entities.
4. Validate and save your changes to the repository.

Specifying Correct Entity Cross-References in Flat File Uploads

It is recommended that you specify an entity type for the Reference Dimension ID when updating the Cross Reference Entities table (IA_XRF_ENTITIES) from a flat file. If you use random dimensions as the Reference Dimension ID, you risk making a cyclic reference in IA_XRF_ENTITIES, in which case the M_PLP_XRF_ENTITIES_CONSOLIDATION mapping creates a partial or incorrect cross-reference.

Figure 31 depicts a correctly structured flat file that matches the Base Dimension ID to the Reference Dimension ID, enabling a proper update to the IA_XRF_ENTITIES table. In the first row, the flat file shows that Supplier1 is the same entity as Customer1. The corresponding update to the IA_XRF_ENTITIES table updates the second row, changing Supplier1's ENTITY_KEY to be the same as Customer1. Supplier1 and Customer1 now share the same ENTITY_KEY—'Entity1'.

The next row in the flat file indicates that Employee1 is also the same entity as Customer1 (and, therefore, as Supplier1). The corresponding update to the IA_XRF_ENTITIES table updates the third row, changing Employee1's ENTITY_KEY value to 'Entity1', matching that of Customer1 and Supplier1.

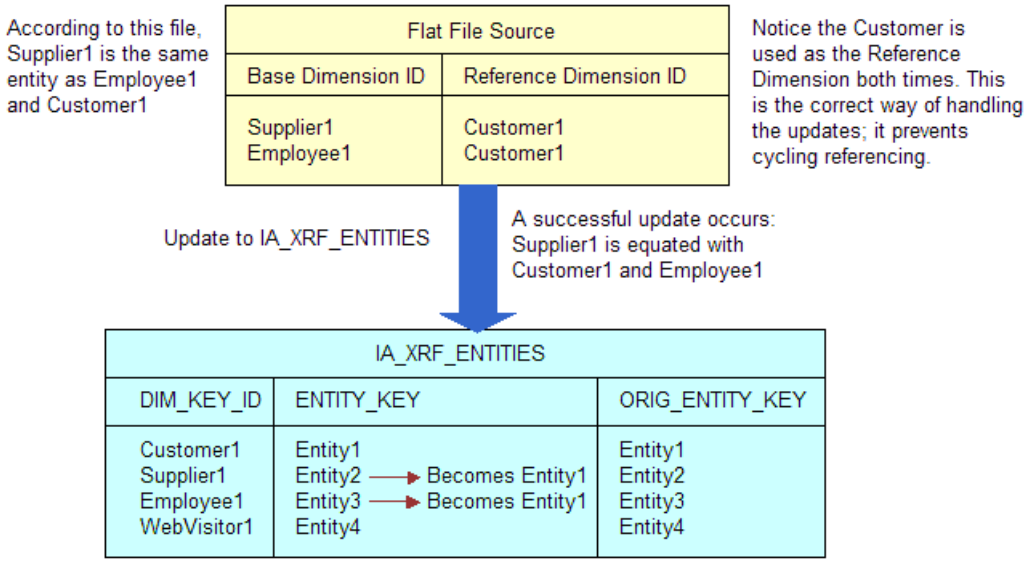


Figure 31. Successful Update of IA_XRF_ENTITIES Using the Same Entity Type as the Reference Dimension

Figure 32 depicts what happens if the entities used as Reference Dimension IDs in the flat file are cyclic from the definition. Supplier1, Employee1, and Customer1 are all still the same entity, as they were in the Figure 31. However, in the previous example, the Customer entity was consistently used as the Reference Dimension, whereas in Figure 32, the Supplier entity is used once and the Customer entity is also used only once.

The first row of the flat file links Employee1 to Supplier1. The corresponding update to the IA_XRF_ENTITIES table changes the ENTITY_KEY of Employee1 to 'Entity2', matching the ENTITY_KEY value of Supplier1. The second row of the flat file links Supplier1 to Customer1. The corresponding update to the IA_XRF_ENTITIES table changes the ENTITY_KEY column of Supplier1 to the value of 'Entity1'. The cross-reference, then, is only partial—it identifies Supplier1 and Customer1 as the same entity, but fails to indicate that Employee1 is yet another role played by that very same entity.

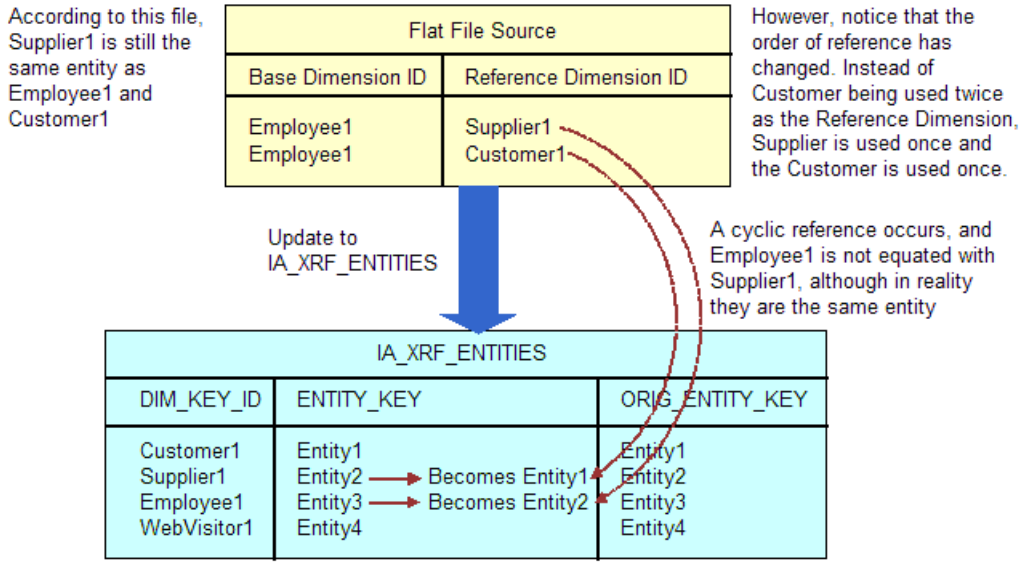


Figure 32. Unsuccessful Update of IA_XRF_ENTITIES Caused by Cyclic Reference

NOTE: After loading cross-references, make sure all relationships are accurately defined.

Configuration for Oracle 11i

Regardless of which module your are implementing, there is some general configuration information that is specific to Oracle 11i. In this section, you learn about each of these points.

Configuring the Group Currency Code

Oracle 11i provides document and local currencies, but not a group currency. Therefore, the Siebel Analytics Enterprise Data Warehouse has created a flat file called grp_curr.csv that contains a group currency code, which is loaded into the data warehouse during the initialization phase. It is initially set as 'USD' for United States dollars. However, if your group currency is different, you must edit the currency code in the flat file before you load it. For example, if your group currency is Belgian Francs, then you might set the currency code to 'BEF'.

To configure the group currency code for Oracle 11i

1 Open the GRP_CURR.CSV file. The file is located in the following directory:

```
<PowerMart Server Install Dir>\SRC FILES\
```

2 Edit the group currency code based on your enterprise's preference.

3 Run the M_I_STAGE_GROUP_CURRENCY mapplet to extract the new group currency code from the file, and load it into the TI_STAGE_GRP_CURR column in the staging table.

Incorporating Oracle Flex Fields into the Data Model

Oracle Applications have two kinds of flex columns:

- **Key flex fields.** Key flex fields are key fields that are required by Oracle, but their definitions can be modified by the user when configuring Oracle Applications.
- **Descriptive flex fields.** Descriptive flex fields are extension fields in the Oracle Applications database.

Key Flex Fields

The Siebel Analytics Enterprise Data Warehouse has prepackaged mappings to extract some of the key flex field data. However, if you configure any of these key flex fields other than the default configuration prepackaged by the Siebel Analytics Enterprise Data Warehouse, you must modify the mappings. The following are the key flex fields preconfigured for extraction:

- **GL Account, account segment only.** If you want to incorporate any other segment, you must modify the extract and load mappings.
- **Territory, segments 1, 2, and 3 only.** These key flex fields are used for Business Organizations Sales Geographical hierarchies only. If you want to incorporate any other segments, you need to modify the extract and load mappings.
- **Product Category, segments 1 and 2 only.** These key flex fields are used for classification only. If you want to incorporate any other segments, you need to modify the extract and load mappings.

If you wish to add other key flex field data to the data model, it is recommended that you use the extension columns available in the tables. For more information on using extension columns, see Overview for Storing, Extracting, and Loading Additional Data on page 323.

Descriptive Flex Fields

The Siebel Analytics Enterprise Data Warehouse prepackages different types of extension columns that you can use to hold any additional data. For more information on using extension columns, see Overview for Storing, Extracting, and Loading Additional Data on page 323.

Setting the Organization ID

The Organization ID for Oracle Applications defaults to 204 as the invoice information in Oracle Applications does not have an Organization ID for inventory organizations. This column must be configured with the correct Organization ID or the ADI is not able to resolve dimension keys, such as the Product key and Sales Product Key, for the applicable fact load.

For example, by default the Sales Order Lines table should supply the load mapping with the Product key. However, if it does not, the ADI performs a lookup to retrieve the Product key from IA_PRODUCTS. The lookup uses the Organization ID to help resolve the Product key. The ADI uses the SOURCE_ID, CREATED_ON_DT, and the PRODUCT_ID, where the PRODUCT_ID is defined as the INVENTORY_ITEM_ID concatenated with the ORGANIZATION_ID. If the wrong Organization ID is provided, then the Product ID is not defined correctly, which results in a failed lookup for the Product key, and the Product key is not loaded into the applicable fact table during the load.

If your business has multiple Organization IDs for inventory organizations, you can use the most commonly used Organization ID, or the master Organization ID as defined in your Oracle Applications instance.

To reset the Organization ID

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Edit all applicable Source Adapter mapplets.

   The following is a list of applicable mapplets for Oracle 11i:

   - MPLT_SAI_SALES_IVCLNS
   - MPLT_SAI_GL_REVENUE
   - MPLT_SAI_GL_REVENUE_UPDATE
3. Double-click the Expression transformation to open the Edit Transformation box.
4. In the Ports tab, edit the expression for the VAR_ORGANIZATION_ID.

   By default, it is set to '204'. Edit it by setting it to the master organization ID.
5. Validate and save your changes to the repository.

Mapping Source Customer Hierarchies to the Customers Dimension Table

Customer hierarchies are typically custom-defined in Oracle Applications. If you want to include this data in the data warehouse, you must perform two customization processes:

- Import the hierarchies into the TI_CUSTOMERS staging table for Oracle 11i.
- Redefine the category lookup so that the new category data is loaded into the Siebel Analytics Enterprise Data Warehouse.

The two processes are illustrated in Figure 33.

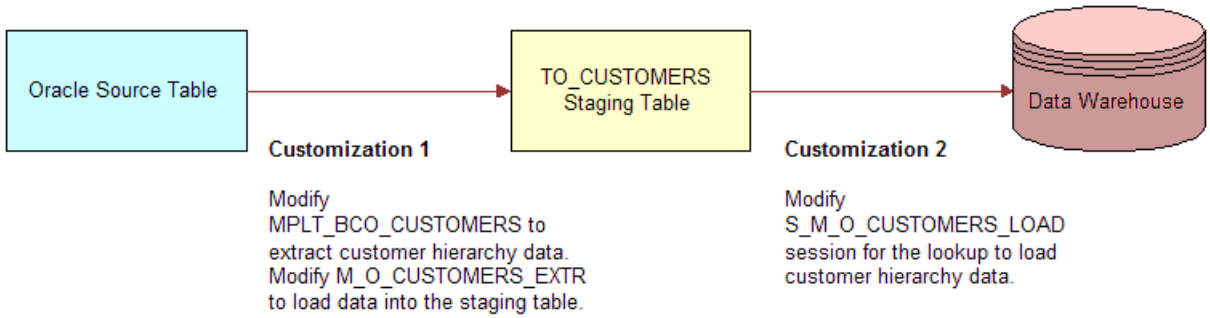


Figure 33. Oracle Applications: Customization Processes for Custom Customer Hierarchies Load

To load the source-defined customer hierarchies into the TO_CUSTOMERS staging table for Oracle 11i, you must first edit the MPLT_BCI_CUSTOMERS Business Component mapplet to extract the hierarchy in addition to the customer information. After your Business Component is set up to extract the customer hierarchies, you must verify that the M_I_CUSTOMERS_EXTRACT extract mapping outputs this data to the data warehouse.

To map Oracle-defined customer hierarchies to the Customers dimension table

1. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder.
2. Open the MPLT_BCI_CUSTOMERS Business Component mapplet for Oracle 11i.

   Modify the mapplet to extract the customer hierarchy columns.

   If the hierarchies and customers are maintained in the same Oracle Applications source table, then load these columns into the SQL Source qualifier and map them to the Business Component output.

   However, if the hierarchies and customers are stored in two different tables, then the Business Component must be modified to include both source tables so that it can include both sets of information.
3. Modify the M_I_CUSTOMERS_EXTRACT extract mapping, to map the source customer hierarchy columns to the extension hierarchy columns in the TI_CUSTOMERS staging table.

   If the source table has hierarchy codes, but no descriptions associated with these codes, map the Oracle Applications codes to both the hierarchy name columns and the hierarchy code columns. Hierarchy name columns are named as CUST_HIER*X*_NAME, where 'X' denotes the level of the customer hierarchy.

   If the source table has code values, but the corresponding descriptions are in a different source table, you must build new codes mappings that load the data into the IA_CODES table.
4. Save your changes to the repository.

   NOTE: After you complete the previous process, you must modify a hierarchy lookup in the customer dimension so that the system extracts the new categories.

To configure the category lookup

1. In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i folder.

   Open the S_M_I_CUSTOMERS_LOAD session for Oracle 11i, to open the Edit Tasks box.
2. In the Transformations tab, modify the lookup in the MPLT_ADI_CUSTOMERS.LKP_CUST_HIER*X* field in the IA_CODES table by adding the new category in the SQL statement.

   Select the arrow to edit the WHERE clause.

   You can use the following statement as a sample of how to structure your SQL statement:

   ```
   WHERE IA_CODES.CATEGORY = 'GENERIC'AND IA_CODES.LANGUAGE = 'E'
   ```

   For example, if you have mapped something to the CUST_HIER1_CODE then the SQL to the IA_CODES table should have the new category code in place of 'GENERIC'.
3. Validate and save your changes to the repository.

Configuring Product Categories

As initially configured, the Siebel Analytics Enterprise Data Warehouse extracts product categories where the CATEGORY_SET_ID is '2' or '27'. However, it is likely that the categories you extract from the source system are different from these prepackaged categories. Therefore, you must reconfigure your product categories by making two customizations:

- Identify and extract only the categories that you want to report against.
- Properly position the data so that it loads into the Siebel Analytics Enterprise Data Warehouse.

The two processes are illustrated in Figure 34.

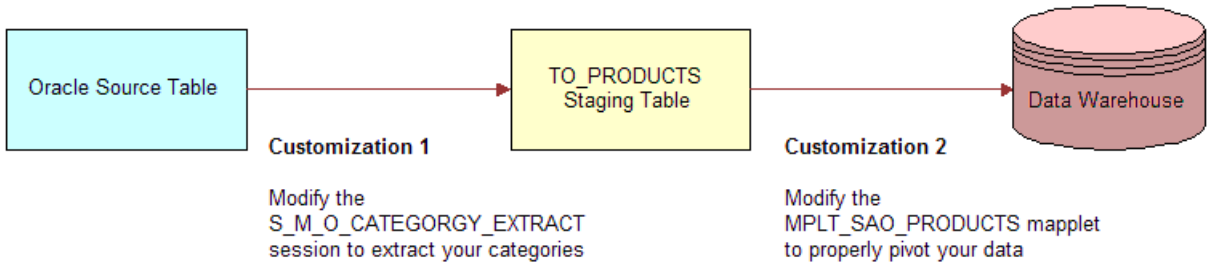


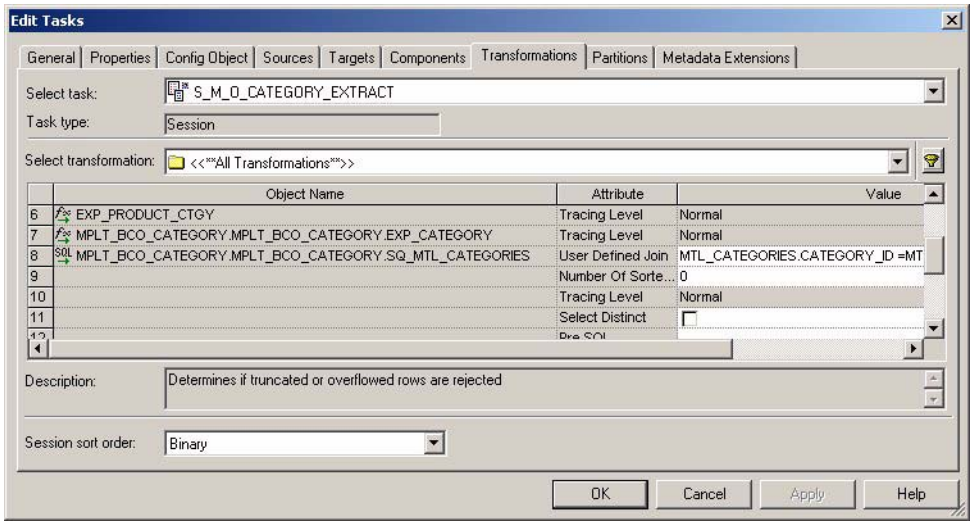
Figure 34. Oracle Applications: Customization Processes for Product Categories

To configure product category extract from Oracle Applications

1. Identify the categories that need to be mapped to the extension hierarchy columns.

   These categories are extracted from the source and placed in the hierarchy column specified.
2. In PowerCenter Workflow Manager, open the Configuration for Oracle Applications v11i.
3. Open the S_M_I_CATEGORY_EXTRACT session to open the Edit Tasks box.

4 In the Transformations tab, select the attribute value field for SQL Query to display an arrow, as shown in the following figure.



5 Select the arrow to edit the WHERE clause.

The following statement is an example of how to structure the WHERE clause:

```
WHERE...MTL_CATEGORY_SETS.CATEGORY_SET_ID IN (3,42,90)
```

In this example, the WHERE clause extracts categories where the Category Set IDs are 3, 42, and 90.

6 Click OK, and then click OK to close the Edit Tasks box.

After you have loaded your selected categories into the staging table, you must define which categories should link to particular hierarchy columns in an IA table.

In the M_I_PRODUCTS_LOAD for Oracle 11i, join the selected category data with the low-level product information. Not all segments of each category may be populated. Table 59 is an example of combined data in the Staging Area.

Table 59. Combined Data in the Staging Area

PRODUCT_NAME	CATEGORY_SET_ID	SEGMENT1	SEGMENT2	SEGMENT3
Tommy Atkins Mango	27	Mangos	Tropical Fruits	
Tommy Atkins Mango	4	Produce		

An aggregator is prepackaged to combine data and load it into the Hierarchy Extension columns to load categories into a single row as opposed to having multiple rows of categories. Using the example in Table 59, the AGG_INVENTORY_ITEMS aggregator contained in the MPLT_SAO_PRODUCTS mapplet would distribute the data so that it would appear as shown in Table 60.

Table 60. Pivoted Data in Hierarchy Extension Columns

PRODUCT_NAME	EXT_PROD_ HIER1_CODE	EXT_PROD_ HIER2_CODE	EXT_PROD_ HIER3_CODE
Tommy Atkins Mango	Mangos	Tropical Fruits	Produce

Your categories and segments may differ from the default configuration, in which case you must modify the AGG_INVENTORY_ITEMS aggregator by reconfiguring the hierarchy code definitions to populate the product hierarchy properly.

TIP: After redefining your product hierarchies, test your logic by tracking the data for only a few products.

To reconfigure the product hierarchy loads

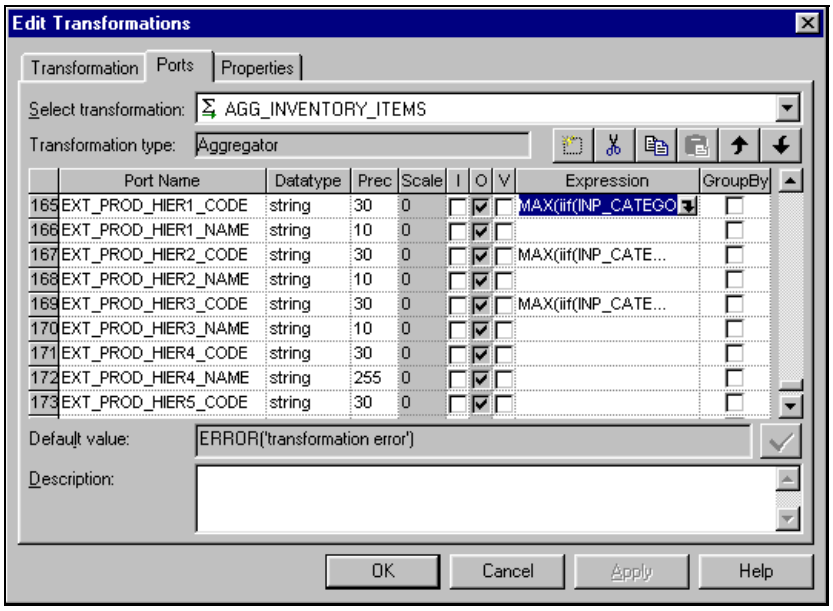
1. Identify the categories to map to the hierarchy extension columns.

   NOTE: There are six hierarchy extension columns in the Siebel Analytics Enterprise Data Warehouse that are available for configuration.

2. In PowerCenter Designer, open the Configuration for Oracle Applications v11i folder, and expand the Mapplets folder.
3. Open the MPLT_SAI_PRODUCTS mapplet for Oracle 11i.
4. Double-click the AGG_INVENTORY_ITEMS aggregator to open the Edit Transformation box.
5. In the Ports tab, scroll down to find the hierarchy code port.

   Hierarchy levels are named with the following convention EXT_PROD_HIER*X*_CODE, where 'X' denotes the level within the hierarchy. For example, if you want to edit the first level of your hierarchy, you must edit the definition for EXT_PROD_HIER1_CODE port.

6 In the Expression column, select the expression for the hierarchy code to display an arrow, as shown in the following figure.



7 Select the arrow to edit the expression for the hierarchy code.

8 Modify the expression that defines the hierarchy level.

For example, if you want to populate Hierarchy 1 with Segment 3 column data, where the CATEGORY_SET_ID = 4, then add the following statement to the expression:

```
FIRST(inp_segment3, inp_category_set_id = 4)
```

9 Validate and save your changes to the repository.

Modifying the Source ID

By default, the Source ID is set to OAP11i in the mappings corresponding to the sessions in Table 61. However, you can modify this definition in the INFA_PARAMETERS.PAR file. The file contains OAP11i as the value for the $$SOURCE_ID field for each of the sessions contained in Table 61.

Table 61. Session with OAP11i as the Default Source ID

Session
S_M_I_BUSN_ORGS_HR_EXTRACT
S_M_I_CODES_FND_COMMON_LOOKUPS
S_M_I_EMPLOYEES_EXTRACT
S_M_I_EMPLOYMENT_EXTRACT

Table 61. Session with OAP11i as the Default Source ID

Session
S_M_I_EMP_EVENTS_END_ABSENCE_EXTRACT
S_M_I_EMP_EVENTS_END_ASSIGN_EXTRACT
S_M_I_EMP_EVENTS_END_SERVICE_EXTRACT
S_M_I_EMP_EVENTS_SALARY_EXTRACT
S_M_I_EMP_EVENTS_START_ABSENCE_EXTRACT
S_M_I_EMP_EVENTS_START_ASSIGN_EXTRACT
S_M_I_EMP_EVENTS_START_SERVICE_EXTRACT
S_M_I_EMP_HISTORY_A1_DERIVE
S_M_I_EMP_HISTORY_A2_DERIVE
S_M_I_EMP_SNAPSHOT_1_EXTRACT_P1
S_M_I_EMP_SNAPSHOT_2_EXTRACT_P2
S_M_I_EMP_SNAPSHOT_3_EXTRACT_P3
S_M_I_EMP_SNAPSHOT_4_EXTRACT_P4
S_M_I_EVENT_TYPES_EXTRACT1
S_M_I_EVENT_TYPES_EXTRACT2
S_M_I_EVENT_TYPES_EXTRACT3
S_M_I_JOBS_EXTRACT
S_M_I_PAYROLL_A1_LOAD
S_M_I_PAYROLL_A2_LOAD
S_M_I_PAYROLL_A3_LOAD
S_M_I_PAYROLL_EXTRACT
S_M_I_PAY_GRADES_EXTRACT
S_M_I_PAY_TYPES_EXTRACT
S_M_I_POSITIONS_EXTRACT
S_M_I_BUSN_ORGS_HR_EXTRACT

After you have modified the file, the next time you run the session, a new Source ID is loaded.

To redefine the Source ID

1 Open the file_parameters_ora11i.csv file.

2 Replace the default Source ID with the new Source ID.

   Perform this action for all applicable sessions.

3 Save and close the file.

Modifying the Last Extract Date

By default, the $$LAST_EXTRACT_DATE is set to 01/01/1970 for Oracle 11i. However, if you need to modify this date, you can do so by modifying the INFA_PARAMETERS.PAR file. After you have modified the file, the next time you run the session, the new Last Extract date is populated.

To redefine the Last Extract date

1 Open the file_parameters_ora11i.csv file.

2 Replace the default Last Extract date with the new Last Extract date.

   Perform this action for all applicable sessions.

3 Save and close the file.

Configuration for Post-Load Processing

This section provides configuration procedures for the Post-Load Processes (PLP) that span multiple modules. For PLP configuration points that are specific to a module, see that module's configuration chapter.

Changing the Time Period for Key Figure Data

Most Siebel Analytics Enterprise Data Warehouse key figure tables store data at the calendar month level by default. To change the time period for any of these tables, you can use the method described in this section.

You may want to store key figure data at an interval different than the default, such as the calendar quarter, calendar week, and so on. To build a key figure table for any time interval other than the default, you must perform the high-level steps described in this section.

1 Make the change by overwriting the SQL in the Source Qualifier of the PLP extract mapping for that table.

   The PLP extract mappings extract data from existing tables in the Siebel Analytics Enterprise Data Warehouse (usually IA tables). Most key figure tables have multiple extract mappings, because they extract data from multiple Siebel Analytics Enterprise Data Warehouse tables. The Source Qualifier of each PLP extract mapping contains the logic for setting up the time period. To change the time period for a key figure table, edit every PLP extract mapping used to build that table.

2 If applicable, change the definition of Previous Period in the Previous Periods derive mapping for the key figure table in question.

   Not all key figure tables have a Previous Period derive mapping.

   Previous period is defined as the period prior to the earliest period showing a transactional change in the extract time window. The previous period is used for cumulative calculations in key figure tables. For instance, the Customer key figure table extracts fact table transactional data for customers and keeps track of not only each transaction, but of the cumulative values for all transactions. If a September extract (extracting transactions that occurred between the beginning and ending dates of September) also shows a changed record for June and two for August, cumulative values must be recalculated from June until the current extract (September).

   If the changed June record is the earliest in the extract, June is considered the minimum period, or the earliest period showing a change in this extract. The Previous Period is one prior to that—May. The Siebel Analytics Enterprise Data Warehouse logic uses the Previous Period's cumulative value as a basis for making new calculations to reflect all subsequent changed periods. So, in this instance, the Siebel Analytics Enterprise Data Warehouse uses May's cumulative value and recalculates June, then July, then August, and finally September's cumulative values.

   If you change the Customer key figure table to be quarterly rather than monthly, you must change the Previous Period to be Previous Quarter rather than Previous Month. This change is made in the Previous Periods derive mapping for the key figure table in question.

