Siebel Enterprise Integration Manager Administration Guide

Version 7.7 Rev. A
September 2004

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

Contents

Chapter 6: Exporting Data

Chapter 7: Deleting Data

Chapter 8: Merging Data

Chapter 9: Running EIM

Chapter 10: EIM Performance Tuning

1 What's New in This Release

What's New in Siebel Enterprise Integration Manager Administration Guide, Version 7.7, Rev. A

Table 1 lists changes in this version of the documentation to support Release 7.7 of the software.

Table 1. What's New in Siebel Enterprise Integration Manager Administration Guide, Version 7.7, Rev. A

Topic	Description
Chapter 2, "Siebel Enterprise Integration Manager: An Overview"	Added content to the opening paragraphs in this chapter to clarify that the data exchange process that occurs between the Siebel database and external databases happens in two parts, and that it is the second part that involves EIM.
"About the Second Row Property on EIM Table Mapping Objects"	Shortened and revised this topic to simplify the explanation of the Second Row property, which is set when a base table is mapped for the second time in an EIM table.
"Recommended Import Order for Importing Legacy Data"	Added note to this topic.
"Importing an Initial Batch of Legacy Data"	Updated the navigation in the task for importing initial batches of data.
"Parameters Used for Imports in Both the Header and Process Sections"	Updated the corresponding EIM table for S_CONTACT.PR_OU_ADDR_ID from EIM_CONTACT2 to EIM_CONTACT in Table 10 within "MISC SQL Parameter" on page 71.
"Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts"	Added this topic.
"Importing Party Records"	Added note explaining that the PAR_PARTY_ID field needs to be populated only when the PARTY_TYPE_CD is set to