3 Schedule every session task or workflow that corresponds to the mapping you altered to run at the new frequency.

   For instance, if you have changed a monthly key figure table to be weekly, you must schedule the session tasks or workflows that create that key figure table to run weekly as well. The session task that creates the base table should be run more frequently than the session task that creates the key figure table; for example, run the session task to create the base table on a daily basis, and the session task to create the key figure table on a weekly basis.

Objects to Edit When Changing Key Figure Time Periods

The following is a list of key figure tables for which you may wish to change time periods. The mappings and Source Qualifiers that require editing are listed for each table:

- **IA_CUST_KF.** To change the time period for the Customer key figure table, modify the components shown in Table 62.

Table 62. Editing IA_CUST_KF Mappings and Source Qualifiers for Key Figure Time Period Changes

Mapping	Source Qualifier
Extract Mappings	
M_PLP_CUST_KF_CUST_EVENTS_EXTRACT	SQ_CUST_KF_CUST_EVENTS_EXTRACT
M_PLP_CUST_KF_CMGN_OFFERS_EXTRACT	SQ_CUST_KF_CMGN_OFFERS_EXTRACT
M_PLP_CUST_KF_PSU_XACTS_EXTRACT	SQ_CUST_KF_PSU_XACTS_EXTRACT
M_PLP_CUST_KF_CUSTOMERS_EXTRACT	SQ_CUST_KF_CUSTOMERS_EXTRACT
PLP Previous Period Derive Mapping	
M_PLP_CUST_KF_PREV_PERIODS_DERIVE	SQ_CUST_KF_PREV_PERIODS_DERIVE
Flat File Previous Period Derive Mapping	
M_F_CUST_KF_PREV_PERIODS_DERIVE	SQ_F_CUST_KF_PREV_PERIODS_DERIVE

- **IA_PRODUCT_KF.** To change the time period for the Product key figure table, modify the components shown in Table 63.

Table 63. Editing IA_PRODUCT_KF Mappings and Source Qualifiers for Key Figure Time Period Changes

Mapping	Source Qualifier
Extract Mappings	
M_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT	SQ_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT
M_PLP_PRODUCT_KF_PSS_XACTS_EXTRACT	SQ_PLP_PRODUCT_KF_PSS_XACTS_EXTRACT
M_PLP_PRODUCT_KF_ PRODUCTS_EXTRACT	SQ_PLP_PRODUCT_KF_PRODUCTS_EXTRACT
PLP Previous Period Derive Mapping	
M_PLP_PRODUCT_KF_PREV_PERIODS_ DERIVE	SQ_PLP_PRODUCT_KF_PREV_PERIODS_ DERIVE
Flat File Previous Period Derive Mapping	
M_F_PRODUCT_KF_PREV_PERIODS_DERIVE	SQ_F_PRODUCT_KF_PREV_PERIODS_DERIVE

Editing Key Figure Mappings to Change the Time Period

To change the time period from the default period, edit each PLP extract mapping for the key figure table you want to change. Within each mapping, change the time period in the Source Qualifier.

To change the time period for a key figure table's extract mapping

1. In PowerCenter Designer, open the `Configuration for Post Load Processing` folder.
2. For the key figure table in question, open each PLP extract mapping that requires editing.

   The previous section lists all extract mappings that need to be edited for each key figure table.
3. Open the `M_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT` mapping and follow the remaining steps. Repeat Step 4 and Step 5 with the `M_PLP_PRODUCT_KF_PSS_XACTS_EXTRACT` mapping.
4. In the PLP extract mapping, edit the SQL in the Source Qualifier to change the period references from the default to the period of your choice.

   The available options are `CAL_WEEK`, `CAL_MONTH`, `CAL_QTR`, `CAL_YEAR`, `FSC_WEEK`, `FSC_MONTH`, `FSC_QTR`, and `FSC_YEAR`.

   The SQL statement you use for week, month, quarter, and year is structurally the same. The only difference is the period name, such as `CAL_MONTH` or `CAL_QTR`.

   For example, to change the time period for `IA_PRODUCT_KF` from the default monthly to quarterly, consult Table 63 to find out which mappings and Source Qualifiers you need to modify. One of the mappings listed in this table is the `M_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT` mapping. When you edit the `M_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT` mapping, edit the `SQ_PLP_PRODUCT_KF_PSU_XACTS_EXTRACT` Source Qualifier and change all `CAL_MONTH` references to `CAL_QTR`.

   For instance, the default SQL for the Source Qualifier transformation is as follows:

   ```
   SELECT DISTINCT NU_PSS_XACTS.PRODUCT_ID, NU_PSS_XACTS.SOURCE_ID,
   IA_DATES.CAL_MONTH_START_DT, IA_DATES.CAL_MONTH_END_DT FROM NU_PSS_XACTS,
   IA_DATES WHERE NU_PSS_XACTS.XACT_DK=IA_DATES.DATE_KEY AND NOT EXISTS (SELECT '1'
   FROM PL_PRODUCT_KF WHERE PL_PRODUCT_KF.PRODUCT_ID=NU_PSS_XACTS.PRODUCT_ID AND
   PL_PRODUCT_KF.SOURCE_ID = NU_PSS_XACTS.SOURCE_ID
   ANDPL_PRODUCT_KF.PERIOD_START_DT=IA_DATES.CAL_MONTH_START_DT
   ANDPL_PRODUCT_KF.PERIOD_END_DT=IA_DATES.CAL_MONTH_END_DT)
   ```

   However, if you changed the time period from month to quarter, then you would modify the SQL as follows:

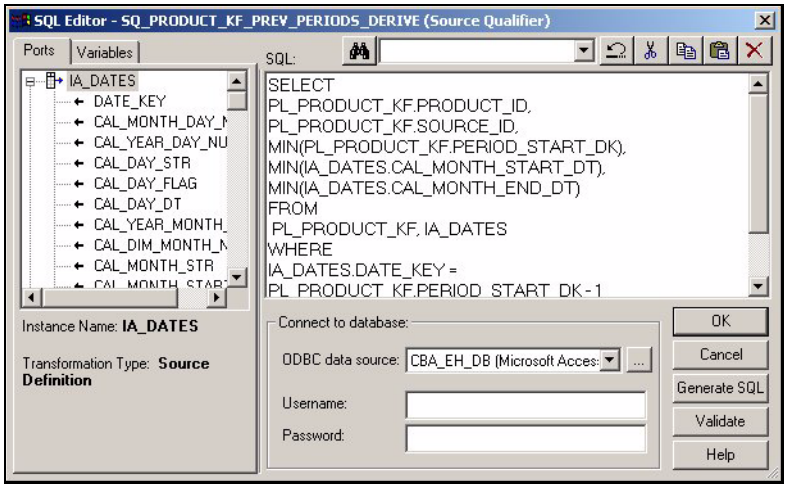
   ```
   SELECT DISTINCT NU_PSS_XACTS.PRODUCT_ID, NU_PSS_XACTS.SOURCE_ID,
   IA_DATES.CAL_QTR_START_DT, IA_DATES.CAL_QTR_END_DT FROM NU_PSS_XACTS, IA_DATES
   WHERE NU_PSS_XACTS.XACT_DK=IA_DATES.DATE_KEY AND NOT EXISTS (SELECT '1' FROM
   PL_PRODUCT_KF WHERE PL_PRODUCT_KF.PRODUCT_ID=NU_PSS_XACTS.PRODUCT_ID AND
   PL_PRODUCT_KF.SOURCE_ID = NU_PSS_XACTS.SOURCE_ID
   ANDPL_PRODUCT_KF.PERIOD_START_DT=IA_DATES.CAL_QTR_START_DT
   ANDPL_PRODUCT_KF.PERIOD_END_DT=IA_DATES.CAL_QTR_END_DT)
   ```
5. Validate and save your changes.

To change the time period for a key figure table's Previous Periods mapping

1. If applicable, for the key figure table in question, open the Previous Periods derive mapping.

2 Edit the Source Qualifier's SQL statement.

The following figure illustrates the SQL for the M_PLP_PRODUCT_KF_PREV_PERIODS_DERIVE mapping's Source Qualifier.



For example, to change the time period for IA_PRODUCT_KF from month to quarter, consult Table 63 to find out the applicable Previous Periods mapping to edit. For the M_F_PRODUCT_KF_PREV_PERIODS_DERIVE mapping, edit the SQ_F_PRODUCT_KF_PREV_PERIODS_DERIVE Source Qualifier and change CAL_MONTH references to CAL_QTR.

For instance, the default SQL for the Source Qualifier transformation is as follows:

```
SELECT PL_PRODUCT_KF.PRODUCT_ID, PL_PRODUCT_KF.SOURCE_ID,
MIN(PL_PRODUCT_KF.PERIOD_START_DK), MIN(IA_DATES.CAL_MONTH_START_DT),
MIN(IA_DATES.CAL_MONTH_END_DT) FROM PL_PRODUCT_KF, IA_DATES WHERE
IA_DATES.DATE_KEY = PL_PRODUCT_KF.PERIOD_START_DK - 1 GROUP BY
PL_PRODUCT_KF.PRODUCT_ID, PL_PRODUCT_KF.SOURCE_ID
```

However, if you changed the time period from month to quarter, then you would modify the SQL as follows:

```
SELECT PL_PRODUCT_KF.PRODUCT_ID, PL_PRODUCT_KF.SOURCE_ID,
MIN(PL_PRODUCT_KF.PERIOD_START_DK), MIN(IA_DATES.CAL_QTR_START_DT),
MIN(IA_DATES.CAL_QTR_END_DT) FROM PL_PRODUCT_KF, IA_DATES WHERE
IA_DATES.DATE_KEY = PL_PRODUCT_KF.PERIOD_START_DK - 1 GROUP BY
PL_PRODUCT_KF.PRODUCT_ID, PL_PRODUCT_KF.SOURCE_ID
```

3 Validate and save your changes.

To change the time interval for loading the session

1 In PowerCenter Workflow Manager, open the appropriate folder.

- For regular PLP mappings, open the Configuration for Post Load Processing folder.

- For loading key figures directly from flat files, open the Configuration for Universal Source folder.

NOTE: This step does not apply for all key figure tables, because not all key figure tables have a Previous Period derive mapping. To find out whether or not the key figure table you are working with has such a mapping, see the discussion on objects to edit when changing key figure time periods in Configuration for Post-Load Processing on page 288. That section also lists which objects you need to edit when changing the time period for a particular key figure table.

2 At the folder level, create a new workflow following the naming conventions as specified in the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

For example, the current mappings for IA_PRODUCT_KF (if you are using regular PLP mappings to build that table) are under the B_PLP_CRA_MONTHLY workflow. This workflow contains all the PLP sessions for the Marketing, Sales, and Service modules that run monthly.

To change the time period for IA_PRODUCT_KF from month to quarter, move the B_PLP_PRODUCT_KF workflow out and under a new workflow at the folder level, called B_PLP_CRA_QUARTERLY. Then, reschedule B_PLP_PRODUCT_KF to run on a quarterly cycle. If you change any other key figure tables to run quarterly, you can move the workflows that build them into the same location.

3 Reschedule the session task's run time.

Excluding Inactive Customers from the Customer and Product Key Figure Tables

The Customer (IA_CUST_KF) and Product (IA_PRODUCT_KF) key figure tables extract data from the Customers and Products respectively, for every customer, regardless of whether there are transactions reported for that customer during the period in question. You may want to include data for active customers or products only. For instance, you may want to base a customer's inclusion in the key figure tables on the Delete or Current Flag in the IA_CUSTOMERS dimension table. To create such a criterion, add a filter in the SQL of the PLP extract mapping.

To filter out inactive customers from IA_CUST_KF and IA_PRODUCT_KF

1 In PowerCenter Designer, open the Configuration for Post Load Processing folder.

2 Open the following extract mapping, depending on which key figure table you want to filter:

- If you want to only include active customers, then you must modify the data loaded into the Customer key figure table, IA_CUST_KF. To do so, open the M_PLP_CUST_KF_CUSTOMERS_EXTRACT mapping, which extracts from the IA_CUSTOMERS dimension table; it is on this table that you can add your filter.
- If you want to only include active products, then you must modify the data loaded into the Product key figure table, IA_PRODUCT_KF. To do so, open the M_PLP_PRODUCT_KF_PRODUCTS_EXTRACT mapping, which extracts from the IA_PRODUCTS dimension table; it is on this table that you can add your filter.

3 Edit the SQL in the Source Qualifier, entering an additional WHERE clause in the SELECT statement to limit the extract to only active customers, products, or channel points.

For example, if you wanted to include only active customers, you might modify the SQL in the Source Qualifier by changing it as follows:

```
SELECT DISTINCT IA_CUSTOMERS.KEY_ID, IA_CUSTOMERS.SOURCE_ID,
IA_DATES.CAL_MONTH_START_DT, IA_DATES.CAL_MONTH_END_DT

FROM

 IA_CUSTOMERS, IA_DATES

WHERE

 IA_DATES.CAL_MONTH_FLAG = 'Previous' AND

IA_DATES.CAL_MONTH_DAY_NUM = 1 AND

IA_CUSTOMERS.CUSTOMER_KEY > 0

IA_CUSTOMERS.CURRENT_FLAG = 'Y'

AND

NOT EXISTS

(SELECT '1' FROM PL_CUST_KF WHERE

PL_CUST_KF.CUSTOMER_ID = IA_CUSTOMERS.KEY_ID AND

PL_CUST_KF.SOURCE_ID = IA_CUSTOMERS.SOURCE_ID AND

PL_CUST_KF.PERIOD_START_DT = IA_DATES.CAL_MONTH_START_DT AND
PL_CUST_KF.PERIOD_END_DT = IA_DATES.CAL_MONTH_END_DT)
```

4 Validate and save your changes.

Changing Index Band Definitions

The Siebel Analytics Enterprise Data Warehouse include some metrics that allow you to place customers in a number of index bands. For instance, you can rate customers on the probability that you can cross-sell to them by placing them into Low, Medium, or High index bands. In this instance, the index indicates whether you have a low, medium, or high chance of selling products from other product lines to customers who have purchased from you.

The logic that determines which index band a customer is placed into is prepackaged in either a PLP mapping or a flat file mapping. (Flat file sources and warehouse tables both can provide information to key figure tables.) This section describes the prepackaged criteria used to determine index band definitions, as well as instructions for changing the definitions to suit your business needs.

NOTE: Index band definitions are hard coded into the ADI mapplets. However, configuration changes should never occur within the ADI. You can refer to the default logic within the ADI, but do not change the default index band definitions. Instead, you can build an Expression transformation that contains the changed logic and place this object directly before the ADI mapplet in the final load mapping. By having the new Expression transformation pass a value to the ADI, the ADI's logic for creating the index is disabled. As a result, the index's value created by the new Expression transformation is passed through the ADI unchanged. This value is then loaded into the key figure table.

Cross-Sell Probability Index in the Customer Key Figure Table

The current definition to determine whether a customer is rated as `low`, `medium`, or `high'` on cross-sell probability is as follows:

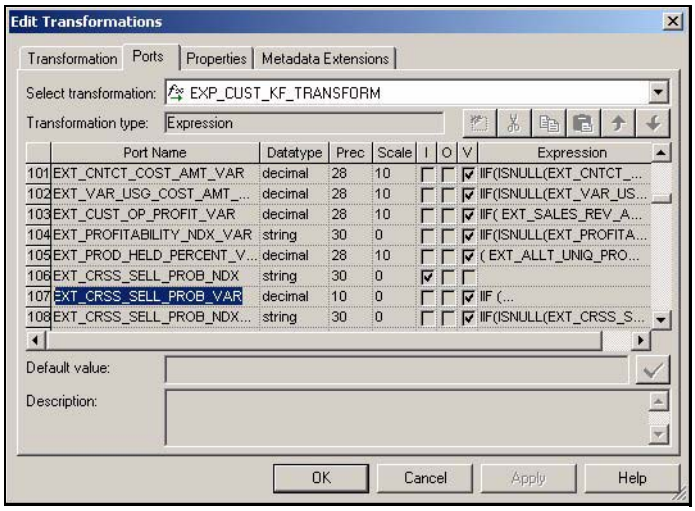
```
If EXT_CRSS_SELL_PROB_VAR > 0.75,
'High'


If EXT_CRSS_SELL_PROB_VAR > 0.5
And EXT_CRSS_SELL_PROB_VAR <= 0.75,
'Medium'


If EXT_CRSS_SELL_PROB_VAR > 0.25
And EXT_CRSS_SELL_PROB_VAR <= 0.5,
'Moderate'


If EXT_CRSS_SELL_PROB_VAR > 0
And EXT_CRSS_SELL_PROB_VAR <= 0.25,
'Low'
```

To change the Cross-Sell Probability Index definition

1 In PowerCenter Designer, open the Siebel Applications folder.

2 In Mapplet Designer, open the MPLT_ADI_CUST_KF mapplet to view the default logic.

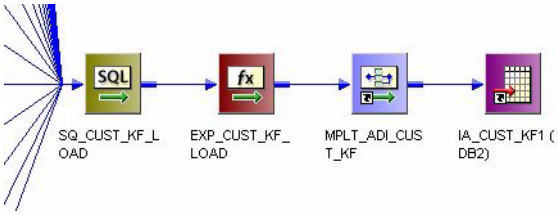
You can find it in the EXT_CRSS_SELL_PROB_NDX_VAR port of the EXP_CUST_KF_TRANSFORM Expression transformation, as shown in the following figure. You can copy this logic into the new Expression and you create and modify it there, as shown in the following figure.



3 Open the final load mapping of the key figure table.

- For regular PLP mappings, in the Configuration for Post Load Processing folder, open the M_PLP_CUST_KF_LOAD mapping.
- If you build your key figures using a flat file upload, in the Configuration for Generic Source folder, open the M_F_CUST_KF_LOAD mapping.

The M_PLP_CUST_KF_LOAD mapping is shown in the following figure.



4 Open the existing Expression transformation located after the Source Qualifier transformation in the final load mapping.

5 In the Expression transformation, enter your modified index band definition logic.

This new index calculation should feed into the ADI. Because it is a precalculated value fed to the ADI, it overrides the ADI's predefined logic.

6 Validate and save your changes.

Life Stage Index Definition in the Customer Key Figure Table

The customer Life Stage Index consists of the following values—Minor, Young Adult, Young Professional, New Nest, Full Nest, Empty Nest, Silver Age, Golden Age, and Other. The default definition to determine the life stage into which a customer falls is as follows:

```
If EXT_CUSTOMER_AGE_VAR <= 18,

'Minor',


If EXT_CUSTOMER_AGE_VAR >= 19

And EXT_CUSTOMER_AGE_VAR <= 24

And EXT_INDV_MARITAL_STATE = 'Single'

And EXT_INDV_EMP_STATUS = 'Working'

And EXT_NO_OF_CHILDREN = 0,

'Young Adult',

If EXT_CUSTOMER_AGE_VAR >= 25

And EXT_CUSTOMER_AGE_VAR <= 35

And EXT_INDV_MARITAL_STATE = 'Married'

Or EXT_INDV_MARITAL_STATE = 'Separated'

And EXT_INDV_EMP_STATUS = 'Working'

And EXT_NO_OF_CHILDREN = 0,

'Young Professional',


If EXT_CUSTOMER_AGE_VAR >= 36

And EXT_CUSTOMER_AGE_VAR <= 45

And EXT_INDV_MARITAL_STATE = 'Married'

Or EXT_INDV_MARITAL_STATE = 'Separated'

And EXT_INDV_EMP_STATUS = 'Working'

And EXT_NO_OF_CHILDREN > 0,

'New Nest',


If EXT_CUSTOMER_AGE_VAR >= 46

And EXT_CUSTOMER_AGE_VAR <= 55)
```

```
And EXT_INDV_MARITAL_STATE = 'Married'
Or EXT_INDV_MARITAL_STATE = 'Separated'
And EXT_INDV_EMP_STATUS = 'Working'
And EXT_NO_OF_CHILDREN > 0,
'Full Nest',

If EXT_CUSTOMER_AGE_VAR >= 56
And EXT_CUSTOMER_AGE_VAR <= 65)
And EXT_INDV_MARITAL_STATE = 'Married'
Or EXT_INDV_MARITAL_STATE = 'Separated'
And EXT_INDV_EMP_STATUS = 'Working'
And EXT_NO_OF_CHILDREN = 0,
'Empty Nest',

If EXT_CUSTOMER_AGE_VAR >= 66
And EXT_CUSTOMER_AGE_VAR <= 75
And EXT_INDV_MARITAL_STATE = 'Married'
Or EXT_INDV_MARITAL_STATE = 'Separated'
And EXT_INDV_EMP_STATUS = 'Retired'
And EXT_NO_OF_CHILDREN = 0,
'Silver Age'

If EXT_CUSTOMER_AGE_VAR >= 76
And EXT_INDV_MARITAL_STATE = 'Separated'
Or EXT_INDV_MARITAL_STATE = 'Widow'
Or EXT_INDV_MARITAL_STATE = 'Widower'
And EXT_INDV_EMP_STATUS = 'Retired'
And EXT_NO_OF_CHILDREN = 0,
'Golden Age'
```

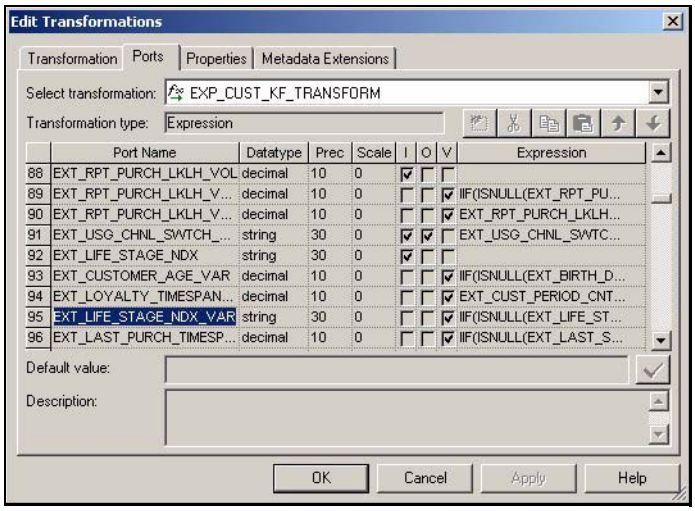
```
Else 'Other'
```

To change the customer Life Stage Index definition

1 In PowerCenter Designer, open the Siebel Applications folder.

2 In Mapplet Designer, open the MPLT_ADI_CUST_KF mapplet to view the default logic.

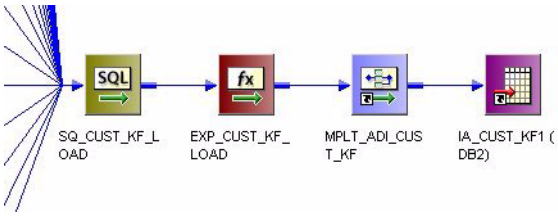
You can find it in the EXT_LIFE_STAGE_NDX_VAR port of the EXP_CUST_KF_TRANSFORM Expression transformation, as shown in the following figure. You can copy this logic into the new Expression and you create and modify it there.



3 Open the final load mapping of the key figure table.

- For regular PLP mappings, in the Configuration for Post Load Processing folder, open the M_PLP_CUST_KF_LOAD mapping.
- If you build your key figures using a flat file upload, in the Configuration for Generic Source folder, open the M_F_CUST_KF_LOAD mapping.

The M_PLP_CUST_KF_LOAD mapping is shown in the following figure.



4 Open the existing Expression transformation located after the Source Qualifier transformation in the final load mapping.

5 In the Expression transformation, enter your modified index band definition logic.

This new index calculation should feed into the ADI. Because it is a precalculated value fed to the ADI, it overrides the ADI's predefined logic.

6 Validate and save your changes.

Customer Loyalty Index Definition in the Customer Key Figure Table

The customer Loyalty Index can have one of the following values, in order of least to most loyal—Customer, Supporter, Sponsor, Patron, and Advocate. The methodology for determining in which category the customer falls is determined by the EXT_CUST_PERIOD_CNT_VAR port. This port provides a count of the number of key figure periods for which the person was a customer of the organization. The default categorization of the customer's loyalty is as follows:

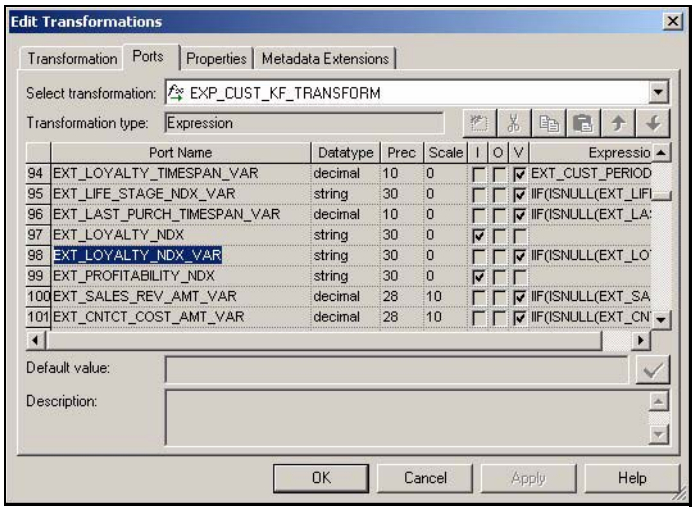
```
If EXT_CUST_PERIOD_CNT_VAR <= 3,

'Customer'


If EXT_CUST_PERIOD_CNT_VAR >= 4

And EXT_CUST_PERIOD_CNT_VAR <= 6,

'Supporter'


If EXT_CUST_PERIOD_CNT_VAR >= 7

And EXT_CUST_PERIOD_CNT_VAR <= 24,

'Sponsor'


If EXT_CUST_PERIOD_CNT_VAR >= 25

And EXT_CUST_PERIOD_CNT_VAR <= 60,

'Patron'


If EXT_CUST_PERIOD_CNT_VAR > 60,

'Advocate'
```

To change the customer Loyalty Index definition

1 In PowerCenter Designer, open the Siebel Applications folder.

2 In Mapplet Designer, open the MPLT_ADI_CUST_KF mapplet to view the default logic.

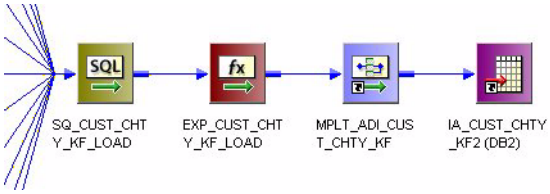
You can find it in the EXT_LOYALTY_NDX_VAR port of the EXP_CUST_KF_TRANSFORM Expression transformation, as shown in the following figure. You can copy this logic into the new Expression and you create and modify it there.



3 Open the final load mapping of the key figure table.

- For regular PLP mappings, in the Configuration for Post Load Processing folder, open the M_PLP_CUST_KF_LOAD mapping.
- If you build your key figures using a flat file upload, in the Configuration for Generic Source folder, open the M_F_CUST_KF_LOAD mapping.

The M_PLP_CUST_KF_LOAD mapping is shown in the following figure.



4 Open the existing Expression transformation located after the Source Qualifier transformation in the final load mapping.

5 In the Expression transformation, enter your modified index band definition logic.

This new index calculation should feed into the ADI. Because it is a precalculated value fed to the ADI, it overrides the ADI's predefined logic.

6 Validate and save your changes.

Profitability Index Definition in the Customer and Customer Channel Type Key Figure Tables

The customer Profitability Index appears in both the Customer key figure table (IA_CUST_KF) and the Customer Channel Type key figure table (IA_CUST_CHTY_KF). In both tables, the current definition to determine whether a customer is rated as Highly Profitable, Profitable, Break-Even, or Unprofitable is as follows:

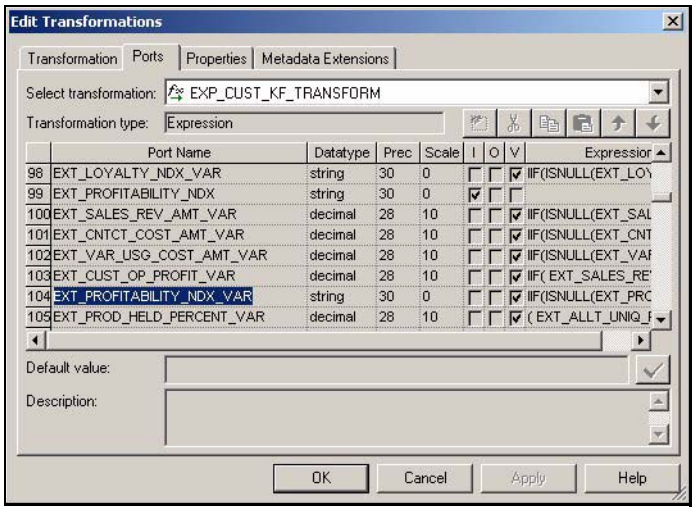
```
If EXT_CUST_OP_PROFIT_VAR > 30,
'Highly Profitable'

If EXT_CUST_OP_PROFIT_VAR > 10
And EXT_CUST_OP_PROFIT_VAR <= 30,
'Profitable'

If EXT_CUST_OP_PROFIT_VAR >= -10
And EXT_CUST_OP_PROFIT_VAR <= 10,
'Break-even'

If EXT_CUST_OP_PROFIT_VAR < -10,
'Unprofitable'
```

To change the customer Profitability Index definition in IA_CUST_KF

1 In PowerCenter Designer, open the Siebel Applications folder.

2 In Mapplet Designer, open the MPLT_ADI_CUST_KF mapplet to view the default logic.

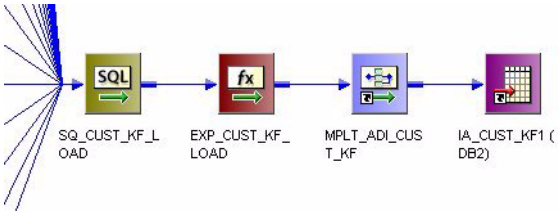
You can find it in the EXT_PROFITABILITY_NDX_VAR port of the EXP_CUST_KF_TRANSFORM Expression transformation, as shown in the following figure. You can copy this logic into the new Expression and you create and modify it there.



3 Open the final load mapping of the key figure table.

- For regular PLP mappings, in the Configuration for Post Load Processing folder, open the M_PLP_CUST_KF_LOAD mapping.
- If you build your key figures using a flat file upload, in the Configuration for Generic Source folder, open the M_F_CUST_KF_LOAD mapping.

The M_PLP_CUST_KF_LOAD mapping is shown in the following figure.



4 Open the existing Expression transformation located after the Source Qualifier transformation in the final load mapping.

5 In the Expression transformation, enter your modified index band definition logic.

This new index calculation should feed into the ADI. Because it is a precalculated value fed to the ADI, it overrides the ADI's predefined logic.

6 Validate and save your changes.

Channel Switch Index in the Customer Key Figure Table

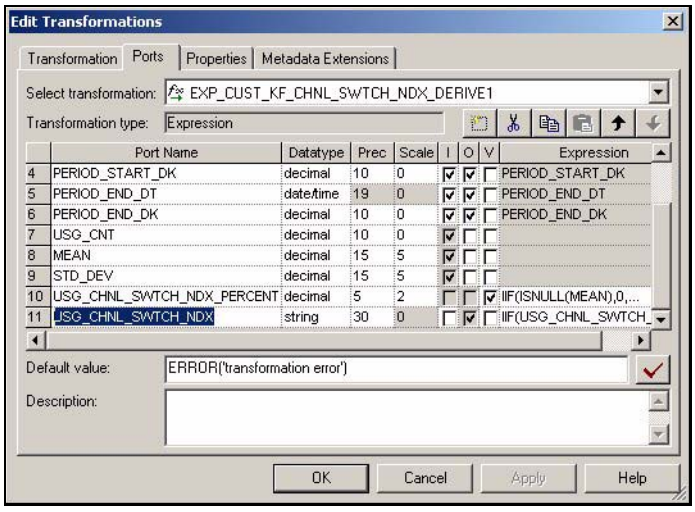
This index is stored in the Customer key figure table, IA_CUST_KF, and refers to a customer's likelihood of switching from one channel to another. The current definition of the usage channel switch index is based on the percentage of standard deviation to the mean of channel usage by each customer. The three bands are determined as follows:

```
If Standard Deviation to the mean of channel usage < 30,

"High Switching Likelihood"


If Standard Deviation to the mean of channel usage > 30 and < 70,

"Moderate Switching Likelihood"


If Standard Deviation to the mean of channel usage > 70,

"Low Switching Likelihood"
```

To change the Channel Switch Index definition

1. In PowerCenter Designer, open the Siebel Applications folder.
2. In Mapplet Designer, open the M_PLP_CUST_KF_CHNL_SWTCH_NDX_DERIVE mapplet to view the default logic.

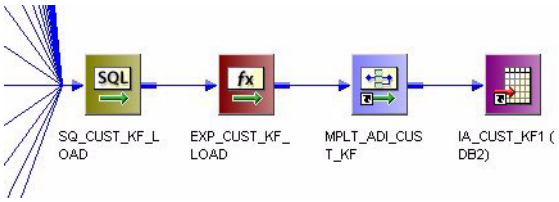
   You can find it in the USG_CHNL_SWTCH_NDX port of the EXP_CUST_KF_CHNL_SWTCH_NDX_DERIVE1 Expression transformation, as shown in the following figure. You can copy this logic into the new Expression and then create and modify it there.



3. Open the final load mapping of the key figure table.

- For regular PLP mappings, in the Configuration for Post Load Processing folder, open the M_PLP_CUST_KF_LOAD mapping.
- If you build your key figures using a flat file upload, in the Configuration for Generic Source folder, open the M_F_CUST_KF_LOAD mapping.

The M_PLP_CUST_KF_LOAD mapping is shown in the following figure.



4 Open the existing Expression transformation located after the Source Qualifier transformation in the final load mapping.

5 In the new Expression transformation, enter your modified index band definition logic.

This new index calculation should feed into the ADI. Because it is a precalculated value fed to the ADI, it overrides the ADI's predefined logic.

6 Validate and save your changes.

Specifying the Customer Location Type Used in the Customer and Customer Channel Type Key Figure Tables

Both the Customer key figure table (IA_CUST_KF) and the Customer Channel Type key figure table (IA_CUST_CHTY_KF) join to the Customer Locations dimension (IA_CUST_LOCS). The Customer Locations dimension stores up to five kinds of locations for each customer—the primary location, the sold-to location, the ship-to location, the bill-to location, and the payer location.

By default, both the Customer key figure table and the Customer Channel Type key figure table join to the Customer Locations dimension by LOCATION_TYPE, and extract only data for locations where LOCATION_TYPE = 'PRIMARY'. You can change this default for one or both tables to use the sold-to, ship-to, or bill-to location instead of the primary location.

To change the customer location type used, edit the SQL in the Source Qualifier of the PLP or Flat File load mapping for the table in question.

Changing the Customer Location Using Regular Post-Load Processing

Follow the instructions in this section if your Customer key figure table (IA_CUST_KF) or Customer Channel Type key figure table (IA_CUST_CHTY_KF) are built from tables in the Siebel Analytics Enterprise Data Warehouse, rather than loaded directly from flat files. If you load key figure tables by using flat files, see the discussion on changing the customer location using a flat file source to build the key figure in Configuration for Post-Load Processing on page 288.

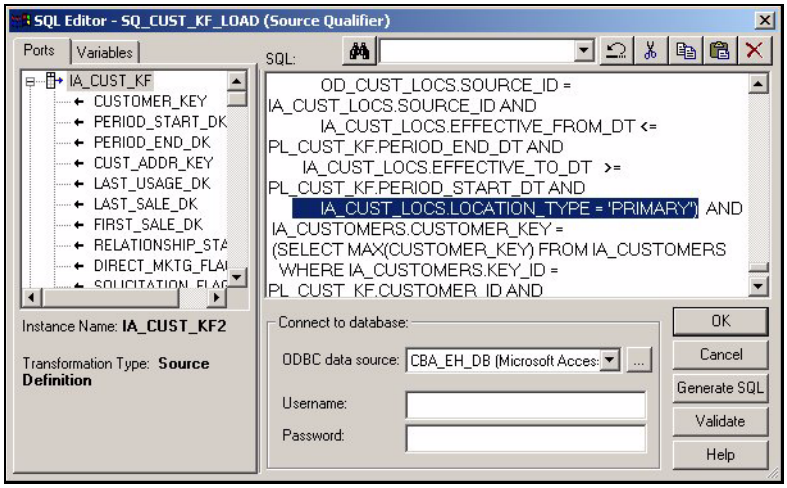
To change the Customer Location type for IA_CUST_KF

1 In PowerCenter Designer, open the Configuration for Post Load Processing folder.

2 Open the M_PLP_CUST_KF_LOAD mapping.

3 Open the SQ_PLP_CUST_KF_LOAD Source Qualifier, and edit the SQL to select the desired customer location details from IA_CUST_LOCS.

The following figure illustrates the SQL that specifies the default customer location type.



For example, if you want to use the sold-to location instead of the primary location, change the filter from:

```
WHERE IA_CUST_LOCS.LOCATION_TYPE = 'PRIMARY'
```

to:

```
WHERE IA_CUST_LOCS.LOCATION_TYPE = 'SOLD-TO'.
```

4 Validate and save your changes.

Changing the Customer Location Using a Flat File Source to Build the Key Figure

Follow the instructions in this section if you use a flat file source to build your key figure tables, rather than extracting data from tables in the Siebel Analytics Enterprise Data Warehouse.

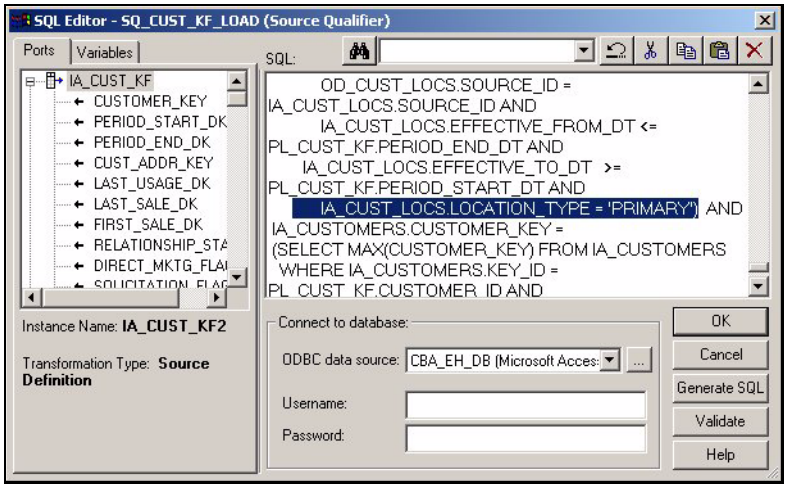
To change the Customer Location type for IA_CUST_KF

1 In PowerCenter Designer, open the Configuration for Universal Source folder.

2 Open the M_F_CUST_KF_LOAD mapping.

3 Edit the SQ_F_CUST_KF_LOAD Source Qualifier, and edit the SQL to select the desired customer location details from IA_CUST_LOCS.

The following figure illustrates the SQL that specifies the default customer location type.



For example, if you want to use the sold-to location instead of the primary location, change the filter from

```
WHERE IA_CUST_LOCS.LOCATION_TYPE = 'PRIMARY'
```

to

```
WHERE IA_CUST_LOCS.LOCATION_TYPE = 'SOLD-TO'.
```

4 Validate and save your changes.

Determining Product Availability in the Customer Key Figure Table

The Customer key figure table, IA_CUST_KF, contains information on the products and services your company considers available to sell to its customers for every period (by default, *period* refers to month when discussing the Customer key figure table). This table stores information on active products both as viewed by your corporation (that is, which products your corporation makes available for sale every period) and as viewed by your customers (that is, a customer's product holding every period).

This section provides instructions for changing the default product extraction mapping to reflect only the products that your corporation makes available within a given time period, as well as for changing the way that customer product holding is defined for a period.

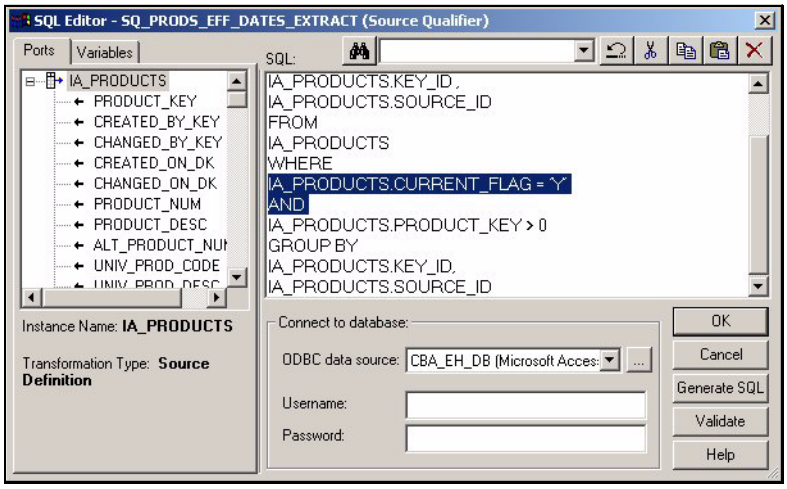
Defining Corporate Product Availability Within a Key Figure Period

To calculate the number of products your corporation makes available for sale in a given period, data is extracted from the Products dimension, IA_PRODUCTS, and loaded into the Customer key figure table, IA_CUST_KF, using the PLP Customer key figure Products Effective Dates Extract mapping. By default, this mapping extracts all the products stored in the Products dimension, meaning that all products appear as being available to all customers. You can change this logic to make certain products unavailable by adding a filter on an IA_PRODUCTS column of your choice in the Source Qualifier of the PLP Customer key figure Products Effective Dates Extract mapping.

To change product availability for IA_CUST_KF

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_CUST_KF_PRODS_EFF_DATES_EXTRACT mapping.
3. Edit the SQL in the SQ_PRODS_EFF_DATES_EXTRACT Source Qualifier to add a filter on an IA_PRODUCTS column of your choice.

   For example, you may decide to base your filter on the DELETE_FLAG or CURRENT_FLAG in IA_PRODUCTS. In the latter case, add a WHERE clause to the SELECT statement in the Source Qualifier to select only records WHERE IA_PRODUCTS.CURRENT_FLAG = 'Y', as shown in the following figure.



4. Validate and save your changes.

Defining Customer Product Holding within a Key Figure Period

The Customer key figure table, IA_CUST_KF, stores the number of active products or services held by each given customer within the period in question. To calculate the number of products or services actively held within the key figure extract period, records are extracted from the Service Provisions dimension if their service order date is on or before the key figure period end date, and if they are marked as current. By default, it is at the individual service level that information is extracted from the Service Provisions dimension and loaded into the Customer key figure table. Therefore, if a given account subscribes to five different services, prepackaged logic loads five product holdings in the key figure table.

However, you may want to change this logic to count product holdings at the account level. In this case, an account subscribing to five different services reports as one product holding rather than five. To accomplish this, you can modify the SQL in the Source Qualifier of the M_PLP_CUST_KF_SRVC_PRVSNS_ALLT_PRODS_DERIVE mapping to use the account open or account reopen date instead of the service order date.

In technical terms, active products or services data is extracted from the Service Provisions dimension, IA_SRVC_PRVSNS. These records are extracted and loaded into the Customer key figure table using the M_PLP_CUST_KF_SRVC_PRVSNS_ALLT_PRODS_DERIVE mapping. The default logic dictates that records extracted for this purpose must meet the following criteria:

- IA_SRVC_PRVSNS.SRVC_ORDER_DK <= IA_CUST_KF.PERIOD_END_DK
- IA_SRVC_PRVSNS.CURRENT_FLAG = 'Y'

To change how active products are calculated in IA_CUST_KF

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_CUST_KF_SRVC_PRVSNS_ALLT_PRODS_DERIVE mapping.
3. Edit the SQL in the SQ_PLP_CUST_KF_SRVC_PRVSNS_ALLT_PRODS_DERIVE Source Qualifier to add a filter on an IA_PRODUCTS column of your choice.

   For example, you may change the portion of the WHERE clause that reads IA_SRVC_PRVSNS.SRVC_ORDER_DK <= IA_CUST_KF.PERIOD_END_DK to read IA_SRVC_PRVSNS.ACCT_OPEN_DK <= IA_CUST_KF.PERIOD_END_DK.
4. Validate and save your changes.

Counting Opened, Closed, and Active Accounts for a Given Period

The Customer key figure table is defined by default as a monthly table (although you can redefine its time interval by following the steps in the discussion on changing the time period for key figure data in Configuration for Post-Load Processing on page 288.) Among other information, the Customer key figure table stores the numbers of accounts that have been opened and closed within the month in question, as well as the numbers of accounts that are active and inactive for that month, which are extracted from the Service Provisions dimension, IA_SRVC_PRVSNS. This section discusses how those numbers are calculated by default, and what you need to do if you decide to change those calculations.

Calculating Accounts Opened

By default, an account is considered to have been opened during the period in question if either the account opening date or the account reopening date falls within that period. If you want to change this criteria (for instance, if you want opened accounts to be based on the account opening date alone) edit the SQL in the Source Qualifier of the PLP Customer key figure Service Provisions Account Open derive mapping, M_PLP_CUST_KF_SRVC_PRVSNS_ACCT_OPEN_DERIVE.

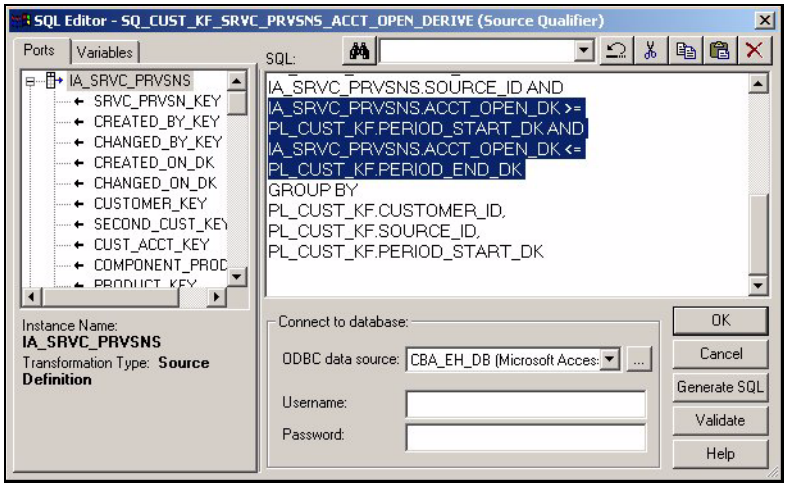
To change the calculation of opened accounts within the period

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_CUST_KF_SRVC_PRVSNS_ACCT_OPEN_DERIVE mapping.
3. Edit the SQL in the SQ_CUST_KF_SRVC_PRVSNS_ACCT_OPEN_DERIVE Source Qualifier to reflect your new criteria.

   The default SQL is:

```
(IA_SRVC_PRVSNS.ACCT_OPEN_DK >= PL_CUST_KF.PERIOD_START_DK AND

IA_SRVC_PRVSNS.ACCT_OPEN_DK <= PL_CUST_KF.PERIOD_END_DK) OR

(IA_SRVC_PRVSNS.ACCT_RE_OPEN_DK >= PL_CUST_KF.PERIOD_START_DK AND

IA_SRVC_PRVSNS.ACCT_RE_OPEN_DK <= PL_CUST_KF.PERIOD_END_DK)
```

   To change the criteria to be based on new open accounts only, edit the SQLs WHERE clause, as shown in the following figure.



4. Validate and save your changes.

Calculating Accounts Closed

By default, an account is considered to have been closed during the period in question if the account closing date falls within that period. If you want to change this criterion—for instance, if you want closed accounts to be based on the account termination date instead of the closing date—edit the SQL in the Source Qualifier of the PLP Customer key figure Service Provisions Account Closed derive mapping, M_PLP_CUST_KF_SRVC_PRVSNS_ACCT_CLOSED_DERIVE.

To change the calculation of closed accounts within the period

1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_CUST_KF_SRVC_PRVSNS_ACCT_CLOSED_DERIVE mapping.
3. Edit the SQL in the SQ_CUST_KF_SRVC_PRVSNS_ACCT_CLOSED_DERIVE Source Qualifier to reflect your new criteria.

   The default SQL is:

   ```
   IA_SRVC_PRVSNS.ACCT_CLOSE_DK >= PL_CUST_KF.PERIOD_START_DK AND

   IA_SRVC_PRVSNS.ACCT_CLOSE_DK <= PL_CUST_KF.PERIOD_END_DK
   ```

4. For example, you may edit the SQLs WHERE clause to read:

   ```
   IA_SRVC_PRVSNS.ACCT_CLOSE_DK >= IA_CUST_KF.PERIOD_START_DK AND
   IA_SRVC_PRVSNS.ACCT_CLOSE_DK <= IA_CUST_KF.PERIOD_END_DK
   ```

5. Validate and save your changes.

Calculating Active Accounts

By default, an account is considered to have been active during the period in question if its opening date falls on or between the period beginning and end dates, and if the account is marked as being current. In technical terms, an account counts as active in the key figure table if its record in the Service Provisions dimension, IA_SRVC_PRVSNS, meets the following criteria:

- IA_SRVC_PRVSNS.ACCT_OPEN_DK >= IA_CUST_KF.PERIOD_START_DK AND
  IA_SRVC_PRVSNS.ACCT_OPEN_DK <= IA_CUST_KF.PERIOD_END_DK
- IA_SRVC_PRVSNS.CURRENT_FLAG = 'Y'

If you want to change this logic—for instance, if you want to include the Account Reopen Date as an acceptable criterion—edit the SQL in the Source Qualifier of the PLP Customer Key Figure Service Provisions Alltime Account derive mapping.

To change the calculation of active accounts within the period

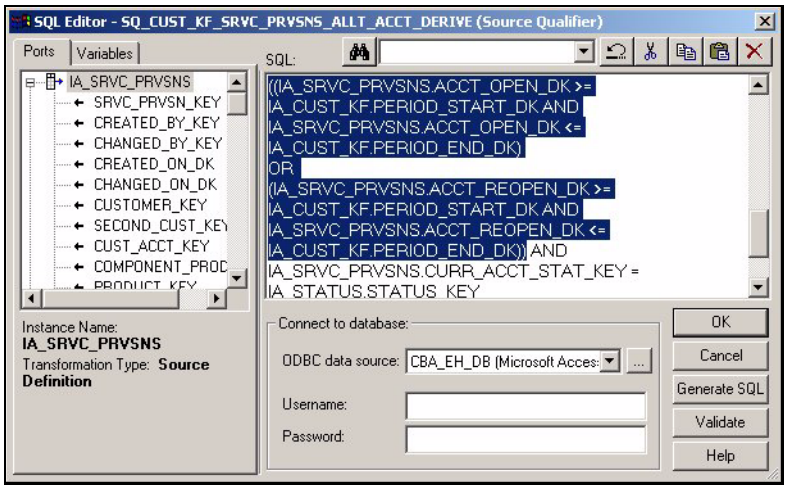
1. In PowerCenter Designer, open the Configuration for Post Load Processing folder.
2. Open the M_PLP_CUST_KF_SRVC_PRVSNS_ALLT_ACCT_DERIVE mapping.

   Edit the SQL in the SQ_CUST_KF_SRVC_PRVSNS_ALLT_ACCT_DERIVE Source Qualifier to reflect your new criteria. The default criteria is:

   ```
   IA_SRVC_PRVSNS.ACCT_OPEN_DK <= PL_CUST_KF.PERIOD_END_DK AND
   ```

```
IA_SRVC_PRVSNS.CURRENT_FLAG = 'Y'
```

To change the default to include reopened accounts, change the SQL's WHERE clause, as shown in the following figure.



3 Validate and save your changes.

Changing Incremental Extract Logic for Sales Booking Lines Data

Most incremental aggregate (NU) tables in the Siebel Analytics Enterprise Data Warehouse are built from the corresponding staging area (T) tables. These incremental aggregate tables are built during the same time as the data warehouse (IA) tables. The Sales Booking Lines incremental aggregate table, IA_SALES_BKGLNS, however, is an exception. It is built from the data warehouse table, IA_SALES_BKGLNS, rather than from a staging table. By default, it extracts the last three days of data. You can change this interval by modifying the SQL in the Source Qualifier of the PLP Sales Booking Lines derive mapping.

To modify the incremental aggregate extract frequency of sales booking lines data

1 In PowerCenter Designer, open the Configuration for Post Load Processing folder.

2 Open the M_PLP_SALES_BKGLNS_DERIVE mapping.

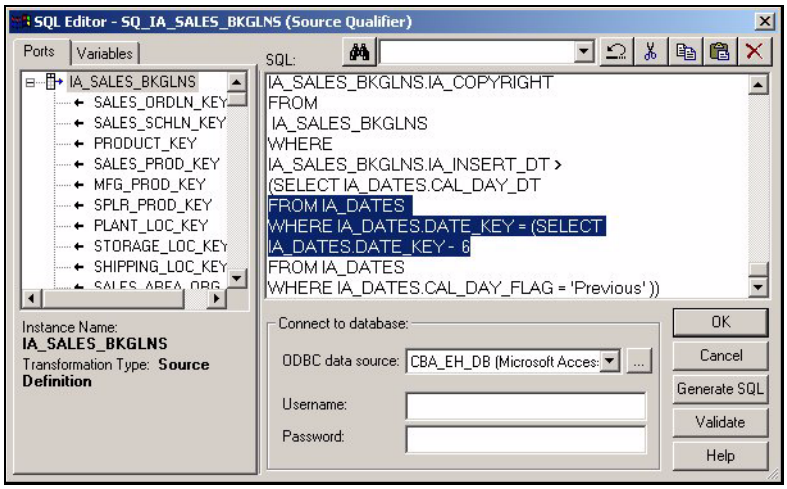
3 Edit the SQL in the SQ_IA_SALES_BKGLNS Source Qualifier to reflect your new criteria.

The default SQL is stated as:

```
IA_DATES.CAL_DAY_DT - 2
```

4 If you want to change the frequency at which the extract happens, you can edit the expression in the SQLs WHERE clause.

For example, if you want to extract the sales booking lines data every week, then you would make the change, as shown in the following figure.



5 Validate and save your changes.

Configuring the Activity Cost ID for Customer Events

Activity costs are associated with Customer Events as well as Usage Transactions. By default, the VAR_ACTIVITY_COST_ID for Customer Events is defined as:

```
CHNL_TYPE_ID||'~'||EVENT_TYPE_ID_VAR||'~'||'CUSTOMER_EVENT'
```

The VAR_ACTIVITY_COST_ID column specifies the unique identifier for each row in the IA_ACTIVITY_COSTS table.

Within the default definition, there are three items that are concatenated to make up the unique identifier. The Channel Type ID refers to channel types such as telephone, Web, in-person, and so on. The Event Type ID refers to event types such as a cancellation, an inquiry, and so on. Customer Event is hard coded to denote that activity cost is strictly for customer events.

The grain at which this column is defined is the grain at which activity costs are stored. If you wish to track costs at a grain different than the default, you need to modify the VAR_ACTIVITY_COST_ID definition.

To redefine the Activity Cost ID

1 In PowerCenter Designer, open the Configuration for Post Load Processing folder.

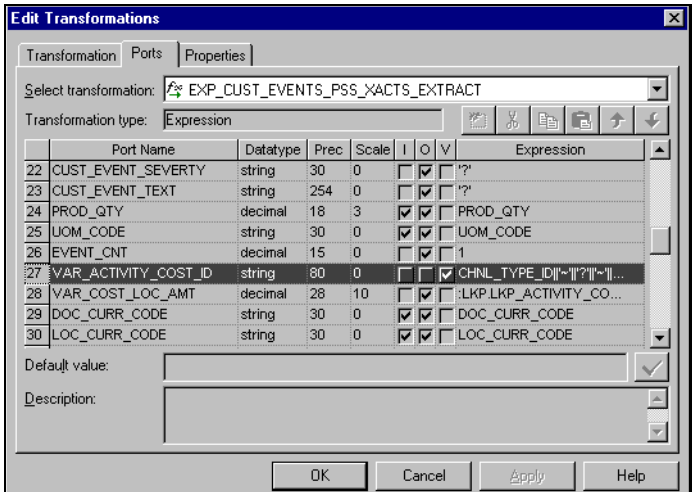
2 Open the applicable mappings.

You must modify two extract mappings:

- M_PLP_CUST_EVENTS_PSS_XACTS_EXTRACT

- M_PLP_CUST_EVENTS_SALES_ORDLNS_EXTRACT

3 Double-click the Expression transformation to open the Edit Transformations box, as shown in the following figure.



4 In the Ports tab, edit the expression for the VAR_ACTIVITY_COST_ID port.

5 Validate and save your changes to the repository.

Understanding the Grain at Which Dollar Amounts and Quantities Are Stored in the IA_SALES_CYCLNS Table

Quantities and dollar amounts may be stored at different grains in the IA_SALES_CYCLNS table, because of their unique ability to be rolled up. Quantities are stored at the parent line level in the IA_SALES_CYCLNS table. (The parent level identifies the bundled package. Each individual product that makes up the package is considered a child.) However, dollar amounts can be stored at either the parent line level or the child line level. Both situations are discussed in this section.

Consider as an example a situation where a customer orders one package, which includes a computer, scanner, printer, and two speakers. In addition to the package, the customer also orders one monitor, which is not included in the package deal. In this case, the sales quantities are listed for the parent line item as well as for each child line items. However, the dollar amounts are only listed for the parent line items; they are not listed for the individual child line items. Table 64 illustrates this example.

Table 64. Storing Dollar Amounts at the Parent Line Level in IA_SALES_CYCLNS

Key_ID	SALES_ ORDER_ NUM	PRODUCT_ ID	ORDHD_ KEY _ID	ORDLN_ KEY _ID	SALES_ QTY	Dollar Amount	Relation
Line_1	1000	Package	1000	Line_1	1	$1000	Parent A
Line_2	1000	Computer	1000	Line_1	1	-	Child A1

Table 64. Storing Dollar Amounts at the Parent Line Level in IA_SALES_CYCLNS

Key_ID	SALES_ ORDER_ NUM	PRODUCT_ ID	ORDHD_ KEY _ID	ORDLN_ KEY _ID	SALES_ QTY	Dollar Amount	Relation
Line_3	1000	Scanner	1000	Line_1	1	-	Child A2
Line_4	1000	Printer	1000	Line_1	1	-	Child A3
Line_5	1000	Speaker	1000	Line_1	2	-	Child A4
Line_6	1000	Monitor	1000	Line_5	1	$400	Parent B (no children)

Consider another example. In this example, a customer orders the same package, which includes a computer, scanner, printer, and two speakers. In addition to the package, the customer also orders one monitor, which is not included in the package deal. In this case, the quantities are provided for the parent and child line item levels. In addition, the dollar amounts are also listed for both the parent and child line item levels. Table 65 illustrates this example.

Table 65. Storing Dollar Amounts at the Child Line Level in IA_SALES_CYCLNS

Key ID	Order #	Product	ORDHD_ KEY_ID	ORDLN_ KEY_ID	Relationship	Qty	Dollar Amount
Line_1	1000	Package	1000	Line_1	Parent A	1	$1000
Line_2	1000	Computer	1000	Line_1	Child A1	1	725
Line_3	1000	Scanner	1000	Line_1	Child A2	1	75
Line_4	1000	Printer	1000	Line_1	Child A3	1	150
Line_5	1000	Speaker	1000	Line_1	Child A4	2	50
Line_6	1000	Monitor	1000	Line_5	Parent B (no children)	1	$400

For more information on parent and child relationships, see Tracking Multiple Products Sold as One Package on page 143.

Filtering Types of Sales Order Dates Loaded into the Order Cycle Time Tables

The Order Cycle Time tables (IA_SALES_CYCHDR and IA_SALES_CYCLNS) store many sales-related dates, which are sourced from the OD_SALES_ORDLNS, OD_SALES_SCHLNS, OD_SALES_PCKLNS, OD_SALES_ORDHLD, and OD_SALES_IVCLNS tables. (See Table 66 for a list of all the dates.) However, most businesses' source systems only store a subset of the sales-related dates available in the Order Cycle Time tables. For all dates that are not stored in your source system, it is recommended that you filter out those order cycle types to prevent unnecessary processing, which could slow down processing performance.

Table 66. Prepackaged Extracted Order Cycle Types

Fact Table	Event Type	Order Cycle Type
OD_SALES_ORDLNS	ORDER ENTRY	CREATED ON, CHANGED ON, ORDERED ON, REQUIRED BY, PURCHASE ORDER DATE, CUSTOMER REQUESTED SHIP DATE, ENTERED ON
OD_SALES_ORDLNS	ORDER BOOKING	BOOKED ON
OD_SALES_ORDLNS	ORDER CANCELLATION	CANCELLED ON
OD_SALES_ORDLNS	ORDER ACKNOWLEDGEMENT	PROMISED ON, ORDER ACKNOWLEDGED ON, CONFIRMED ON
OD_SALES_ORDLNS	INVENTORY DEMAND	INVENTORY DEMAND PLACED ON
OD_SALES_ORDLNS	ORDER VALIDATION	VALIDATED ON
OD_SALES_ORDLNS	ORDER CLOSURE	CLOSED ON
OD_SALES_SCHLNS	SCHEDULING	SCHEDULE REQUEST, PLANNED MANUFACTURING START, PLANNED AVAILABILITY DATE, PLANNED PICKING DATE, PLANNED ASSEMBLY DATE, PLANNED TRANSPORT PLAN DATE, PLANNED PACKING DATE, PLANNED LOADING DATE, PLANNED SHIPPING DATE, PLANNED DELIVERY DATE
OD_SALES_PCKLNS	MANUFACTURING START	MANUFACTURED ON
OD_SALES_PCKLNS	MANUFACTURING FINISH	AVAILABILITY ON
OD_SALES_PCKLNS	PICKING	PICKED ON
OD_SALES_PCKLNS	ASSEMBLY	ASSEMBLED ON
OD_SALES_PCKLNS	PACKING	PACKED ON
OD_SALES_PCKLNS	LOADING	LOADING ON
OD_SALES_PCKLNS	SHIPPING	SHIPPED ON
OD_SALES_PCKLNS	DELIVERY	DELIVERED ON

Table 66. Prepackaged Extracted Order Cycle Types

Fact Table	Event Type	Order Cycle Type
OD_SALES_IVCLNS	INVOICE CREATION	INVOICED ON
OD_SALES_IVCLNS	INVOICE POSTING	POSTED ON, POSTING PERIOD DATE
OD_SALES_ORDHLD	HOLD PLACEMENT	HOLD1 PLACED ON, HOLD2 PLACED ON, HOLD3 PLACED ON, HOLD4 PLACED ON, HOLD5 PLACED ON, HOLD6 PLACED ON, HOLD7 PLACED ON, HOLD8 PLACED ON, HOLD9 PLACED ON
OD_SALES_ORDHLD	HOLD RELEASE	HOLD1 RELEASED ON, HOLD2 RELEASED ON, HOLD3 RELEASED ON, HOLD4 RELEASED ON, HOLD5 RELEASED ON, HOLD6 RELEASED ON, HOLD7 RELEASED ON, HOLD8 RELEASED ON, HOLD9 RELEASED ON
OD_SALES_ORDLNS	ORDER ENTRY	CREATED ON

The Siebel Analytics Enterprise Data Warehouse prepackages a flat file, file_sales_event_types.csv, which holds a list of all different order cycle types that are supported by the Siebel Analytics Enterprise Data Warehouse sales fact tables. See Table 66 for a list of order cycle types.

By default, the PROCESS_FLAG field for all these order cycle types is set to 'Y'. If the PROCESS_FLAG is set to 'Y' for a particular order cycle type, then that order cycle type is loaded into the PL_SLS_EVENT_TYPES staging table, as shown in Figure 35.

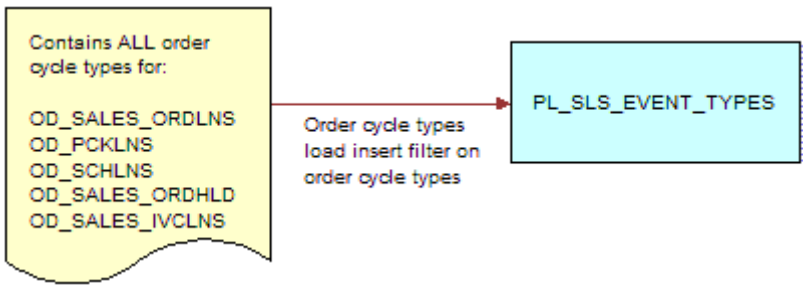


Figure 35. Filtering the Order Cycle Types

After you have loaded the PL_SLS_EVENT_TYPES staging table with the select set of order cycle types, when you extract dates from the OD tables, you only retrieve those dates for which an order cycle types exist in the PL_SLS_EVENT_TYPES staging table. Each OD table (OD_SALES_ORDLNS, OD_SALES_SCHLNS, OD_SALES_PCKLNS, OD_SALES_ORDHLD, and OD_SALES_IVCLNS) goes through the process shown in Figure 36.

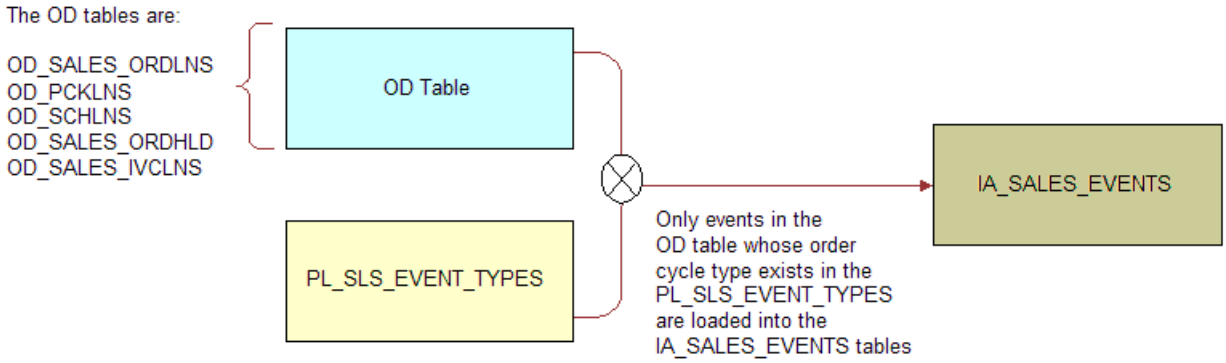


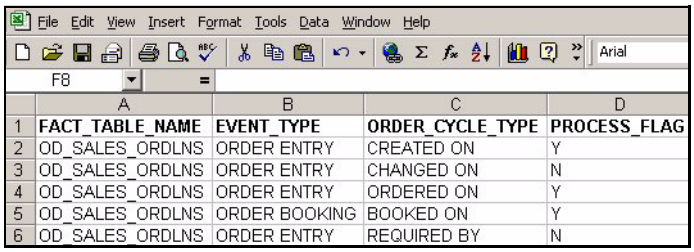
Figure 36. Loading Events into the IA_SALES_EVENTS Table

Therefore, if you wish to weed out certain order cycle types from the Sales Events table, then you must set the PROCESS_FLAG field to 'N' for each order cycle type that you do not want to process. By doing that, the Sales Events table only pulls events whose order cycle type exists in the staging table.

To filter the order cycle types extracted

1 Open the file_sales_event_types.csv file.

The following figure provides an illustration of the file's layout.



	A	B	C	D
1	FACT_TABLE_NAME	EVENT_TYPE	ORDER_CYCLE_TYPE	PROCESS_FLAG
2	OD_SALES_ORDLNS	ORDER ENTRY	CREATED ON	Y
3	OD_SALES_ORDLNS	ORDER ENTRY	CHANGED ON	N
4	OD_SALES_ORDLNS	ORDER ENTRY	ORDERED ON	Y
5	OD_SALES_ORDLNS	ORDER BOOKING	BOOKED ON	Y
6	OD_SALES_ORDLNS	ORDER ENTRY	REQUIRED BY	N

2 Modify the PROCESS_FLAG field.

3 Save the file.

Integrating Additional Types of Sales Order Dates into the Order Cycle Time Tables

It is recommended that you do not change the structure of any of the data warehouse tables, with the exception of the Cycle Time tables (IA_SALES_CYCHDR and IA_SALES_CYCLNS). In particular, the Order Cycle Time table may not store every sales-related date. As a result, you want to create a new date field in this table to accommodate your additional dates.

To add more dates, you should first understand how the Order Cycle Times tables are populated. The PL_SALES_CYCHDR and the PL_SALES_CYCLNS tables contains a filter that restricts particular dates from being loaded into the IA_SALES_CYCHDR and IA_SALES_CYCLNS tables as shown in Figure 37. Thus, if you want to change the dates loaded into the Cycle Time tables (IA_SALES_CYCHDR and IA_SALES_CYCLNS), then you have to modify the M_PLP_SALES_CYCHDR_LOAD and the M_PLP_SALES_CYCHDR_LOAD mappings that take the dates from the PL_* tables and load them into the IA_* tables.

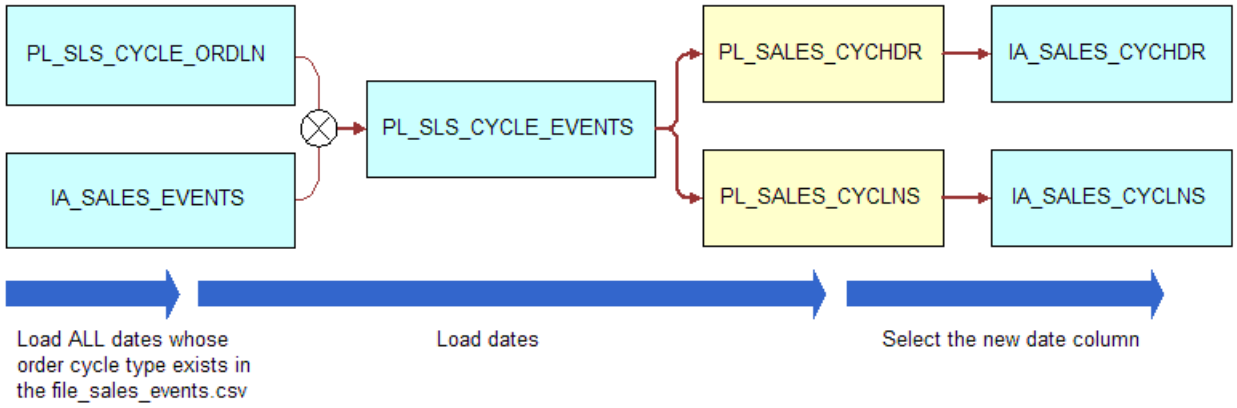


Figure 37. Adding Dates to the Cycle Time Table

To add dates to the Cycle Time table load

1. Be sure that the date is already being extracted and stored in IA and OD data warehouse tables.
2. Make sure that events are generated for this date column.
   a. Open the flat file file_sales_event_types.csv.
   b. Modify the value for PROCESS_FLAG field to 'Y' for the order cycle type to which the date belongs.

   For more information on this procedure, see the discussion on filtering types of sales order dates loaded into the order cycle time tables in Configuration for Post-Load Processing on page 288. For example, if you are adding a new date VALIDATED_ON_DT, then you have to make sure that the order cycle type for this date is being processed.
3. In PowerCenter Designer, open the Configuration for Post Load Processing folder.

4 In Warehouse Designer, modify the table definition for the target table to verify that it has a field to store this date.

   For example, if you are loading the Validated on Date in the IA_SALES_CYCLNS table, then you need to create a new column VALIDATED_ON_DT.

   The PL_SALES_CYCLNS table pumps data into the IA_SALES_CYCLNS table; therefore, you have to modify the target definition of the IA_SALES_CYCLNS table.

5 In Source Analyzer, modify the table definition of the source table to include this new column.

   Continuing with the example, you would include the VALIDATED_ON_DT column in the IA_SALES_CYCLNS source table.

6 Create the table in the database with the new table structure.

   TIP: If you have already loaded data in the IA_SALES_CYCLNS table, then make sure that you backup the data before you recreate this table.

   As previously mentioned, you have already taken care of loading the IA_SALES_EVENTS table with the events for the new order cycle type VALIDATED_ON_DT.

   The cycle times ADI—the mapping's ADI that takes data from the PL_SLS_CYCLE_EVNTS and loads it into the PL_SALES_CYCHDR and PL_SALES_CYCLNS tables—has logic to compute the VALIDATED_ON_DT for the purpose of cycle time calculations. After events are loaded into the Cycle Time table, the calculations are automatically performed and the data is loaded into the intermediate table PL_SALES_CYCLNS.

7 Modify the mapping M_PLP_SALES_CYCLNS_LOAD to select this new column from the source PL_SALES_CYCLNS table and load it to the target table IA_SALES_CYCLNS table.

Making the Cycle Time Mappings Incremental

The Cycle Time PLP mappings are prepackaged to do full extracts. However, it is strongly recommended that you run PLP mappings on an incremental basis. The following sessions should be modified to make sure that the data is extracted on an incremental basis from the base fact tables:

- S_M_PLP_SALES_ORDLNS_SALES_ORDLN_EXTRACT
- S_M_PLP_SALES_ORDLNS_SALES_SCHLN_EXTRACT
- S_M_PLP_SALES_ORDLNS_SALES_PCKLN_EXTRACT
- S_M_PLP_SALES_ORDLNS_SALES_IVCLN_EXTRACT
- S_M_PLP_SALES_ORDLNS_SALES_ORDHLD_EXTRACT

Most prepackaged incremental extracts are set up to extract the last three days of activity. To set up the incremental extract for the past three days, you would have to modify the SQL in the session.

To make the PLP mappings an incremental mapping

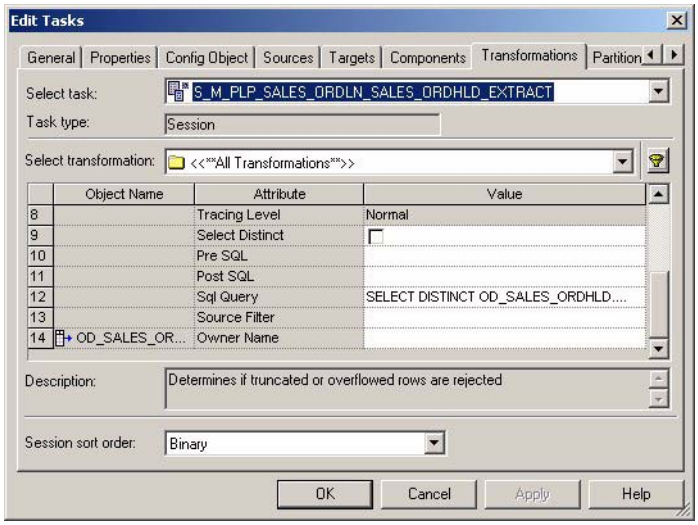
1 In PowerCenter Workflow Manager, open the applicable source configuration folder.

2 Open the applicable extract session to open the Edit Tasks box.

In the Transformations tab, insert the condition in the WHERE clause in the SQL Query field. As a sample, the default WHERE clause for the S_M_PLP_SALES_ORDLNS_SALES_ORDHLD_EXTRACT session is:

```
WHERE

 OD_SALES_ORDHLD.CURR_IA_UPDATE_DT > (SELECT CAL_DAY_DT FROM IA_DATES WHERE
CAL_DAY_FLAG = 'Previous')

AND

NOT EXISTS (SELECT 'X' FROM PL_SLS_EVTS_ORDLNS WHERE
PL_SLS_EVTS_ORDLNS.SALES_ORDLN_ID = OD_SALES_ORDHLD.SALES_ORDLN_ID AND
PL_SLS_EVTS_ORDLNS.SOURCE_ID = OD_SALES_ORDHLD.SOURCE_ID)

AND

OD_SALES_ORDHLD.ORDHD_KEY_ID IS NOT NULL
```

as shown in the following figure.



3 Validate and save your changes to the repository.

17 Storing, Extracting, and Loading Additional Data

This chapter discusses the methodology for storing additional data in the data warehouse. In addition, it gives general procedures for extracting and loading new data.

This chapter includes the following topics:

Overview for Storing, Extracting, and Loading Additional Data

Siebel Analytics Enterprise Data Warehouse contains an open (source-independent) data model. When you set up your data warehouse, you find that the packaged mappings load most of your source data. However, you may have additional data that does not have a storage place. To accommodate this scenario, Siebel Analytics Enterprise Data Warehouse has set aside extension columns in most warehouse tables that act as placeholders for additional data.

Extension columns make it possible to extend any fact or dimension table without changing the schematic structure of the Siebel Analytics Enterprise Data Warehouse data model, or making modifications to the load mapping, because load mappings already include the extension column loading logic. In addition, extension columns have far less impact on the database size than building entirely new tables, which could also have implications when upgrading. Each of the following sections provides you with information on the preferred methods of integrating different types of data without affecting database size or impeding functionality.

TIP: It is recommended that you use the extension columns, instead of changing the data model structure. As long as you do not change the data model structure, you can implement upgrades without losing any of your customizations. In addition, it is easier to provide technical support for unchanged data models.

Classifying Additional Data

There are four types of data that you might want to store in the Siebel Analytics Enterprise Data Warehouse:

- **Attributes.** Attributes include descriptive data that allow you to look at metrics under different circumstances. Examples of attributes are product name, product description, and sales order number.
- **Amounts and Quantities.** Amounts and quantities are two types of metrics, which are sometimes referred to as facts. Amounts are monetary values, such as costs, revenues, profits, and so on. Quantities are counts of items, such as the number of sales orders, number of products sold, and number of backlogged sales orders.
- **Dates.** Dates are self-explanatory.
- **Reference Data.** Reference data is data that is used to perform a calculation or a transformation. An example of reference data is exchange rates. When converting currency, the system looks up the appropriate exchange rate to convert from one currency type to another.

Depending on the type of data that you want to store in the data warehouse, you may choose a particular type of table, as well as an extension column with a particular data type. Siebel Analytics Enterprise Data Warehouse prepackages extension columns in fact tables, dimension tables, and class tables, where each type of table contains different types of extension columns.

The following sections suggest the types of tables and columns to use when storing particular types of data in your data warehouse.

Types of Extension Columns

Each fact and dimension table has a unique set of extension columns. The following sections outline the types of extension columns for each type of table.

There are two types of dimension tables. Table 67 provides descriptions of the types of extension columns present in both types of dimension tables. The only difference between the two types is that one type of dimension table has more columns than the other.

Table 67. Extension Columns in Dimension Tables

Type of Extension Column	Description	Number of Columns
Code	You can use these columns to store codes.	3 to 10
Code Name	You can use these columns to store code names. Code names are looked up to decipher cryptic codes.	3 to 10
Text	You can use these columns to store text. This is a pass-through column.	3 to 10

Fact tables generally have more types of extension columns than dimension tables. This is due to the nature of a star schema. Table 68 provides descriptions of the extension columns present in the two types of fact tables. The only difference between the two types is that one type of fact table has more columns than the other.

Table 68. Extension Columns in Fact Tables

Type of Extension Column	Description	Number of Columns
Code	You can use these columns to store codes.	3
Code Name	You can use these columns to store code names. Code names are looked up to decipher cryptic codes.	3
Text	You can use these columns to store text. This is a pass-through column.	3 or 8
Date Key	You can use these columns to store date keys. The ADI transforms any dates into Julian dates.	3
Amount	You can use these columns to store monetary values. Amounts come in triplets—document, local, and group currencies. For information on the three types of currencies, see Working with Document, Local, and Group Currencies on page 246.	9 (3 Document, 3 Group, 3 Local)
Quantity	You can use these columns to store units.	3 or 6
Dimension Key	You can use these columns to store dimension keys. You can use this to join the fact table to a new dimension table.	3 or 5

In the following sections, you learn how to incorporate attributes, metrics, reference data, and dates into fact, dimension, and class tables.

Incorporating Additional Attributes into the Data Model

There are two different types of attributes—domained and free text. Domained attributes are restricted to a set of values. An example of a domained attribute is a State column that requests a state within the United States. The State column's domain of values consists of all the states in the United States.

NOTE: Siebel Analytics Enterprise Data Warehouse domain values are different from domained attributes. Domained attributes are source values, not values necessarily loaded into Siebel Analytics Enterprise Data Warehouse; they are a set of values that can be used for a particular field. On the other hand, domain values are Siebel Analytics Enterprise Data Warehouse values for particular fields that are used to create metrics. These values are called Siebel Analytics Enterprise Data Warehouse domain values, which are generically referred to as domain values. For more information on domain values, see Working with Domain Values on page 268. For a list of domain values, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

In addition to domained attributes, your source system may also provide free text attributes, which can have any value. Unlike domained attributes, where there is a select set of values, there are no restrictions on the value of free text attributes. An example of a free text attribute is a description column in which users enter a description. Descriptions can vary widely; there is no standard set of values for descriptions.

Depending on the type of attribute you want to store, domained or free text, there are different recommendations on how to incorporate them into your data warehouse. The following sections describe how to store each type of attribute.

Determining the Type of Table to Use for Attributes

Before you can store domained or free text attributes in the data warehouse, you must first decide what table the data goes in. The following sections describe the options available to you; the options are listed in the order that they are recommended.

First Recommendation—Store Additional Attributes in an Existing Dimension Table

The preferred option is to store additional attributes is an existing dimension table that primarily stores this type of data. There are a few requirements that must be met before you can do this:

- The attribute stored in the dimension table must be at the same base grain as the table or higher. Changing to a lower base grain may negatively affect joins to other tables, and it is therefore recommended that you do not change the base grain of the table.
- The relationship between the dimension table base grain to the attribute can be one-to-one, or many-to-one, but not one-to-many. For example, the IA_PRODUCTS table's base grain is the product number. So, for example, you can incorporate a store location code column, which takes the base grain of product, and matches it with several store attributes, changing the grain from products code to store code. However, you can add a color column to track that attribute of the product without changing the grain of the table.

- All fact tables containing the facts associated with this attribute must be joined to the class table to allow analysis of the facts by this attribute.
- Given the limited number of extension columns, you must be selective when choosing data that you want to incorporate into the data warehouse. If you require more extension columns than are provided, keep attributes that are the most closely associated with the dimension table in that table, and place all other attributes in the other tables. For example, if you had a dimension table that covered the attributes of storage capacity for your warehouse, and you had both additional storage and location attributes to incorporate, you would choose to create a new location table, rather than split the storage capacity information. For information on creating new tables, see Table Formats on page 341.

Second Recommendation—Store Additional Attributes in a Class Table

If there are no extension columns available in an existing dimension table, or if there is not an existing dimension table closely associated with the attribute data, incorporate the attribute into a class table.

Class tables are dimension tables that store dimension records that are similar in structure. Class tables are sometimes referred to as multi-class tables, because each class table can store multiple classes of information that relate to the subject of the table. For example, there is a class table called Business Organizations that stores predefined different classes of organizations within a business. Sample categories include—Sales Area Organization, Control Area, Sales Geography, and Purchase Organization. The organizations are defined by region, channel, and so on. Although there are predefined category types, you can add or customize category types in the Class tables.

The following describes the available class tables. Within each class table, there are corresponding, predefined categories. For example, within the Business Organizations class table there are categories for Business Area (a vertical market in which a product is competing) and Company (each legal entity of the enterprise by location), and for the Business Locations class table there is Storage (Storage location for the enterprise), and Plant (manufacturing location for the enterprise). Additional categories can be added if required.

For a list of class tables and their corresponding categories, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

When selecting a class table to store your attribute data, check the following:

- The attribute data must be associated with the subject area of the class table.
- Each fact table to which you want to join this attribute must have a dimension key available to the dimension class table.

Third Recommendation—Store Additional Attributes in a Fact Table

If there are no existing dimension tables available, and there is no class table that satisfies your requirements, the third choice is to store the attribute in a fact table. However, note that for each attribute that you store, you could potentially change the grain of the fact table, and consequently affect your data integrity.

For example, assume you want to incorporate an attribute into the Purchase Orders data warehouse table, such as Supplier Location. You can incorporate the location attribute directly into the fact table without affecting the grain, because it still references the same set of suppliers. However, if you store an attribute that further defines the suppliers by the warehouses to which they deliver, the grain of the fact table may be affected and may not supply you with the high-level information about suppliers that you require.

Therefore, check the following when storing an additional attribute in a fact table:

- If the attribute requires a change in the fact table grain, you must check the integrity of the data.
- The attribute data must be associated with the subject area of the fact table.
- If you are only loading one or two attributes, you can store them directly in the fact table without greatly affecting processing time. However, as the table size grows, the response time increases. Therefore, this third option is not recommended if you have numerous attributes to incorporate.

Fourth Recommendation—Store Additional Attributes in a New Dimension Table

If you have a group of related attributes to load, but they cannot be incorporated into an existing dimension table or class table, then your final option is to create a new dimension table. For example, if you have a group of attributes that all relate to Profit Centers, then you can create a Profit Center dimension table. It is recommended that you create dimension tables for attributes that can be grouped by a business area. For information on creating new tables see Table Formats on page 341.

Do not create a dimension table to store a disparate set of attributes. If you decide to create a new dimension table, use the same structure and naming conventions as the prepackaged dimension tables. Structurally, the new dimension table must contain columns such as primary key, Source ID, Key ID, fact keys, dimension keys, and so on. In addition, with each of these columns, there are naming conventions. For information on naming conventions, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

Determining the Type of Extension Column to Use for Attributes

When you have determined which type of table to store the attribute in, you then have to decide what type of extension column to use. The extension column you use depends on the type of attribute. There are two types of attributes data—free-text and domained.

Free Text Attributes

It is recommended that you use a Text extension column as a placeholder for free text attribute data. Text extension columns are available in fact, dimension, and class tables. The Text data type extension column offers the most character space; they are varchar (254).

Domained Attributes

It is recommended that you use Code Name and Code extension columns as a placeholder for domained attribute data. Referring back to our sample domained attribute State, the State column's domain of values is restricted to the states in the United States, which can be denoted by a code. For example, California can be denoted by CA. In this case, the code name is California and the code is CA. With domained attributes, you can issue codes to each possible value, because you know the possible values for this variable. However, do not use codes for columns that have too many values, because it could significantly slow the processing time. For example, even though you might use a Code and Code Name extension column for a state, you might not use it for cities.

If a code does not exist in the IA_CODES table and you want to use a lookup for the code name, then you must add the code to the IA_CODES table. Three possible scenarios occur when working with codes and code names. Each of the following scenarios are also described in Figure 39. The three scenarios are described in the sections that follow.

Storing Codes and Code Names in the Siebel Analytics Enterprise Data Warehouse.

If the code and code name pair reside in the IA_CODES table and you load only the code column, the ADI resolves the code name through a lookup. (See Figure 38.) This is the way to enforce domained values. For information on adding new codes to the IA_CODES table, see Creating a Codes Mapping on page 361.

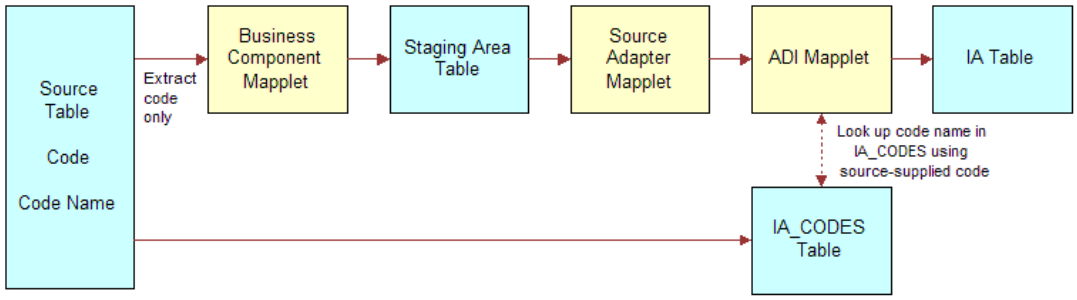


Figure 38. Storing Codes and Code Names in an IA Table

Storing Codes Without Code Names in the Siebel Analytics Enterprise Data Warehouse

If the source supplies only the code without the code name, and the code and code name pair do not exist in the IA_CODES table, then the ADI tries to resolve the code name by a lookup. However, because the code and code name do not exist in the IA_CODES table, no data is retrieved. As a result, the ADI loads only the code into the IA table. This method does not enforce domained values.

Storing Code Names Without Codes in the Siebel Analytics Enterprise Data Warehouse

If the source supplies the code name without the code, then the ADI loads only the code name into the IA table. (See Figure 39.) This method does not enforce domained values.

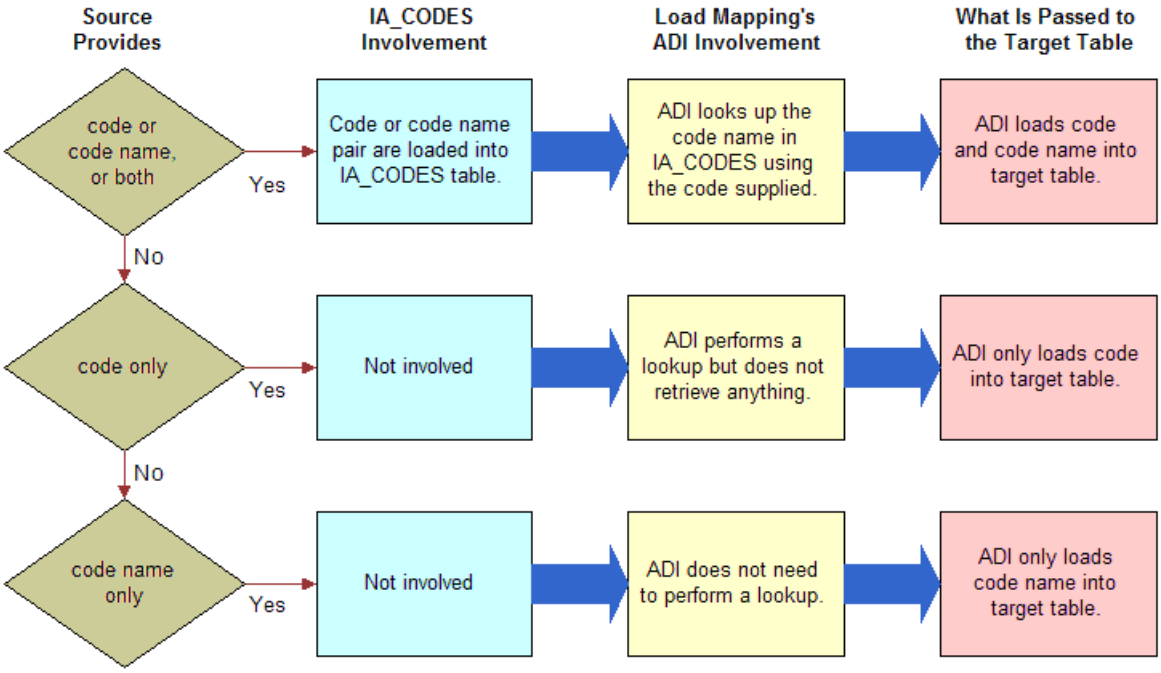


Figure 39. ADIs Handling of Code and Code Names

Storing Additional Attributes in the Data Warehouse

Siebel Analytics Enterprise Data Warehouse comes with several prepackaged attributes. Because you may want to incorporate additional attributes to the data warehouse, this section provides detailed instructions for storing free text and domained attributes.

There are a variety of scenarios that occur when storing a new attribute. Depending on where the data resides, you can take particular steps to incorporate new attributes and make them available in the data warehouse and your front-end schema. Generally speaking, there are three major areas where data resides—Source database, staging table, and Siebel Analytics Enterprise Data Warehouse. Figure 40 illustrates the three scenarios and the components that are affected when trying to store the new data.

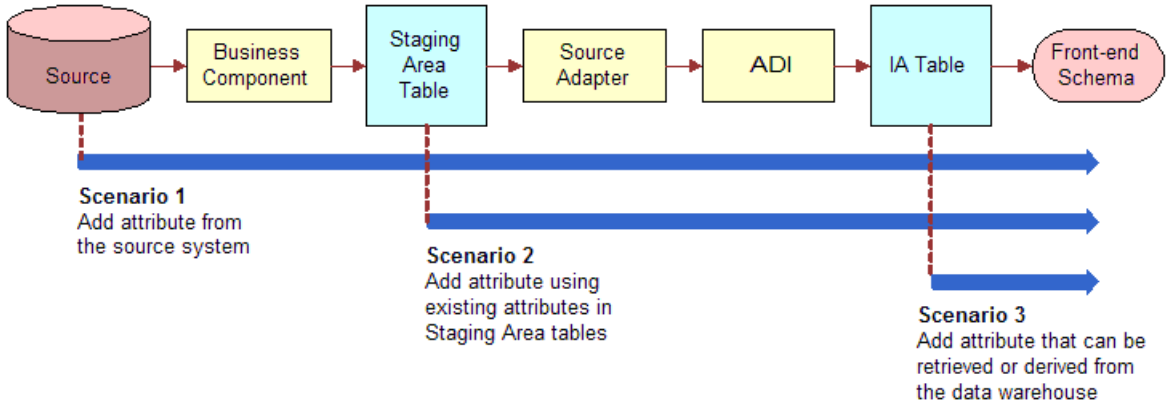


Figure 40. Scenarios for Storing New Attributes

In the sections that follow, you can find procedures for storing new attributes for each of the three scenarios. Each of the steps in the procedure corresponds to larger topics described in later sections of this chapter. The steps provide you with a high-level overview, and the larger topics provide the details.

To store additional attributes from the source system

1 If applicable, load the code and code name into the IA_CODES table.

This step applies only if you are storing domained attributes. In addition, perform this step only if the code and code name are not already loaded into the IA_CODES table. For information on creating a codes mapping, see Creating a Codes Mapping on page 361.

2 Modify the existing extract mapping or create a new extract mapping to extract this new information from the source system.

For information on modifying the existing Business Component mapplet of an extract mapping, see Business Component Configuration on page 374. For information on creating a new extract mapping, see Creating an Extract Mapping on page 354.

TIP: You can perform calculations in the Business Component mapplet of the extract mapping or in the Source Adapter mapplet of the load mapping. However, it is not recommended that you perform performance-expensive calculations in the extract. This protects you from interfering with your source transaction system. For these types of calculations, it is recommended that you perform them in the Source Adapter mapplet in the load mapping. One of the major reasons why Siebel Analytics Enterprise Data Warehouse splits the extract process from the load process into two separate mappings is to minimize the amount of time tying up your source transaction system.

3 Modify the existing load mapping to take the data from the staging table and load it into the data warehouse.

For information on modifying the existing Source Adapter mapplet of a load mapping, see Source Adapter Configuration on page 375. For information on creating a new load mapping, see Creating a Load Mapping on page 358.

TIP: If you map the data to a staging table's extension column, you must determine the type of extension column to use. For information on the type of extension column to use for attribute data, see Determining the Type of Extension Column to Use for Attributes on page 328.

4 Incorporate the new attribute into your front-end schema.

To store additional attributes derived from other staging area objects

1 Modify the existing load mapping to take the data from the staging table and load it into the data warehouse.

For information on modifying the existing Source Adapter mapplet of a load mapping, see Source Adapter Configuration on page 375. For information on creating a new load mapping, see Creating a Load Mapping on page 358.

TIP: If you map the data to a staging table's extension column, you must determine the type of extension column to use. For explicit information on the type of extension column to use for attribute data, see Determining the Type of Extension Column to Use for Attributes on page 328.

2 Incorporate the new attribute into your front-end schema.

NOTE: This procedure can also be used to store additional metrics. For more about storing metrics, see Storing Additional Metrics in the Data Warehouse on page 333.

To store additional attributes derived from data warehouse objects

■ Incorporate the new attribute into your front-end schema.

NOTE: This procedure can also be used to store additional metrics. For more about storing metrics, see Storing Additional Metrics in the Data Warehouse on page 333.

Incorporating Additional Metrics into the Data Model

There are two different types of metrics—quantities and amounts. Quantity metrics are basically counts of items. For example, the number of dining tables ordered is a quantity. Amount metrics are monetary values. For example, the cost of a dining table is an amount metric.

Determining the Type of Table to Use for Metrics

Regardless of the type of metric, it is recommended that you load metric data into fact tables.

However, before loading a metric into a fact table, perform the following checks:

- You must make sure that the grain of the table is the same grain as the new metric. If the table and the data are not at the same grain level, it is strongly recommended that you do not change the grain of a table to match the grain of the metric data. By doing so, you may negatively affect joins to other tables, as well as reports.
- Determine the type of extension column you use to store the metric. Different extension columns are provided for both the quantity and amount type of metric. Extension columns for quantity are identified by the QTY suffix, and extension columns for amount are identified by the AMT suffix.
- The metric data must be associated with the subject area of the fact table.

In addition to determining the appropriate fact table, you must also determine which type of extension column to load the metric data into. The following section suggests the type of column to use for each type of metric data.

Determining the Type of Extension Column to Use for Metrics

It is recommended that you use a Quantity extension column (*_QTY) to store quantity metric data. As well, it is recommended that you use an Amount extension column (*_AMT) to store amount metric data.

Storing Additional Metrics in the Data Warehouse

Siebel Analytics Enterprise Data Warehouse comes with several prepackaged metrics. Because you may want to store additional metrics to the data warehouse, this section provides instructions for storing Quantity and Amount metrics.

There are a variety of scenarios that occur when incorporating a new metric. Depending on where the data resides, you can take particular steps to incorporate the new metrics and make them available in your data warehouse and front-end schema. Generally speaking, there are three major areas where data resides—source database, staging table, and Siebel Analytics Enterprise Data Warehouse. Figure 41 illustrates the three scenarios and the components that are affected when trying to store the new data.

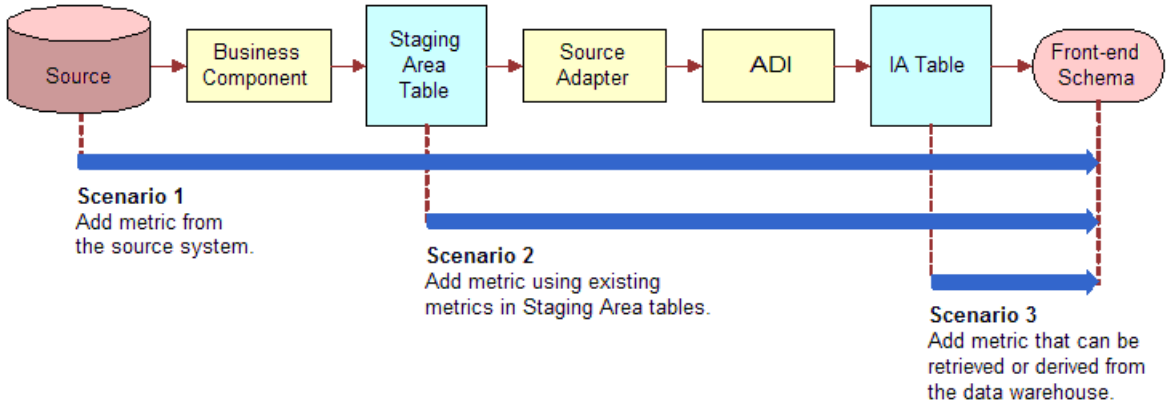


Figure 41. Scenarios for Storing New Metrics

In the sections that follow, you can find procedures for storing new metrics given each of these three scenarios. Each of the steps in the procedure correspond to larger topics discussed in later sections of this chapter. The steps provide you with a high-level overview, and the larger topics provide the details.

To store additional metrics from the source system

1 Modify the existing extract mapping or create a new extract mapping to extract this new information from the source system.

For information on modifying the existing Business Component mapplet of an extract mapping, see Business Component Configuration on page 374. For information on creating a new extract mapping, see Creating an Extract Mapping on page 354.

TIP: You can perform calculations in the Business Component mapplet of the extract mapping or in the Source Adapter mapplet of the load mapping. However, it is not recommended that you perform large calculations in the extract. This protects you from interfering with your source transaction system. For these types of calculations, it is recommended that you perform them in the Source Adapter mapplet in the load mapping. One of the major reasons why Siebel Analytics Enterprise Data Warehouse splits the extract process from the load process into two separate mappings is to minimize the amount of time tying up the source transaction system.

2 Modify the existing load mapping to take the data from the staging table and load it into the data warehouse.

For information on modifying the existing Source Adapter mapplet of a load mapping, see Source Adapter Configuration on page 375. For information on creating a new load mapping, see Creating a Load Mapping on page 358.

TIP: If you map the data to a staging table's extension column, you must determine the type of extension column to use. For explicit information on the type of extension column to use for attribute data, see Determining the Type of Extension Column to Use for Metrics on page 333.

3 Incorporate the new metric into your front-end schema.

NOTE: You can also store additional metrics derived from other staging area tables or from data warehouse objects. The steps for these tasks are identical to those for storing new attributes. See the procedures in Storing Additional Attributes in the Data Warehouse on page 330.

Incorporating Additional Reference Data into the Data Model

Reference data is used solely for calculation and transformation purposes; it does not appear in reports or queries. For example, if you are trying to convert from one currency to another, you need an exchange rate. The exchange rate is reference data that allows you to perform a calculation on one currency to derive another.

Reference Data is stored in one of three tables; it is recommended that you store your reference data in one of the following:

- **Exchange Rate table (IA_XRATES).** The Exchange Rates table stores exchange rates for converting currencies to and from document, local, and group currencies.
- **Codes table (IA_CODES).** The codes table stores codes and code names that allow users to look up code names using a code.
- **Standard Cost table (IA_STD_COSTS).** The Standard Cost table stores standard cost information for every product. For most of the facts, the ADI performs a lookup into the Standard Cost table using the product number and retrieves the unit standard cost, also known as the cost per unit, in local currency. This standard cost multiplied by quantity gives the total cost in local currency (COST_LOC_AMT). The total cost is then multiplied by the exchange rate to convert it to the total cost in group currency (COST_GRP_AMT). For information on document, local, and group currencies, see Working with Document, Local, and Group Currencies on page 246.

Storing Additional Reference Data in the Data Warehouse

If your reference data resides in the source database, then you can use mappings to extract the data and load it into the corresponding table. For example, there are codes mappings that extract different types of codes from different source systems. For information on how to create a new codes mapping to extract codes and load them into the IA_CODES table, see Creating a Codes Mapping on page 361.

Incorporating Additional Dates into the Data Model

There are a number of dates that are populated when you initialize Siebel Analytics Enterprise Data Warehouse. For example, purchase orders use an ordered-on-date, a requested-date, a required-on-date, and so on. However, for date types that are not captured by the prepackaged extract and load processes, you can use extension date columns.

To map an additional date column to an extension date column

1. Map the additional date from the source into one of the extension columns in the staging area.

   To do this, you must modify an extract mapping. Within the Business Component:

   a. Include the source table that supplies the date as a source table in the extract mapping.

   b. Map the data from the source table to the Source Qualifier transformation.

   c. Map the data from the Source Qualifier to the Mapplet Output Object (MAPO).

   d. Map the data from the Business Component to the Expression transformation. If your date is not in the proper date format, you can convert it using this Expression transformation.

   e. Map the data from the Expression transformation to the staging table.

   After this is complete, the corresponding load mapping pulls the date data and loads it into the corresponding IA table.

2. You also must modify the front-end schema to report on this date.

Integrating Data from Sources Without Prepackaged Business Adapters

Siebel Analytics Enterprise Data Warehouse provides universal business adapters for sources that have no prepackaged business adapter. These sources are referred to as universal sources. The universal business adapter extracts, transforms, and loads data from these sources. However, unlike the prepackaged business adapters for particular source systems, the universal business adapters are prepackaged to perform general transformations, such as enabling Type II Slowly Changing Dimensions, date data type conversion logic, and deletion logic. The philosophy behind the universal source architecture is to extract and load data from any source. As a result, you may need to configure the data or prepackaged extract, or both, and load mappings to incorporate your unique transformation logic. For more information on the universal source data requirements, and an architectural overview of the ETL processes, see the following sections.

Extracting Universal Source Data

All universal source data goes through a similar extract mapping that is made up of a flat file template, Source Qualifier, Expression transformation, and staging table. The basic extract mapping for universal source flat files is shown in Figure 42.

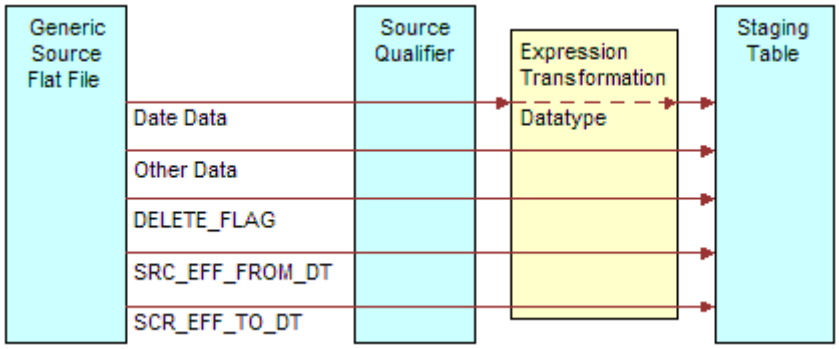


Figure 42. Basic Extract Mapping for Universal Source Data

NOTE: Unlike other adapters, the universal adapter requires values for the DELETE_FLAG, SRC_EFF_FROM_DT, and SRC_EFF_TO_DT.

Each of the extract mapping components is described in the following sections.

Universal Source Flat File Template

Given the vast differences in source data, Siebel Analytics Enterprise Data Warehouse does not prepackage transformation logic other than date transformations. Therefore, there are a few requirements on how to present source data when using the universal ETL components. Siebel Analytics Enterprise Data Warehouse provides a flat file template that delineates what data to provide, as well as the format in which the data must be presented. Each flat file template corresponds to exactly one data warehouse table. The following are the specifications for the source flat file:

- Supply all the universal source data in a comma delimited flat file (*.csv). Because flat files have only number and string data types, all dates must be provided in the String (14) data type. For example, you can use '200112310000' for December 31, 2001.
- Specify the values for the Key ID and Source ID in the flat file. Unlike prepackaged sources, these values are not formed in the Source Adapter for universal sources.
- Preset the record deletion flags. The delete flag can have the values 'Y' or 'N'.
- If applicable, supply the source Effective To and Effective From dates in the flat file. If the effective dates are not supplied, Siebel Analytics Enterprise Data Warehouse inserts default values—January 01, 1899 for the Effective From date and January 01, 3714 for the Effective To date.

- Each flat file template has 10 system columns that are not used. These columns are named RESERVED_1, RESERVED_2, and so on. These are not customization columns for your use; Siebel Analytics Enterprise reserves these columns for future development.

  NOTE: The IA_COPYRIGHT column is populated during an initial load (M_Z_INITIAL_LOAD) from one of the prepackaged CSV files.

Each of the extract mapping components performs the following functions:

- **Source Qualifier.** The Source Qualifier provides the means for extracting the data from the source.

  NOTE: You can use the Source Qualifier to join multiple source flat files from the same database platform. However, if you are sourcing from multiple sources that belong to different database platforms you cannot use a Source Qualifier; you must use a Joiner Transformation to join the sources.

- **Business Component.** The Business Component is omitted from the extract mapping for universal sources as there are no transformations to perform other than the date data type change. Therefore, you can either transform the data prior to it being extracted by the universal adapter or incorporate transformation logic within the universal adapter.

- **Expression Transformation.** As previously mentioned, the Expression transformation only converts dates with the string(14) data type into a timestamp data type, which is the standard data type used in the data warehouse tables.

- **Staging Table.** The staging tables are the target tables for the extract mappings. Each staging table mirrors its corresponding load control table structure. However, it does not have control columns, such as CURR_KEY, IA_INSERT_DT, IA_UPDATE_DT, and other CURR_* columns. If a load control table is not available, the staging table mirrors the IA table, except that instead of *_KEY and *_DT columns, it uses *_ID and *_DT columns.

After data is extracted, the data then goes through a load mapping, described in the following section.

Load Process for Universal Source Data

All universal source data goes through a similar load mapping that is made up of a staging table, Source Adapter mapplet, ADI mapplet, and data warehouse table. The basic load mapping for universal source flat files is shown in Figure 43.

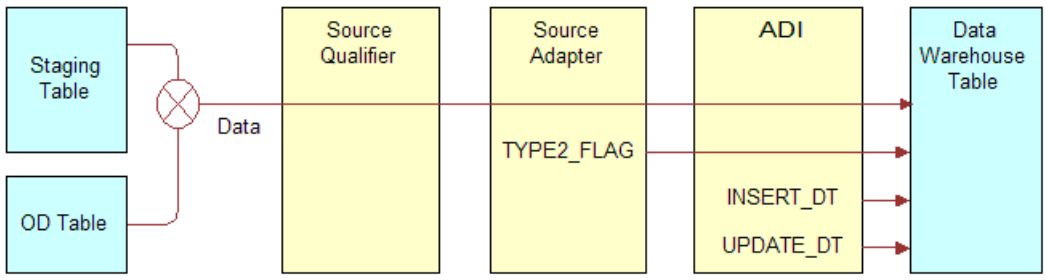


Figure 43. Basic Load Mapping for Universal Source Data

Each of the load mapping components perform as follows:

- **Staging Table.** The staging table serves as the source table for the load mapping. Each staging table mirrors its corresponding load control table structure. However, it does not have control columns, such as CURR_KEY, IA_INSERT_DT, IA_UPDATE_DT, and other CURR_* columns. If a load control table is not available, the staging table mirrors the IA table, except that instead of *_KEY and *_DT columns, it uses *_ID and *_DT columns.
- **OD Table.** The load control table determines if the source records loaded in the staging table are new, updated, or unchanged. New records are inserted into the data warehouse table. Updates to existing records overwrite the existing records if Type I Slowly Changing Dimensions are enabled. Updates to existing records are inserted if Type II Slowly Changing Dimensions are enabled. Unchanged records are rejected from the data warehouse because they already exist.
- **Source Qualifier.** The Source Qualifier joins the staging table and load control table and provides the Source ID and Key ID.
- **Source Adapter.** The Source Adapter mapplet adds the Type II Flag to the sourced data. If you want to incorporate additional data transformation logic, add it within this object.
- **ADI.** The ADI mapplet creates values for the INSERT_DT and UPDATE_DT columns. In addition, the ADI also performs the update strategy that either inserts new records, updates existing records, or rejects records from the load.
- **Data Warehouse Table.** The resulting data warehouse table (IA) stores the data for end user querying purposes. Although not illustrated in Figure 43, each time the IA table is loaded, the ADI also reloads the OD table after truncating it. Only the most recent snapshot of all column values are stored in OD tables; they do not store historical information.

18 Integrating Additional Data

This chapter provides procedural information for creating and modifying the ETL components to populate data in the data warehouse tables as well as creating and modifying the PLP components to populate aggregate and key figure tables.

This chapter includes the following topics:

Overview of Integrating Additional Data

As you perform gap analysis, you may find that you need to create new components to either extract, load, transform, or store additional data. This chapter discusses how to perform each of these tasks, including use of the table extension columns. You should read Chapter 17, "Storing, Extracting, and Loading Additional Data," before this chapter.

This chapter assumes you are familiar with PowerCenter Designer and its component tools. For more information on PowerCenter Designer and its component tools, see the *PowerCenter/PowerMart Designer Guide*.

As you begin creating new components, you should be consistent in the way you name your objects. For all custom-built objects, the naming convention should be prefixed with Z_. For example, if you create a new profile table for customers, then you might name it as Z_CUSTOMER_PROFILE. For a list of naming conventions used for objects prepackaged by Siebel Analytics Enterprise Data Warehouse, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.

Table Formats

You can create additional tables in Siebel Analytics Enterprise Data Warehouse to meet your business requirements. This section lists the specific formats for various tables, which are commonly used standards.

Although there are exceptions to the rules, these are the formats you should follow when creating new tables. All tables, with the exception of new staging tables, should be created in the Siebel Analytics Enterprise Applications folder in the PowerCenter repository. Only staging tables are created within the prepackaged, source-specific configuration folders, such as Configuration for Oracle 11i and Configuration for PeopleSoft.

The following topics discuss additional table formats you can create in Siebel Analytics Enterprise Data Warehouse. They describe how to create an entirely new table, as well as how to make a new table using a copy of an existing table:

Each of the table types in Siebel Analytics Enterprise Data Warehouse has a specific format and naming convention that must be followed to create tables.

You can also create tables with two additional table formats, Profile tables and Key Figure tables.

- Because of the complex nature of Key Figure tables and the mappings they require, Key Figure tables are discussed separately in the topic Creating and Populating Custom Key Figure Tables on page 364.
- Profile tables are covered separately in the topic Creating a Profile Table Using Domain Values on page 352.

Fact Table Format

A fact table that is linked to other tables requires a surrogate key. If the fact table is used in other mappings the surrogate key becomes the primary key. If the table is not used in other mappings, the primary key is a composite of the KEY_ID and SOURCE_ID. The naming convention for fact tables is IA_[SUBJECT], for example, IA_AP_XACTS.

The following is the order of the columns, data type, and precision for fact tables.

1. **Surrogate key.** The data type is decimal (10,0) or (15,0) (for example, in the fact table for Accounts Payable transactions, IA_AP_XACTS, the surrogate key is AP_XACTS_KEY).

   NOTE: This column does not apply to all fact tables.

2. **Dimensions keys.** The data type is decimal (10,0) or (15,0) (for example, the GL_ACCOUNT_KEY in IA_AP_XACTS).
3. **Date Key columns.** The data type is decimal (15,0) in Julian format with a DK suffix (for example, CREATED_ON_DK is a date key in the IA_AP_XACTS fact table).
4. **Amount columns.** The data type is decimal (28,10) ordered by currency type and with an AMT suffix (for example, AP_GRP_AMT in IA_AP_XACTS).
5. **Quantity columns.** The data type is decimal (18,3) with a QTY suffix (for example, XACT_QTY).
6. **Code columns.** The data type of all codes is varchar (30) with a CODE suffix (for example, UOM_CODE). There are two kinds of codes:

- The first code type contains units of measure and currency.
- The second code type is code-name pairs where the code and the code name are both stored, such as states where there is both CA and California.

7 **Other fact attributes.** The data type of the attribute is subjective to the kind of attribute. For example, for ACCT_DOC_NUM, the data type is varchar (30), while for ACCT_DOC_ITEM the data type is decimal (15,0).

8 **Description columns.** The data type for the description columns is varchar (254) or varchar (255) with a DESC suffix (for example, GL_ACCOUNT_DESC).

9 **Name columns.** The data type of the name columns is varchar (254) or varchar (255) with a NAME suffix (for example, GL_ACCOUNT_NAME).

10 **Extension columns.** Extension columns have the same data type and precision as their column type (for example, amount, quantity, or code). The order for the extension columns is as follows:

- The dimensional key naming convention is [FACT TABLE ABBREVIATION]_DIM[SEQUENTIAL NUMBER]_KEY (for example, APXT_DIM1_KEY in IA_AP_XACTS).
- The format for data keys is [FACT TABLE NAME]_DATE[SEQUENTIAL NUMBER]_DK (for example, APXT_DATE1_DK in IA_AP_XACTS).
- The quantity naming convention is [FACT TABLE NAME]_[SEQUENTIAL NUMBER]_QTY (for example, APXT_1_QTY in IA_AP_XACTS).
- The amount naming convention is [FACT TABLE NAME]_DOC[SEQUENTIAL NUMBER]_AMT (for example, APXT_DOC1_AMT in IA_AP_XACTS).
- The code and name pair naming convention is [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_CODE and [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_NAME (for example, APXT_ATTR1_CODE and APXT_ATTR1_NAME in IA_AP_XACTS).
- The text column naming convention is [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_TEXT (for example, APXT_ATTR1_TEXT in IA_AP_XACTS). (The data type for text columns is varchar (254).)

11 **Control columns.** The data type and precision of the control columns varies. KEY_ID is data type varchar (80), SOURCE_ID is data type varchar (30), and IA_COPYRIGHT is data type varchar (254). IA_INSERT_DT and IA_UPDATE_DT are dependent on the database; datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2.

Load Control Table Format for Fact Tables

There are different load control table formats for both fact tables and dimension tables. This section focuses specifically on load control tables for fact tables. The first column in a load control table is the Current Key (CURR_KEY). It is the Current Key that links the load control table with the surrogate key in the fact table. The naming convention for a load control table is OD_[SUBJECT], for example, OD_AP_XACTS.

The following is the order of the columns, data type, and precision for load control tables for facts.

1 **Current key (CURR_KEY).** The data type for the current key is decimal (15, 0).

2 **ID columns.** The data type for ID columns is varchar (80), with a suffix of ID, which corresponds to the *_KEY columns in the fact (IA) table (for example, SALES_ORDLN_ID in the OD_SALES_PCKLNS load control table corresponds to the SALES_ORDLN_KEY).

NOTE: These columns only exist in an OD_* load control table if the corresponding IA_* warehouse table contains a surrogate key. For example, the IA_SALES_ORDLNS warehouse table uses a surrogate key for SALES_ORDLNS_KEY column; therefore, OD_SALES_ORDLNS table has a SALES_ORDLN_ID column.

3 **Date columns.** The data type of date columns is datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2, with a suffix of DT (for example, CREATED_ON_DT). This column corresponds to the *_DK columns in IA fact tables.

4 **Amount columns.** The data type for the amount columns is decimal (28, 10) with a suffix of AMT (for example, NET_GRP_AMT). As with the amount columns in the fact table, they are in order of currency—group, local, and document.

5 **Codes.** The data type of all codes is varchar (30), with a suffix of CODE (for example, UOM_CODE). The code columns correspond with the fact table, therefore if the fact table has both units of measure and currency as well as code-name pairs, the load control table has these values too.

6 **Other attributes.** The data type varies to suit the attribute, but corresponds to what is found in the fact table.

7 **Extension columns.** As found in the fact table and in the same order:

- Dimensional IDs with a format of [FACT_TABLE_NAME]_DIM[SEQUENTIAL_NUMBER]_ID, such as APXT_DIM1_ID.
- Date with a format of [FACT_TABLE_NAME]_DATE[SEQUENTIAL_NUMBER]_DT, such as APXT_DATE1_DT.
- Quantity columns with a format of [FACT_TABLE_NAME]_[SEQUENTIAL_NUMBER]_QTY, such as APXT_1_QTY. Amount columns with a format of [FACT_TABLE_NAME]_DOC[SEQUENTIAL_NUMBER]_AMT, such as APXT_DOC1_AMT.
- Code and Name pairs with a format of [FACT_TABLE_NAME]_ATTR[SEQUENTIAL_NUMBER]_CODE, or [FACT_TABLE_NAME]_ATTR[SEQUENTIAL_NUMBER]_NAME, such as APXT_ATTR1_CODE and APXT_ATTR1_NAME.
- Text columns with a format of [FACT_TABLE_NAME]_ATTR[SEQUENTIAL_NUMBER]_TEXT, such as APXT_ATTR1_TEXT. The data type for text columns is varchar (254).

8 **Control columns.** The data type and precision of the control columns varies. KEY_ID is varchar (80), SOURCE_ID is data type varchar (30), and IA_COPYRIGHT is data type varchar (254). IA_INSERT_DT and IA_UPDATE_DT are dependent on the database; datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2.

Dimension Table Format

Dimension tables always start off with a surrogate key. The surrogate key is always the primary key for the table. The naming convention for dimension tables is the same as fact tables, IA_[SUBJECT], for example, IA_PRODUCTS.

The following is the order of the columns, data type, and precision for dimension tables:

1. **Surrogate key.** The data type is decimal (10, 0) or decimal (15, 0) (for example, PRODUCT_KEY).
2. **Dimension keys.** The data type for the dimension key is decimal (15,0) (for example, the VISITOR_KEY in the IA_CUSTOMERS dimension table).
3. **Date keys**. The data type is decimal (15, 0) in Julian format and the suffix DK (for example, CREATED_ON_DK).
4. **Attribute columns.** Attribute columns can be of several different types, including descriptive and code, and the data type varies depending on the type. If it is a descriptive type such as name, the data type is varchar (254); if it is a code, it is varchar (30); all others, including number types, are either varchar (80) or varchar (30).
5. **Hierarchy columns.** The data type for codes is varchar (30), and for names the data type is varchar (254). The number of the hierarchy columns is based on the number of nodes in the hierarchy and there is no specific limit. Hierarchy columns are entered in code or name pairs, with the naming convention of [DIMENSION TABLE ABBREVIATION]_HIER[SEQUENTIAL NUMBER]_CODE, or [DIMENSION TABLE ABBREVIATION]_HIER[SEQUENTIAL NUMBER]_NAME. Each pair uses the same number, corresponding to its level in the hierarchy, for example, PROD_HIER1_CODE and PROD_HIER1_NAME in the IA_PRODUCTS dimension table are at the first level.

   NOTE: This column is only applicable to a few dimension tables.
6. **Code and Description columns.** The data type for codes is varchar (30), and the data type for descriptions is varchar (254), with respective CODE and DESC suffixes (for example, DIVISION_CODE and DIVISION_DESC).
7. **Extension columns.** There are principally two kinds of extension columns in the following order:
   - Extension columns for additional code or name pair attributes have naming conventions of [DIMENSION TABLE ABBREVIATION]_ATTR[SEQUENTIAL NUMBER]_CODE and [DIMENSION TABLE ABBREVIATION]_ATTR[SEQUENTIAL NUMBER]_NAME, with data types varchar (30) and (254), and CODE and NAME suffixes (for example, PROD_ATTR1_CODE and PROD_ATTR1_NAME).
   - Extension columns for additional textual information have a naming convention of [DIMENSION TABLE ABBREVIATION]_ATTR[SEQUENTIAL NUMBER]_TEXT, with data type varchar (254), and TEXT suffix (for example, PROD_ATTR1_TEXT and PROD_ATTR1_NAME).
8. **Control columns.** Control columns include CURRENT_FLAG and DELETE_FLAG varchar (1), KEY_ID varchar (80), SOURCE_ID varchar (30), and IA_COPYRIGHT varchar (254), IA_INSERT_DT, IA_UPDATE_DT, EFFECTIVE_FROM_DT and EFFECTIVE_TO_DT.where EFFECTIVE_FROM_DT and EFFECTIVE_TO_DT are added to handle slowly changing dimensions and are dependent on the database; datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2.

Load Control Table Format for Dimension Tables

The first key is the current key, which points to the surrogate key in the dimension IA table. The naming convention for load control tables is OD_[SUBJECT], for example, OD_GL_ACCOUNTS.

The following is the order of the columns, data type, and precision for load control tables for dimension tables:

1. **Current key (CURR_KEY).** The data type for the current key is decimal (15, 0).
2. **Dimension ID keys.** The data type for dimension ID keys corresponding with the IA dimension table is varchar (80).
3. **Date.** The data type of date columns is datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2, with a suffix of DT (for example, CREATED_ON_DT). This column corresponds to the *_DK columns in IA fact tables.
4. **Other attributes.** The data type, length and precision is as found in the corresponding dimension table (IA table).
5. **Hierarchy columns.** The data type for codes is varchar (30,0) and for name the data type is varchar (254). The sequential number of the hierarchy column matches those code or name pairs found in the IA dimension table.
6. **Code and Description columns.** The data type for codes is varchar (30) and for descriptions the data type is varchar (254). The code-name pairs are found in the corresponding IA dimension tables.
7. **Extension columns.** Extension columns corresponds with those found in the IA dimension table.
8. **Control columns.** The data types correspond with those control columns found in the dimension table, including IA_COPYRIGHT and a composite primary key composed of the KEY_ID, SOURCE_ID, and SRC_EFF_FROM_DT. IA_INSERT_DT and IA_UPDATE_DT are dependent on the database; datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2.

   NOTE: The SRC_EFF_FROM_DT column is not available in all dimension tables.

Aggregate Table Format

Aggregate tables aggregate facts across a dimension or set of dimensions. You can have several aggregate tables linking to the same tables, but aggregating data for different time periods. for example, there are three payroll aggregate tables (IA_PAYROLL_A1, IA_PAYROLL_A2, and IA_PAYROLL_A3) that aggregate gross deductions and net pay by day, month, and quarter respectively.

The first keys in an aggregate table are the Dimension keys that link the table to the dimension tables for which it is aggregating data. As illustrated in the IA_PAYROLL_A1 example, the naming convention for aggregate tables is IA_[SUBJECT]_A[SEQUENTIAL NUMBER].

The following is the order of the columns, data type, and precision for IA aggregate tables.

1. **Dimension keys.** The data type is decimal (10, 0) or decimal (15, 0). Dimension keys link the aggregate table to the dimension tables that could be used to analyze the aggregate table. for example, EMPLOYEE_KEY links IA_PAYROLL_A1 with the Employee dimension table. There can be several dimension keys drawing information from several different dimension tables.
2. **Date columns.** The data type is decimal (15,0), with a DK suffix (for example, PAYROLL_A1_DK).
3. **Amount columns.** The data type is decimal (28,10), with an AMT suffix (for example, GRP_AMT).
4. **Quantity columns.** The data type of (18,3), with QTY suffix (for example, PRA1_1_QTY).
5. **Currency Code columns**. The data type is varchar (30), with a code suffix (for example, LOC_CURR_CODE).
6. **Extension columns.** Format for Extension columns varies and is as follows:
   - Dimensional keys with a format of [FACT TABLE NAME]_DIM[SEQUENTIAL NUMBER]_KEY, such as APXT_DIM1_KEY.
   - Date keys with a format of [FACT TABLE NAME]_DATE[SEQUENTIAL NUMBER]_DK, such as APXT_DATE1_DK.
   - Quantity columns with a format of [FACT TABLE NAME]_[SEQUENTIAL NUMBER]_QTY, such as APXT_1_QTY.
   - Amount columns with a format of [FACT TABLE NAME]_DOC[SEQUENTIAL NUMBER]_AMT, such as APXT_DOC1_AMT.
   - Code and Name pairs with a format of [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_CODE, or [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_NAME, such as APXT_ATTR1_CODE and APXT_ATTR1_NAME.
   - Text columns with a format of [FACT TABLE NAME]_ATTR[SEQUENTIAL NUMBER]_TEXT, such as APXT_ATTR1_TEXT.

   NOTE: The data type for text columns is varchar (254).
7. **Control columns.** Control keys include CREATED_BY_KEY, CHANGED_BY_KEY, CREATED_ON_DK, CHANGED_ON_DK, DELETE_FLAG, IA_COPYRIGHT, and a composite primary key containing the KEY_ID and SOURCE_ID. IA_INSERT_DT and IA_UPDATE_DT are dependent on the database; datetime (26, 6) for SQL Server; date (26,6) for Oracle; and timestamp (26,6) for DB2.

Incremental Table Format

Incremental tables follow the same exact format of the corresponding fact or load control table from which they source information. Therefore, to create an incremental table, you must first create or locate the fact or load control table that is being used as the source. The fact or load control table can then be copied and renamed, as shown in Using Existing Tables to Create New Tables on page 350. Set the session parameter so that the table gets truncated before each load. The naming convention for incremental tables is NU_[SUBJECT], for example, NU_GL_ACTIVITY.

Staging Table Format

Unlike the other tables that are created in the Siebel Analytics Enterprise Applications folder, the staging tables are source dependent and are created in the prepackaged source specific configuration folders in the PowerCenter Repository. These tables have the following requirements:

- The default database for staging tables is DB2, however it can be changed as necessary. The primary key for staging tables is a composite of the KEY_ID and SOURCE_ID.
- The naming convention for staging tables is T[SOURCE]_[NAME].
- For Oracle 11i, the staging table is TI_USERS.

The purpose of the staging tables is to speed data extraction by temporarily storing extracted source data that are transformed in an Expression transformation in preparation for the source-independent Siebel Analytics Enterprise Data Warehouse. Therefore, the order of the columns, and contents is determined by the corresponding source data.

For example, in the staging table TO_USERS, the first eight columns contain attributes about user information, some of which include CREATED_BY, LAST_UPDATED_BY, USER_NAME, and EMAIL_ADDR, and so on, and which are then followed by extension columns and the last two control columns that comprise the primary key, the KEY_ID and the SOURCE_ID.

In contrast, TS_USERS, which is the staging table for user information in SAP, has several of columns containing attribute information about users including DEPARTMENT_CODE, DELEGATEE_NAME, REGION_CODE, and LANGUAGE_CODE. The only significant similarities are that the attributes are followed by extension columns and that the primary key is usually a composition of KEY_ID and SOURCE_ID columns. Sometimes, however, the primary key also includes the effective date for the record.

Creating New Tables

After reviewing the required format and naming convention for the tables you want to create, you can create the actual table itself. The process of creating the table is the same regardless of the kind of tables you want to create. The important difference is the naming convention and to maintain their respective formats. The following procedure provides information on creating a target table. If the table is used as a source as well, you can copy the table into the Sources folder and modify its properties as necessary.

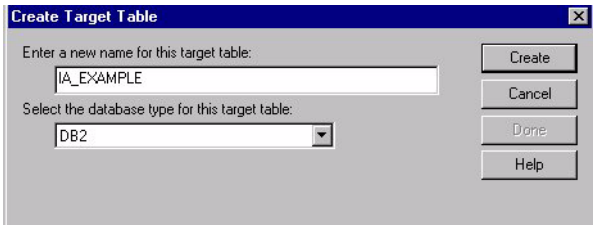
To create new target tables

1. Open PowerCenter repository and select the Siebel Analytics Enterprise Applications folder, if you are creating any kind of table except a staging table.

   If you are creating a staging table, open the corresponding prepackaged source specific configuration folder. The rest of the procedure remains the same.

2. Select Tools > Warehouse Designer.
3. Select Targets > Create to open the Create Target Table window.
4. Following the appropriate naming convention for the type of table, enter a new name for the table.

5 Select the database type from the list window and click Create, as shown in the following figure.



NOTE: The required database type for all tables is DB2.

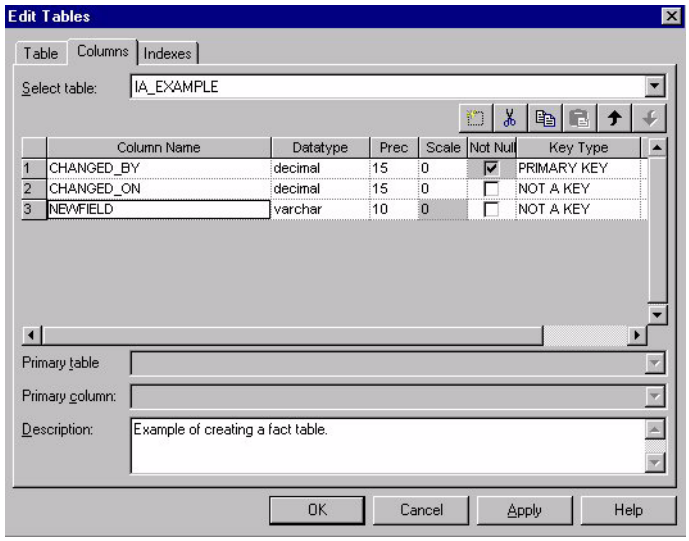
6 When the table appears in the Warehouse Designer, click Done to close the Create Target Table window.

7 Double-click the newly created table to open the Edit Table window.

8 Enter a description in the Description window on the Table tab.

9 Click the Columns tab to add columns to the table.

Click the Add Column icon and add columns, data type, and set precision as required, as shown in the following figure.



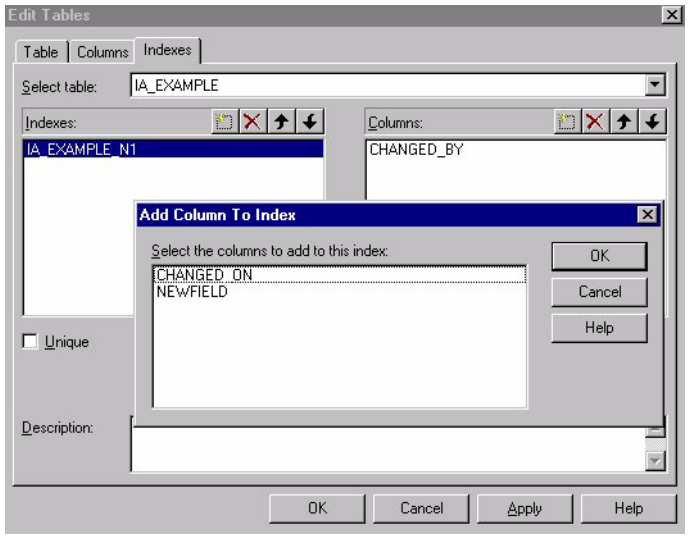
10 Click the Indexes tab, and then click the New Insert button to enter the table name with the appropriate index suffix in the Indexes window.

NOTE: The format for the index is [TABLE NAME]_N[SEQUENTIAL NUMBER] for nonunique indexes, or [TABLE NAME]_U[SEQUENTIAL NUMBER] for unique indexes. For example, IA_EXAMPLE_N1 or IA_EXAMPLE_U1. Although indexes are not required, they help speed processing time by connecting nonunique tables.

11 Click the New Insert button in the Columns window to open the Add Column to Index dialog box.

- Highlight the column from the list that you want the index to look locate, and click OK.

- You must repeat clicking the New Insert button, highlight the appropriate column, and then click OK for each column you want to add, as shown in the following figure.



12 If the index is unique, select the Unique check box.

Do not check the box for nonunique indexes.

13 Click OK to return to the Warehouse Designer.

The new table is automatically added to the repository. Save changes before exiting PowerCenter Designer.

Using Existing Tables to Create New Tables

You can use existing tables in Siebel Analytics Enterprise Data Warehouse to create new tables by making a copy of the table that most closely corresponds with your needs. This is the simplest way to make sure the table format is correct, and it saves time if some of the required columns are already present. The following procedure provides information on creating a target table. If the table is used as a source as well, you can copy the table into the Sources folder and modify its properties as necessary.

To create new target tables by copying existing tables

1 In PowerCenter Designer, open the Siebel Analytics Enterprise Applications folder, unless you are making a new staging table, in which case select the specific source configuration folder.

2 Select Tools > Warehouse Designer.

3 Open the Targets folder, copy the table most closely representing the table you want to create, and drag it into the Warehouse Designer.

- If you use the copy and paste functions in the Edit menu and you paste them back into the Siebel Analytics Enterprise Applications folder, you are prompted to rename the table because it already exists.
- By default all tables in Siebel Analytics Enterprise Data Warehouse are created with a database type of DB2. If you want to select a different database type, select the Database Type list window and select from the list.

4 Open the new table in Warehouse Designer to edit its properties.

5 Click the Columns tab and add, delete, or modify columns as necessary.

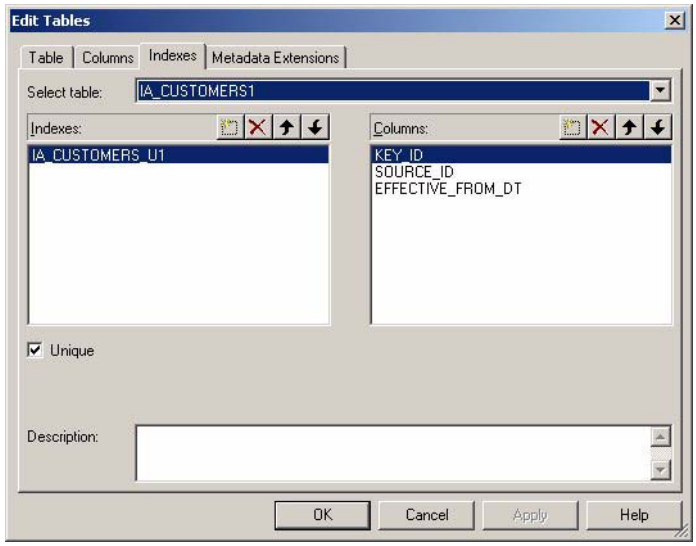
Be sure to conform column to format for data type, length, and precision.

6 Click the Indexes tab, and then click the New Insert button to enter the table name with the appropriate index suffix.

NOTE: The format for the index is [TABLE_NAME]_N[SEQUENTIAL_NUMBER] for nonunique indexes or [TABLE_NAME]_U[SEQUENTIAL_NUMBER] for unique indexes. For example, IA_BANK_LOCAL_N1 or IA_BANK_LOCAL_U1. Although indexes are not required, they help speed processing time by connecting the nonunique tables.

7 Click the New Insert button in the Columns window to open the Add Column to Index dialog box.

- Highlight the column from the list that you want the index to look for and click OK.
- You must repeat clicking the New Insert button, highlighting the appropriate column and then clicking OK for each column you want to add, as shown in the following figure.



8 If the index is unique, select the Unique check box.

Do not check the box for nonunique tables.

9 Click OK to return to the Warehouse Designer.

The new table is automatically added to the repository. Save changes before exiting PowerCenter Designer.

Creating a Profile Table Using Domain Values

This section describes how to create new profile tables. A *profile table* is a table that holds all possible combinations of specific multistate columns. You can create an unlimited number of profile tables, containing all possible combinations of domains and their values or the cross-product of the contents of IA_DOMAINS.

A profile table is built with the help of two prepackaged tables—the Domains table and the Profile Specifications table. The Domains table, IA_DOMAINS, stores predefined values for columns that can only hold certain values, as shown in Table 69. The Profile Specifications table, shown in Table 70, IA_PROFILE_SPECS, is used to join to the Domains table to generate all the possible combinations of values you may want to use in a profile table.

Table 69. Join IA_DOMAINS

Domain Name	Domain Position	Domain Value	Domain Flag	Language Code
REGN	0	Not Performed	Y	E
REGN	1	Started	N	E
REGN	2	Completed	N	E
RESEARCH	0	Not Performed	Y	E
RESEARCH	1	Started	N	E

Table 70. Join IA_PROFILE_SPECS

PROFILE NAME	Language Code	Domain Name - 1	Domain Name - 2	Domain Name - 3
WEB_SESS_CTXT	E	REGN	RESEARCH	E
NEW PROFILE	E	Domain-1	Domain-2	E

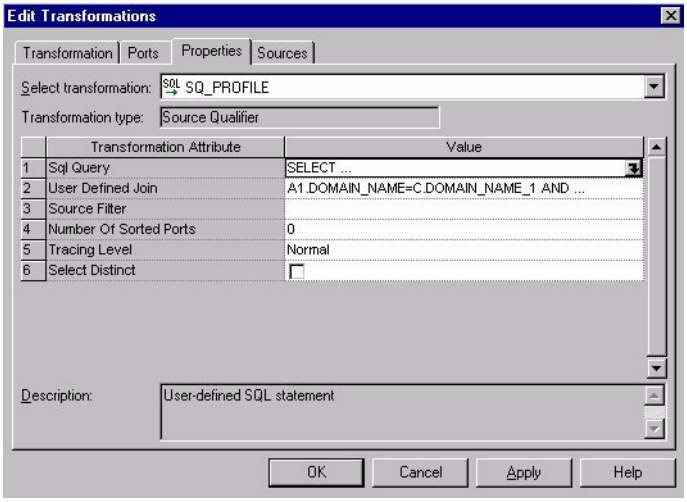
You can create profile tables using the already loaded data in IA_DOMAINS and IA_PROFILE_SPECS. For this purpose, Siebel Analytics Enterprise Data Warehouse includes a Business Component mapplet, which you can replicate and configure for every profile table you want to build. The Business Component joins up to ten instances of the IA_DOMAINS table to IA_PROFILE_SPECS using the DOMAIN_NAME column.

To create a new profile table, edit the SQL override in the Business Component Profile mapplet, adding your new profile table name in the PROFILE_NAME column. Then, edit the Expression transformation, mapping up to ten domains (that exist in IA_DOMAINS) in the Domain Name columns provided.

NOTE: You can build a profile table with a maximum of ten domains, if the first domain holds no more than three values and the other nine hold no more than ten each. If you build a profile table with fewer than ten domains, each domain can hold a maximum of ten values.

To build a new profile table using IA_DOMAINS and IA_PROFILE_SPECS

1. In PowerCenter Designer, open the Siebel Analytics Enterprise Applications folder.
2. Select Tools > Mapplet Designer.
3. Open Mapplets folder, and drag and drop MPLT_BCZ_PROFILE mapplet into Mapplet Designer.
4. Double-click the SQ_PROFILE Source Qualifier to open the Edit Transformations window.
5. Click the Properties tab, and then click the down arrow to access the SQL statement for the SQL query, as shown in the following figure.



6. Edit the SQL override by entering your new profile table name (for example, IA_SALES_PROFILE) in the PROFILE_NAME column and the Domain Names that comprise it in the DOMAIN_NAME_* columns.
7. Select Generate SQL, and then click OK.
8. Validate and save your changes to the repository.

NOTE: Fact tables in the Sales and Service modules contain foreign keys to profile tables. For all other tables that do not contain a profile key, use one of the dimension key extension columns packaged in the table.

Creating New Mappings

This section describes the following configuration procedures:

Creating an Extract Mapping

Extract mappings are used to extract source data and store it in the staging table. They usually contain the following components:

- Business Component mapplet
- Expression transformations
- Staging tables

NOTE: Universal Source extract mappings do not contain business components. For information on Universal Source mappings, see Integrating Data from Sources Without Prepackaged Business Adapters on page 336.

M_I_PURCH_ORDERS_EXTRACT, shown in Figure 44, is an example of a basic extract mapping.

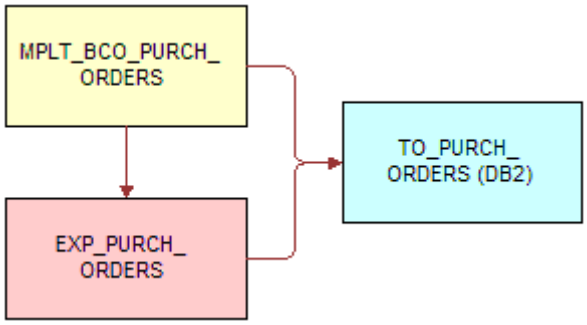


Figure 44. Basic Structure of an Extract Mapping

The Business Component mapplet extracts data from the source tables. For more information on creating a new Business Component, see the discussion on creating a new Business Component in Business Component Configuration on page 374. After the Business Component mapplet extracts the source data, it then passes the data to at least one Expression transformation, which configures the Source ID and the Key ID. The data can pass through as many different types of Expression transformations as necessary to transform the data into a usable state.

After the transformations occur, the data is passed to the staging area target table. When creating a new extract mapping, create a new staging table as well; do not use any existing staging tables as it may impact performance of other mappings using that same table. For information on creating a new staging table, see the discussion on the staging table format in Staging Table Format on page 348. For information on working with extension columns, see Overview of Integrating Additional Data on page 341.

TIP: If you are adding generic source data—that is data that is extracted from source systems other than the prepackaged sources, such as Oracle, PeopleSoft, and SAP—then you do not need a Business Component mapplet in the extract mapping. You can omit this step, because generic sources must be in a ready-to-load state and do not need business components. Transformations are not prepackaged in the Universal Adapter mappings that extract, transform, and load this type of data. For more information on adding generic source data, see Integrating Data from Sources Without Prepackaged Business Adapters on page 336.

NOTE: If a new staging table is created, you must create a new load mapping to move the data into the data warehouse. For information on creating a new load mapping, see Creating a Load Mapping on page 358. If an existing staging table's extension columns are used, the load mapping is prepackaged to move any data from the Staging table's extension columns into the Siebel Analytics Enterprise Data Warehouse's extension columns.

To create an extract mapping

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Select Tools > Mapping Designer.
3. Select Mappings > Create to create the mapping.
4. Enter the mapping name.

   For a list of naming conventions, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.
5. From the Repository Navigator, open the mapplet folder and drag the Business Component mapplet you require into Mapping Designer.
6. Open the transformation folder, and drag the reusable Expression transformation EXP_SOURCE_ID_FORMATION into Mapping Designer.

   NOTE: If no reusable Expression transformation is available, proceed to Step 10.
7. Double-click on the Expression transformation to open the Edit Transformations window, and select Rename.
   a. Enter a new name for the transformation and click OK.

      The naming convention for Expression transformations is EXP_[SUBJECT], where Subject is the subject that identifies the entity such as Suppliers or Customers.

   b. Enter a Description of the new Expression transformation for future reference, and click OK.
8. Drag the output ports from the Business Component mapplet to the Expression transformation input ports to connect them.
9. Create a KEY_ID column in the Expression transformation, and then:

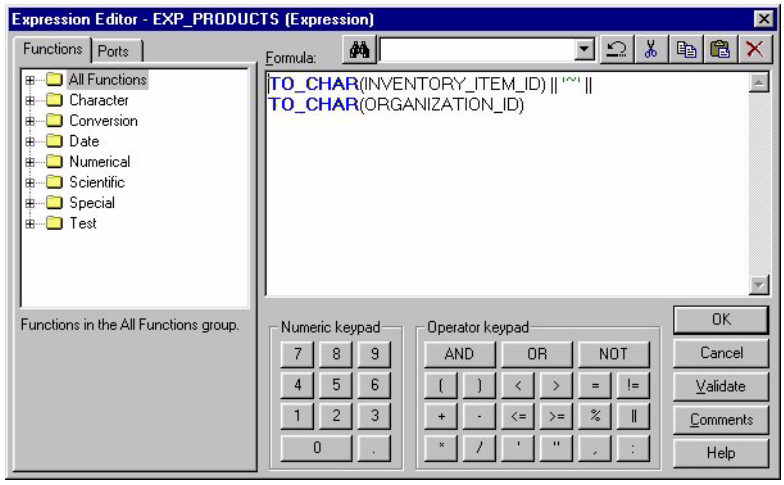
a On the Ports tab, edit the definition of the KEY_ID.

The Key ID uniquely identifies records within a source. The KEY_ID port should have 'string(80)' as the data type. This port should be an output only port—select output (O) flag.

b In the Expression column, enter the definition for the Key ID.

You must include all columns that make the records unique.

NOTE: While forming the Key ID, do explicit type conversions inside the Expression Editor box for those ports that are not of string data type. After you create your new KEY_ID port, you should be able to view it in the Expression transformation box, as shown in the following figure.



TIP: It is recommended you keep the columns having the greatest number of different values first and the columns with the least distinct values last, separating each column name with a tilde (~).

10 If a reusable transformation was available as defined in Step 9, continue directly to Step 12.

a If a reusable transformation is not available to provide a Source ID, you must create a new one by selecting Transformation > Create.

b Select Expression from the Select Transformation Type list window.

Enter a new name for the transformation and select Create, then Done.

c Drag the output ports from Copy the Business Component mapplet to the Expression transformation input ports to connect them.

11 Double-click the transformation to open the Edit Transformation window and create the Source ID.

NOTE: In addition to creating a Key ID, you must also create the Source ID.

a Edit the Expression transformation to add another column called SOURCE_ID, with data type of string, 30.

b On the Ports tab, edit the definition of the SOURCE_ID to an output port only.

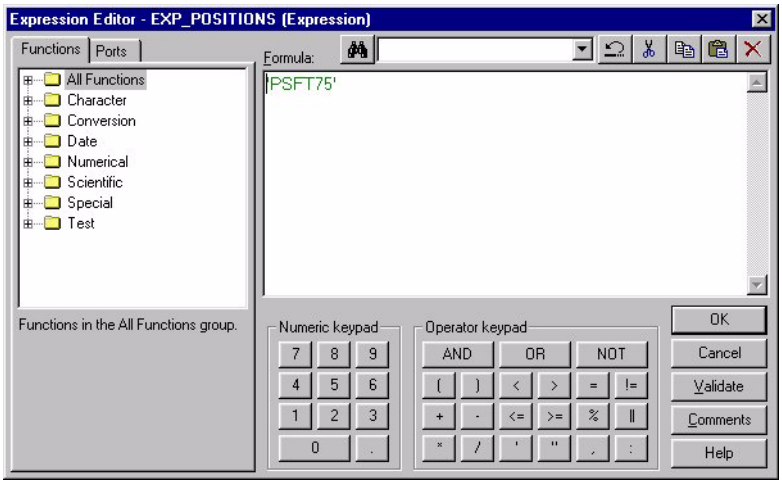
c Enter an abbreviation for the source.

The abbreviations for preconfigured sources are shown in the following table.

Source	Source Abbreviation for SOURCE_ID
Oracle 11i	OAP11i
PeopleSoft	PSFT75
SAP R/3	SAPR3

NOTE: You may change the default Source ID if you use multiple instances of the same source. For example, you may be running multiple instances of SAP and, therefore, want separate Source IDs for each instance. For example, you might call the first instance SAPR3_1, and the second instance SAPR3_2, and so on.

After you create your new SOURCE_ID port, you should be able to view it in the Expression transformation box, as shown in the following figure.



12 Open the Target folder, and drag and drop the staging table into Mapping Designer.

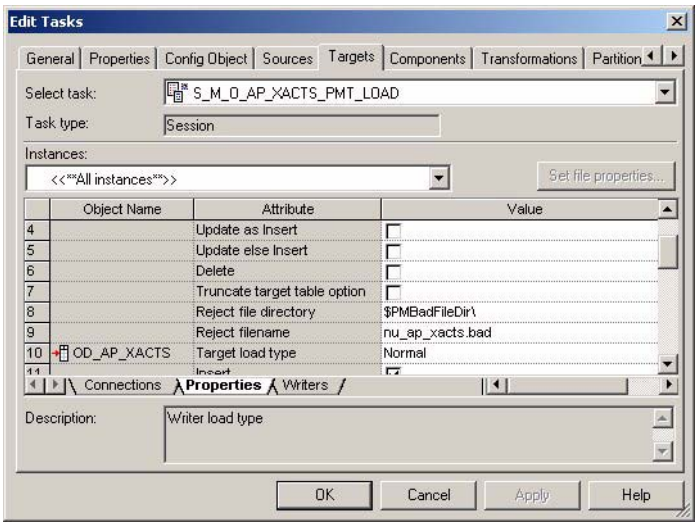
13 Connect the ports from the Expression transformation to the staging area target table.

The target table may have extra ports or extension columns created for later customization. These can be left empty if there is no corresponding port coming in from the Expression transformation.

14 Validate the mapping and save your changes to the repository.

15 In PowerCenter Workflow Manager, you must now create a session for this mapping.

Be sure to set the target load properties to truncate the staging table by selecting the Truncate check box as shown in the following figure.



Creating a Load Mapping

Load mappings transform data from a staging table and load it into the IA and OD target tables. They usually contain the following components:

- Staging table
- Shortcut to a load control area (OD_*) table or IA_* table
- Source Adapter mapplet
- Shortcut to ADI mapplet
- Shortcut to IA and OD target tables

The staging table provides temporary storage for extracted source data. The load mapping compares extracted source data in the staging table against the load control table for updates to existing data.

If there is no load control table, you should use a shortcut to the IA table. The IA table and the staging table pass the data to the Source Adapter. The Source Adapter contains most of the business logic used in preparing the data for the Analytic Data Interface (ADI). The Source Adapter uses an Expression transformation to handle data type conversions, source-specific lookups, and to create control columns used in loading.

The load mapping prepares data for the ADI, matching its output with the input of the ADI mapplet. The ADI mapplet contains source-independent transformation logic, such as:

- Lookups to resolve keys and codes for dimension table loads
- General calculations, such as currency conversions for fact table loads

The mapping, M_I_PURCH_ORDERS_LOAD, which is shown in Figure 45, provides a sample of a typical fact load mapping.

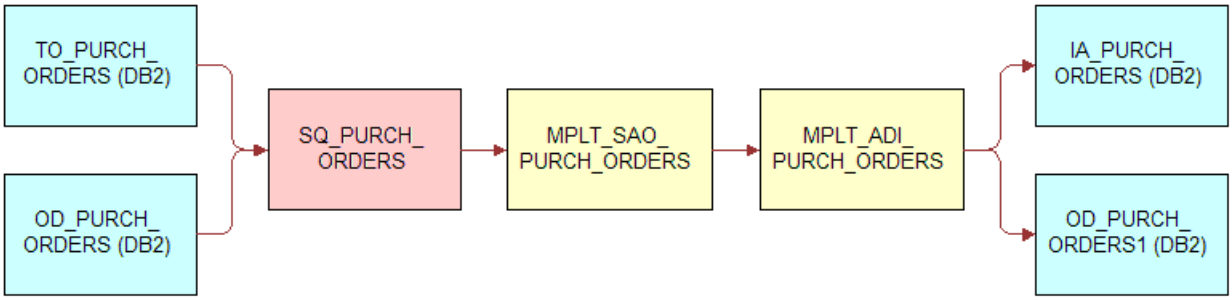


Figure 45. Basic Structure of a Fact Load Mapping

M_I_CUSTOMRS_LOAD shown in Figure 46 provides a sample of a typical dimension load mapping.

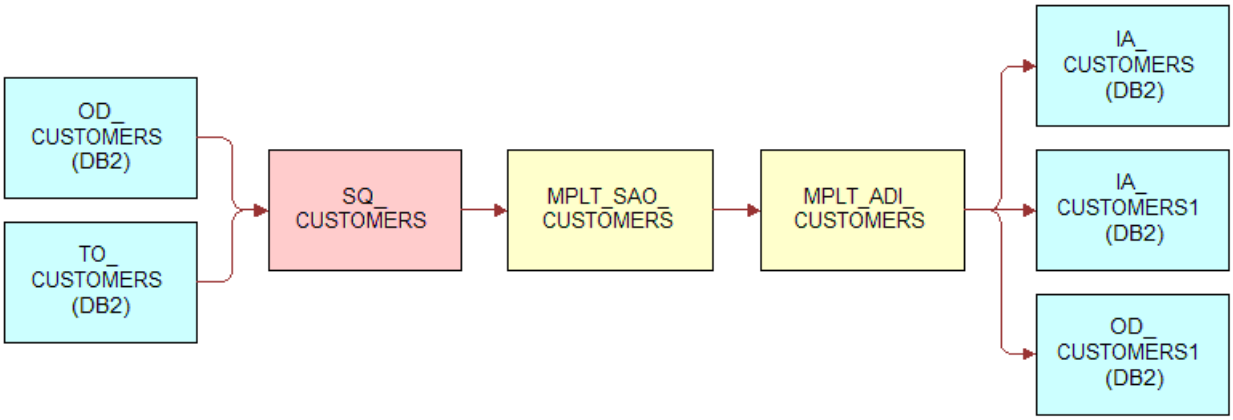


Figure 46. Basic Structure of a Dimension Load Mapping

To create a load mapping

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Select Tools > Mapping Designer.
3. Select Mappings > Create to create the mapping.
4. Enter the mapping name, and click OK.

   For a list of naming conventions, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.
5. From the Repository Navigator window, drag and drop the following items into Mapping Designer:
   - Staging table founds in IA_[SOURCE]_STAGE. For example, IA_ORA_STAGE for Oracle Applications.

- Shortcut to Control table (OD table or IA table) found in IA_[SOURCE]_CONTROL. For example, IA_ORA_CONTROL for Oracle Applications.
- Source Adapter mapplet from the Mapplet folder. For example, MPLT_SAO_AR_XACTS in Oracle Applications.
- Shortcut to ADI mapplet from the Mapplet folder. For example, MPLT_ADI_AR_XACTS.
- Shortcut to IA and OD target tables from the Target folder. For example, IA_AR_ACTIVITY.

NOTE: If you are going to require Type II slowly changing dimension support, you must import two instances of the IA target table into your dimension load mappings. For more information on Type II slowly changing dimension, see Type I and Type II Slowly Changing Dimensions on page 241.

6 Delete any unnecessary Source Qualifier transformations.

If set up to do so, PowerCenter Designer creates separate Source Qualifier transformations for each source table you drag into a mapping.

For example, if you drag in a staging table and a control table (or even multiple instances of the same table) as the sources for your mapping, you have one Source Qualifier for each of them. Because only one Source Qualifier is required, all other Source Qualifier transformations can be deleted.

7 After deleting any unnecessary Source Qualifier transformations, connect all ports to the one remaining Source Qualifier:

NOTE: In the previous example, the control table ports and the staging table ports are both connected to the same Source Qualifier transformation.

a Connect the Source Qualifier output ports to the Source Adapter mapplet input ports.

b Connect the Source Adapter mapplet output ports to the ADI mapplet input ports.

c Connect the first set of ADI mapplet output ports, classified under the MAPO_[SUBJECT]_IA1 heading, to one of the IA table instances.

NOTE: If you require only Type I changing dimension support, there is only one instance of the IA table and you can move directly to Step 9. For more information on Type I slowly changing dimension, see Type I and Type II Slowly Changing Dimensions on page 241.

8 If it is a dimension load mapping with Type II support, connect the second set of mapplet output ports, classified under the MAPO_[SUBJECT]_IA2 heading, to the second instance of the IA table.

This set is different from the main set of ports for the IA table; it only has the surrogate key and the control ports.

9 Connect the third set of mapplet output ports, classified under the MAPO_[SUBJECT]_OD heading, to the control table. If the dimension does not have a control table, then you only need to connect to the IA tables.

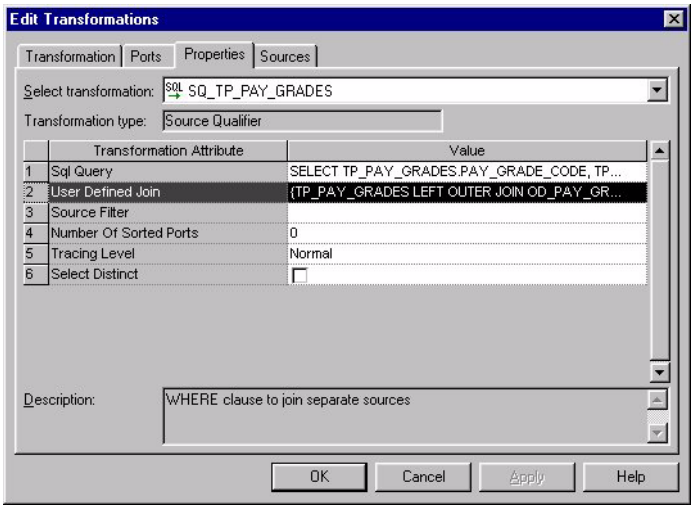
10 Double-click the Source Qualifier to open the Edit Transformations box.

You need to modify the User Defined Join and the SQL Query ports.

11 Click the Properties tab, and then click the small arrow in the Value column by User Defined Join to open the SQL Editor.

12 Edit the SQL statement for the User Defined Join port, as shown in the following figure, and enter the join condition between the source tables in this port.

Generally, the join is based only on the KEY_ID and SOURCE_ID. (Sometimes, SRC_EFF_FROM_DT, in addition to the KEY_ID and SOURCE_ID, is used in the join condition.) The join condition is an outer join between the staging area and the OD table. All staging area records should be selected, whether or not they are present in the OD table.



13 Click the small arrow in the Value column by SQL Query to open the SQL Editor.

14 Edit the SQL Query statement.

Generate the SQL statement by selecting the Generate SQL button.

15 Validate the query by clicking the Validate button.

TIP: Before validating the query, you must remove the WHERE clause from the SQL statement, and temporarily store the text until you validate the SQL. PowerCenter uses a database-independent outer join syntax, and the WHERE clause is one of the database independent outer join syntaxes not understood by SQL. After validation, replace the WHERE clause.

16 Save your changes to the repository.

17 In PowerCenter Workflow Manager, you must now create a session for this mapping.

For information on how to resolve dimension keys, see Resolving Dimension Keys on page 261.

Creating a Codes Mapping

To write new codes to the IA_CODES table, you need to design a codes mapping to extract data from the source and load it into the IA_CODES table. Codes mappings have the following five objects, as shown in Figure 47:

- Business Component mapplet (MPLT_BC[SOURCE_VARIABLE]_CODES_[SOURCE])
- Expression transformation (EXP_CODES_[SUBJECT])

- Source Adapter mapplet (MPLT_SA[SOURCE_VARIABLE]_CODES)
- ADI mapplet (MPLT_ADI_CODES)
- IA table (IA_CODES)

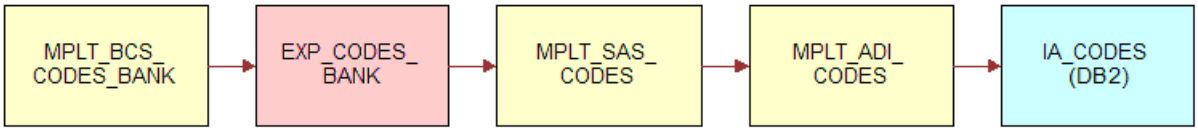


Figure 47. Codes Mapping

All codes mappings have the following three objects in common:

- Source Adapter mapplet
- ADI mapplet
- IA table

What makes each codes mapping unique are the remaining two objects—the Business Components mapplet and the Expression transformation. Therefore, when creating your new codes mapping, you can copy any codes mapping that comes from the same source and then change the Business Component mapplet and the Expression transformation.

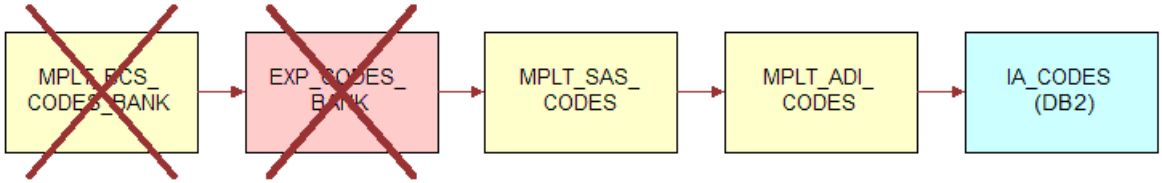
NOTE: Not all codes mappings require an Expression transformation.

To create a new codes mapping

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Select Tools > Mapping Designer.
3. Copy any existing codes mapping into Mapping Designer that has the same source as the one you want to create.

   By using an existing codes mapping you can avoid recreating the Source Adapter. Rename the codes mapping using the appropriate naming convention. For a list of naming conventions, see the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.
4. Delete the mapping's existing Business Component mapplet and, if applicable, Expression transformation.

   You need to create a new Business Component mapplet and Expression transformation for your new mapping.



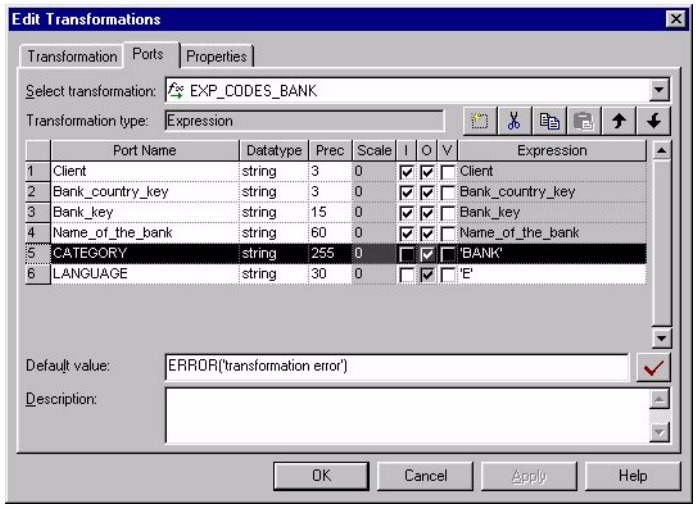
5 Create a new Business Component mapplet that includes a source definition, Source Qualifier, and mapplet output object (MAPO).

For more information on creating a new Business Component, see the discussion on creating a new Business Component in Business Component Configuration on page 374.

6 Drag the new Business Component mapplet into the mapping in Mapping Designer.

7 In the new codes mapping, create an Expression transformation for the Business Component mapplet.

In this expression, you need to set the new category, as shown in the following figure.



This category is the descriptive name of the type of code you are trying to create.

a Connect the detached ports in the new Codes mapping by first connecting the Business Component mapplet's output ports to the Expression input ports.

b Then, connect the Expression transformation output ports to the Source Adapter input ports from the mapping you copied in Step 3.

The remaining output ports of the Source Adapter, as well as the input and output ports of the ADI, should already be connected from this copied mapping.

8 Save your changes to the repository.

Creating a Derive Mapping

Derive mappings provide additional logic between the extract and load processes. They are primarily used to modify the format of data, or to merge data from two target tables and move it into a third. The naming convention for derive mappings is M_[SOURCE]_[SUBJECT]_DERIVE.

Derive mappings are only used in special cases. However, there may be a situation that requires additional features in a mapping. In this case, you need to build an additional mapping based on your needs.

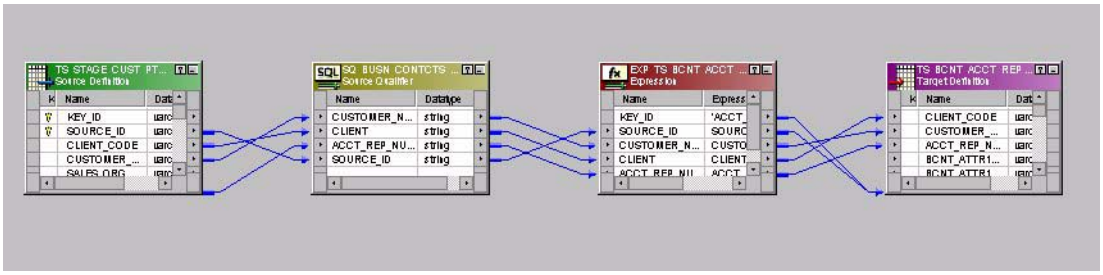
The procedure for building a derive mapping is the same as building a regular mapping. The structure of the mapping depends on the tables from which you want to derive data. The following are examples of derive mappings for SAP R/3:

- M_S_SALES_ORDLNS_LN_DERIVE

  Derives sales order line related partner and business data.

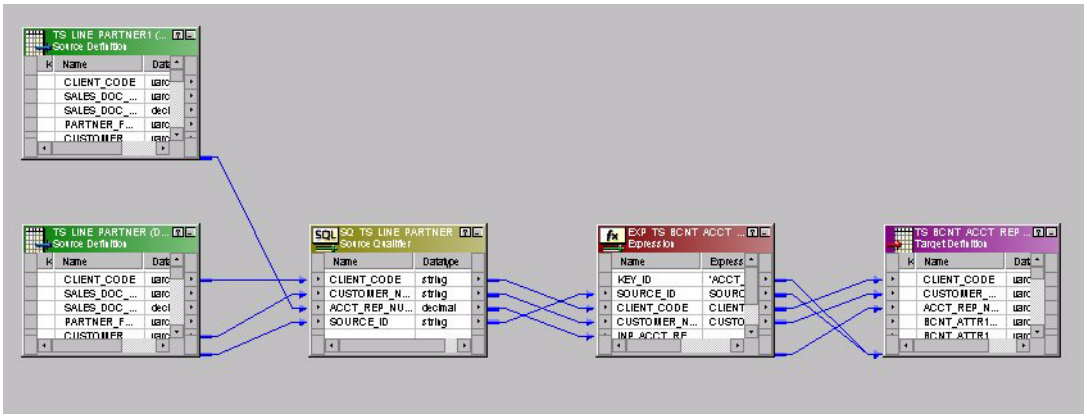
- M_S_BUSN_CONTCTS_ACCOUNT_REP_DERIVE_MASTER

  Extracts Sales representative information from SAP R/3, as shown in the following figure.



- M_S_BUSN_CONTCTS_ACCOUNT_REP_DERIVE_ORDLNS

  Extracts Account representative (order line level) information from SAP R/4, as shown in the following figure.



Creating and Populating Custom Key Figure Tables

Using key figure tables allows you to view a collection of facts from the point of view of a single dimension or dimension composite to answer key business questions. This section describes how a key figure table works and provides instructions for building your own tables to fit your business needs. This section covers the following topics:

- Understanding Key Figure Tables on page 365.

Understanding Key Figure Tables

Some decisions cannot be supported through a single fact alone; rather, a combination of facts must be viewed through the lens of a dimension, over a period of time. For example, your data warehouse contains a large number of facts about your customers. Taken individually, at the time they are captured, these facts can be useful in a limited way (for example, you could obtain a count of current customers). However, these single facts do not provide you with an overall picture of customer profitability.

Determining true profitability is more than looking at simple revenue. To consider all aspects of profitability, Siebel Analytics Enterprise Data Warehouse creates a key figure table that collects facts such as:

- Each customer's total sales generated
- Number of service calls placed
- Number of returns
- Incentives used such as campaigns
- Other facts required to determine a customer's overall profitability

Siebel Analytics Enterprise Data Warehouse then calculates this profitability metric, then stores it in the key figure table for use in analytics. Without a key figure table, a metric listing top ten profitable customers would not provide all the information necessary for well-rounded business decisions. Furthermore, facts stored in your data warehouse do not allow you to view trends in performance historically.

Without key figure tables, you may view the number of customers stored in your data warehouse, but that figure is only accurate until the next data load. While slowly changing dimensions does some work to preserve historical information, they only preserve records. In contrast, key figure tables allow you to see how a metric, such as number of customers, changes over a period of time chosen by you. For example, in addition to viewing your top ten most profitable customers, you can see how this key figure changes over the period of a month (or any other time period you select).

You can identify peak activity during your sales cycle, once-loyal customers who have not purchased recently, and other time-critical information. This information helps you forecast when your revenue grows, when to expect declines and also helps identify problems in monthly sales performance.

Reasons to Create a Custom Key Figure Table

In implementing key figure tables, you may find that your business requires you to view a collection of facts from the point of view of a dimension over a period of time.

For example, assume you have several product families in your inventory, such as a family of servers, each at a different price point. You also have information about this family of products stored in different tables in your data warehouse, such as descriptive details in the Product dimension, and a wide variety of facts about the individual products (number of sales, number of returns, number of service calls, cost of components, component suppliers, and so on).

A key figure table allows you to collect information about the key dimensions and facts of your products that is currently distributed over several tables into a single table. Therefore, you can produce more tailored, all-inclusive reports in an efficient time frame.

Preparation for Creating a Key Figure Table

After you have decided to create a key figure table you need to perform a variety of tasks. This section describes the tasks in preparation for creating individual key figure components—determining the grain, selecting fact tables, and determining required mappings and tables.

If you want to move directly to creating the key figure components, see Creating Key Figure PowerCenter Objects on page 370.

Determining the Grain

Selecting the grain of your key figure table requires you to balance your business requirements with your system performance capabilities. In choosing the grain of your key figure table, you must determine the time period you want the key figure to cover.

For example, you may decide to look at customer profitability by month, because viewing profitability more frequently does not allow you to factor in items such as returns and service calls. In contrast, you may decide to look at campaign effectivity on a daily basis, because you want to determine when marketing campaign response starts, peaks and stops.

Another factor in determining your key figure table's grain is approximating how many records your key figure table includes and the impact this has on your analytic system's performance. For example, if you try to calculate customer profitability each day, your analytic system would have to perform its normal daily data capture tasks as well as perform daily post-load profitability calculations on thousands of records. However, using existing aggregates to calculate profitability by month requires far fewer cycles of your analytic system while still providing meaningful business intelligence.

Other types of figures must be calculated daily to be useful. Events that are of brief duration, such as marketing campaigns, would not have enough detail if you only looked at the campaign's performance over a month. This kind of inquiry generally produces far fewer records, so the impact to your analytic system is minimal.

Selecting Fact Tables

In this task, you need to decide which facts must be included in your key figure table. Furthermore, you must select which facts are stored and which are calculated ad hoc, with only the calculation results stored in the table. For example, if you are building a key figure table for your product family, you store facts about total number of sales for the product family, number of sales per distinct product, number of different customers purchasing items in your product family, and so on. These facts can be derived from your data warehouse and stored.

It is simple to make an inventory of the facts and fact tables required for this part of your key figure table. However, there may be other facts that must be calculated from two different facts included in your key figure. Rather than store these individual facts and then perform calculations at the time the original data warehouse tables are loaded, performance is enhanced when the calculations are generated after the data warehouse tables load. This approach allows the calculation to be performed and inserted when the session is run to load the key figure table.

For example, assume you also need revenue in your product family key figure table, but revenue is derived from two facts; revenue from PSU_XACTS and number of sales from CUST_EVENTS. Rather than store these two disparate sets of figures, it is easier to use them to calculate revenue, and then store that data in the key figure table. To set up your key figure table in this way, you need to create an inventory of all calculations you want to store, their dependent facts, and the fact sources in your data warehouse.

After you have determined the facts and calculations to store in your key figure table, you are ready to determine the mappings required to collect the information required for your key figure table.

Determining Required Mappings and Tables

Populating a key figure table requires a number of mappings to populate intermediate tables to establish the grain of your key figure, collect the facts required, perform desired computations, and load the results. You must populate the following tables:

- Intermediate Parts Tables
- Primary Driving Table
- Key Figure Table

Intermediate Parts Tables

These store your collected facts and precalculated data in preparation for its final load.

You need a minimum of one mapping and one part table per fact table to extract, transform and load data in your key figure table. Intermediate Part tables stage aggregated fact data before the final key figure load. These are targets for derive mappings, and sources for the final key figure table.

Some fact tables may require more than one mapping; for example, if you want to store the number of purchases and the number of distinct products purchased, you need two mappings and two parts tables. Each of these two objects has the same source (for this example, the CUST_EVENTS table), but has different transformations and is loaded to a different target, as shown in Figure 48.

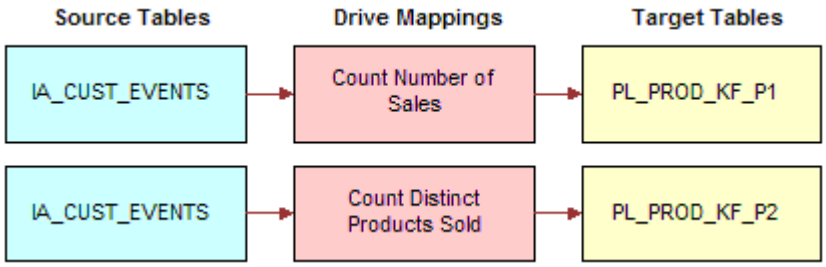


Figure 48. Populating an Intermediate Parts Table

Furthermore, you cannot join fact tables to perform calculations, because joined fact tables compromise the data in your warehouse (for example, creates duplicate counts of customer events). For this reason, you need to create mappings that aggregate the facts you are going to use in calculations and load them into intermediate parts tables. After you create and load the mappings, you then can use that aggregated data as your source for further computations, as shown in Figure 49.

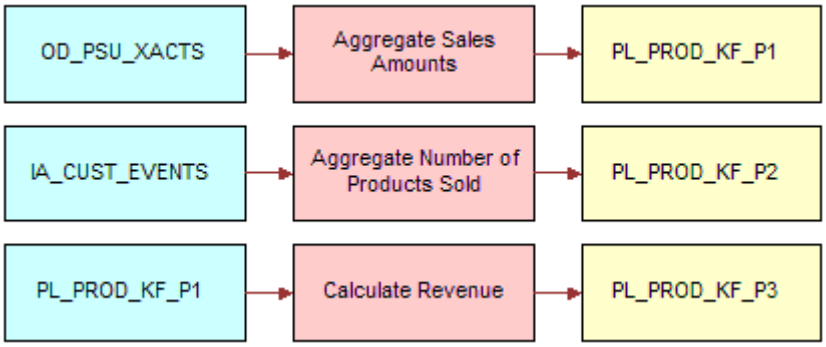


Figure 49. Aggregating Facts in an Intermediate Parts Table

Primary Driving Table

This table determines the grain of your key figure table and stores the key ID from your selected dimension. You only use one primary driving table per key figure table. The primary driver table contains key IDs, source IDs and effective dates for the dimension or dimension composite that defines all of the facts in your key figure. In this sense, the primary driver table drives the aggregation and calculations of these facts. It is the target for your truncate mappings that captures aggregated dimension data, and a source for the final key figure table.

To determine the grain and collect your key IDs, you extract the key ID from your dimension table (in this example, IA_PRODUCTS) and extract period start and period end dates from each fact table you are using in your key figure table, as shown in Figure 50.

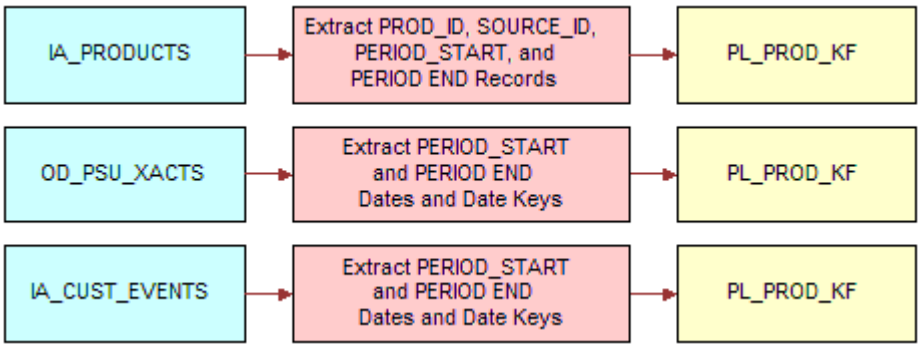


Figure 50. Populating the Primary Driving Table

Key Figure Table

You need one final mapping to populate your key figure table. The key figure table is the final target for your key figure load mapping. It contains the driving dimension's key IDs and source IDs, as well as the update strategy performed on the collection of facts you selected to support your key figure. In this final mapping, you join the parts tables, which contain your aggregated and precalculated facts and the key figure table itself, and then compare it to the last time you loaded your key figure table, set an update strategy for new and changed records, and load the key figure (Figure 51).

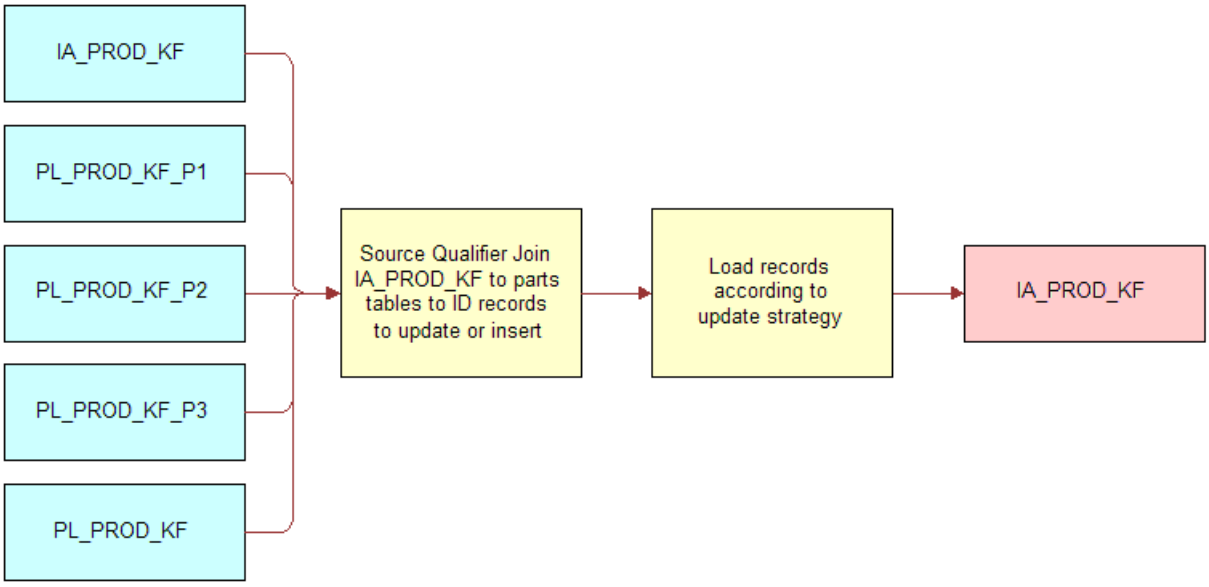


Figure 51. Populating the Key Figure Table

After the design phase is complete, you must develop the required database and PowerCenter objects required to support your key figure. See Creating Key Figure PowerCenter Objects on page 370 for the procedures required to build your key figure table.

Creating Key Figure PowerCenter Objects

When you understand the basic concepts behind each key figure component, you are ready to begin your own custom key figure table. This section presents procedures for creating the required database and PowerCenter objects.

Creating Required Database Objects

This section details instructions for creating the following required database tables:

- Intermediate parts tables
- Primary driver table
- Key figure table

The tables you are creating are created following the procedure in Creating New Tables on page 348, and Using Existing Tables to Create New Tables on page 350. You should be familiar with those sections of the chapter before proceeding. The steps in the following procedures assume you are familiar with creating tables in Siebel Analytics Enterprise Data Warehouse.

Creating Intermediate Parts Tables

Intermediate parts tables contain aggregated counts and amount facts. You must create at least one part table for each query required to build your key figure, so you may have more than one part table per source table. The intermediate parts table naming convention is PL_[TABLE_NAME]_KF_P#.

To create intermediate parts tables

1. Open PowerCenter Designer and select appropriate source configuration folder.
2. Select the fact tables from the Targets folder you want to use as sources for your parts tables and identify the columns you wish to extract.
3. Select Tools > Warehouse Designer.
4. Select Target > Create to create a new target table with the columns you want to extract from the fact tables selected in Step 2.

   Make sure to match data type and precision exactly.
5. Include the KEY_ID, SOURCE_ID and all period columns from your primary driving table.

   Mark these as Not Null and make them primary keys.
6. Select Target > Generate/Execute SQL.
7. Repeat these steps for every query you need to run to build your key figure.

Creating the Primary Driver Table

The primary driver table holds the KEY_IDs, SOURCE_IDs, start dates, and end dates for the facts and dimensions in your key figure. Its purpose is to define the dimension that describes your key figure and determine the update strategy for rows returned during data capture. The primary driver table naming convention is PL_[TABLE_NAME]_KF.

To create the primary driver table

1. In PowerCenter, open the Warehouse Designer. Create a new table, using the naming convention PL_[TABLE_NAME]_KF.
2. Create a column for the KEY_ID and SOURCE_ID of your dimension and make these primary keys.

   The data type should be varchar and the precision should match the source table.
3. Create a PERIOD_START_DT column, data type timestamp, and precision 26.
4. Create a PERIOD_START_DK column, data type decimal, and precision 10.
5. Create a PERIOD_END_DT column, data type timestamp, and precision 26.
6. Create a PERIOD_END_DK column, data type decimal, and precision 10.
7. Mark each column Not Null, and click OK.
8. Select Target > Generate/Execute SQL.

Creating the Key Figure Table

The key figure table contains the aggregated, calculated fact and dimension data to support key figure reporting. Its naming convention is IA_[TABLE_NAME]_KF.

To create the key figure table

1. Open the part tables you want to use as sources for your key figure table and identify the columns to extract.
2. In the Warehouse Designer, create a new target table with these columns.

   Make sure to data type and precision match exactly with the source.
3. Create additional columns to hold newly calculated data as appropriate.
   a. Include the KEY_ID, SOURCE_ID, and all period columns from your primary driving table.
   b. Mark CUSTOMER_ID, SOURCE_ID, PERIOD_START_DK and PERIOD_END_DK as Not Null and make them primary keys.
4. Select Target > Generate/Execute SQL.

Creating Required Mappings

In this phase you can create the extract, derive and load mappings required to populate your key figure table. The following procedures assume that you are also familiar with the process of creating basic mappings. For more information about mappings, see Creating New Mappings on page 353.

Extract Mappings

The extract mappings extract data from the fact and dimension tables you defined as required to support your final key figure. The data is loaded into incremental staging tables before being aggregated and loaded either in the primary driver table or an intermediate part table. The extract mapping naming convention is M_PLP_[TABLE_NAME]_EXTRACT.

Derive Mappings

The derive mappings extract required facts from the fact tables you identified as required for your key figure. For example, you may need to identify how many times a customer contacted customer service. In this situation, you would need to extract every row that matched the customer's ID from IA_CUST_EVENTS. The derive mapping naming convention is M_PLP_[KEY_FIGURE_TABLE_PRIMARY_FACT_TABLE]_DERIVE.

Key Figure Load Mapping

This mapping joins all of the part tables for your key figure with the key figure table itself (this identifies which records have changed and which have not). It then runs the final calculations necessary to answer your key business question identified during the design stage of this process. Its naming convention is M_PL_[KEY_FIGURE_TABLE_NAME]_LOAD.

The following procedures describe how to create extract mappings, derive mappings, and a key figure load mapping.

To create extract mappings to populate incremental tables

1. Begin a mapping with the IA fact or dimension table of your choice and the IA_DATES tables as sources.
2. Connect the ports you require for your key figure to the Source Qualifier.
3. Connect the CAL_DAY_DT port from the IA_DATES table to the Source Qualifier.
4. Create an expression to pass through required ports and determine which ports may return a question mark (?) if null.
5. Create an aggregate transformation to summarize source fact data.

   NOTE: Make sure the grain of each fact aggregate matches the grain of your overall key figure table.
6. Save and validate the mapping.
7. Repeat this process for every fact table you require.

To create part table derive mappings

1. Begin a mapping with your primary driving table, primary fact table, and any other desired tables as sources.

   Save this mapping as M_PL_[KEY_FIGURE_TABLE_NAME]_KF_[TABLE_NAME]_DERIVE.

2. Connect the key ID, source ID and period columns from your primary driver table, and the fact columns from all required fact tables you want to include in this parts table to your Source Qualifier.
3. Connect the appropriate post-load part table (PL_[KEY_FIGURE_TABLE_NAME]_KF_P#) to the Source Qualifier.
4. Save and validate the mapping.
5. Repeat this process for every primary fact table you wish to include in your final key figure table.

To create the key figure load mapping

1. Begin a mapping with all of your parts tables as sources. Save this mapping as M_PL_[KEY_FIGURE_TABLE_NAME]_LOAD.
2. Join your tables in the Source Qualifier.

   You may specify a user-defined join; otherwise simply connect your parts table columns to the Source Qualifier.

3. Create transformations to perform any additional calculations as needed to support your key figure.
4. Connect your Source Qualifier to any transformations you may have created.
5. Create a transformation to handle your update strategy.
6. Connect all required columns to the transformations you just created.
7. Save and validate your mapping.

Creating and Modifying Business Adapters

Business adapters consist of a Business Component mapplet and a Source Adapter mapplet. A single business adapter extracts a specific type of data (for example, sales order data) from a specific source (for example, SAP R/3), translates it into understandable business language where required, and prepares it for the source-independent Analytic Data Interface (ADI).

Because business adapters are source-specific, only certain business adapters for prepackaged sources are included in Siebel Analytics Enterprise Data Warehouse. If you are extracting data from a source for which no prepackaged business adapters are available, you can feed data into Siebel Analytics Enterprise Data Warehouse through a flat file, using Siebel Analytics Enterprise's universal business adapters. You can also build your own business adapters.

As you begin creating new mappings, you may also need to create, or modify, mapplets, transformations, and so on. The following sections provide instructions for modifying the Business Component mapplet and the Source Adapter mapplet. Within each procedure, you can also find how to modify the transformations contained in the mapplets.

Business Component Configuration

This section describes configuration procedures for creating a new Business Component and modifying an existing Business Component.

Creating a New Business Component

Business components reside in the configuration folder for each source and exist as mapplets. You can add new Business Component mapplets for a source.

To create a new Business Component

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Select Mapplets > Create.
3. At the Mapplets Name prompt, enter a name for your new Business Component mapplet.

   Siebel Analytics Enterprise Data Warehouse naming conventions are described in the *Siebel Analytics Enterprise Data Warehouse Data Model Reference*.
4. Expand the Sources folder.

   Drag and drop the source table into Mapplet Designer.
5. Add Source Qualifiers as appropriate.

   Depending on your PowerCenter settings, the Source Qualifier may be created automatically when you drag in the Source table.
6. Select Transformations > Create.
   a. At the Create Transformation prompt, select Mapplet Output from the drop down list window.
   b. Name the transformation using the MAPO_* naming convention.

      Select Create, and then click Done.
7. Drag and drop ports from the Source Qualifier to the Output transformation.

   Do not forget to link new data to the appropriate extension column. For information on the type of extension column to use, see Types of Extension Columns on page 324.
8. Edit the SQL statement in the Source Qualifiers.
9. Generate the SQL overrides and save your changes to the repository.

Modifying a Business Component

When modifying a Business Component mapplet, you need to identify what areas of the mapplet you wish to modify. The following procedure contains instructions for adding a new source definition, connecting the ports to the Source Qualifier, editing the Source Qualifier, connecting the ports to the Output transformation, and editing the Output transformation. First, define what areas of the Business Component mapplet you wish to modify, and then follow the procedure only for those specific areas.

To modify an existing Business Component mapplet

1. In PowerCenter Designer, open the applicable source configuration folder.
2. Select Tools > Mapplet Designer.
3. Open the applicable Business Component mapplet.
4. Expand the Sources folder, and copy a source table into your new mapplet by dragging and dropping the table into Mapplet Designer.
5. Drag and drop required columns from the new source definition to the Source Qualifier.
6. Double-click the Source Qualifier to open the Edit Transformations box, and then:
   a. Click the Ports tab, and make any changes to the new ports as necessary.
   b. Click the Properties tab to make changes to the SQL query as necessary.
   c. Click OK.
7. Drag and drop ports from the Source Qualifier to the Output transformation.
8. Save your changes to the repository.

Source Adapter Configuration

This section describes configuration procedures for creating a new source adapter and modifying existing source adapter expression transformations.

Creating a New Source Adapter

Source Adapter mapplets are source-specific objects. Therefore, when creating them, you must make sure you put them in their applicable folder.

To create a new Source Adapter

1. In PowerCenter Designer, open the configuration folder where the Source Adapter mapplet should be stored.

   For example, if the Source Adapter mapplet is for Oracle 11i, then open the Configuration for Oracle Applications v11i folder.
2. Select the generic mapplet of your choice from the navigator panel and select Edit > Copy.

3 Expand the configuration folder for the source for which you wish to create a new Source Adapter.

4 Highlight the mapplets node and select Edit > Paste.

At the prompt, Copy the mapplet...?, click Yes.

5 Rename the Source Adapter mapplet to a name different than the one being copied.

6 Open your new mapplet, select Mapplets > Edit, and rename it to reflect its new function using the Siebel Analytics Enterprise Data Warehouse naming conventions.

Click OK.

7 Save your changes to the repository.

Modifying Source Adapter Expression Transformations

When you edit a Source Adapter mapplet, you add the desired input port (named INP_*) to the input (MAPI) side of the mapplet, as shown in Figure 52. You then copy the new input port from the MAPI to the Expression transformation and then link the two ports. In the Expression, link the port to the closest existing extension (EXT_*) port. Step-by-step procedures follow. See Figure 52

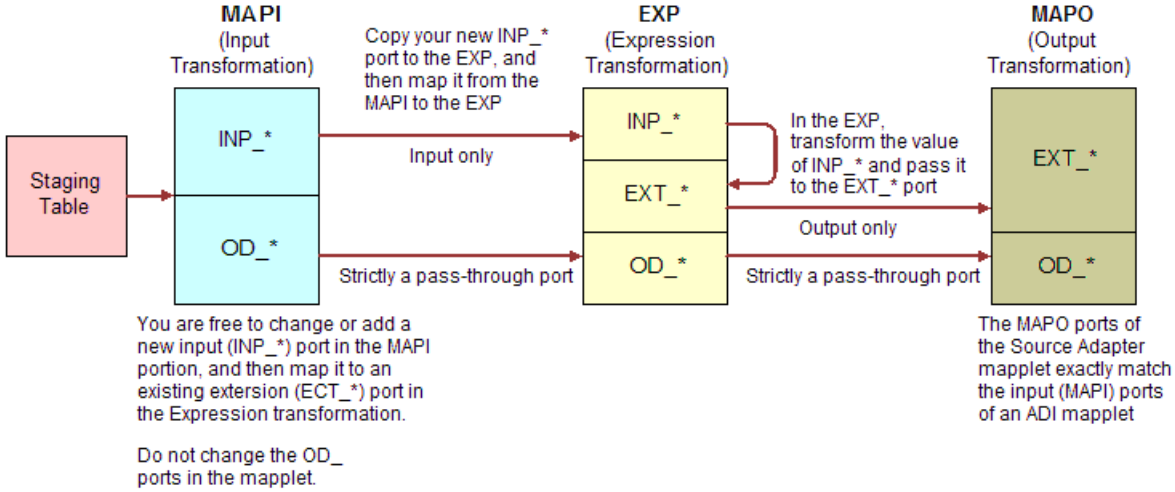


Figure 52. Input, Expression, and Output Ports of a Source Adapter Mapplet

To edit a Source Adapter Expression transformation

1 In PowerCenter Designer, open the applicable source configuration folder.

2 Select Tools > Mapplet Designer, and open the applicable Source Adapter mapplet.

3 Add a new input port to the MAPI side of the mapplet, following the INP_* naming convention.

4 Copy this new input port and add it to the Expression portion of the mapplet.

5 Link the new port from the MAPI to its new counterpart in the Expression.

6 In the Expression, clear Output indicator for the new port.

   You are mapping this port to an existing extension (EXT_*) port in Step 7.

7 From the available existing extension ports (EXT_*), select the closest appropriate port and double-click to open the Edit Transformations box.

8 On the Ports tab, find the chosen extension port, and select its Expression to open the Expression Editor.

   a In the Expression Editor, select your new input (INP_*) port.

      The extension column now contains the value of the new port you added.

   b Validate the new expression.

   c Edit any required ports in the Source Adapter.

      These ports vary depending on the dimension table.

9 If necessary, map the extension port you have configured to the corresponding EXT_* port in the Output transformation (MAPO).

   In Figure 52, the ports in the Output transformation of the mapplet match the input ports of the ADI mapplet exactly. This exact match makes sure that whatever comes out of the Output transformation directly feeds into the ADI.

10 Save your changes to the repository.

Index

No index available for this guide.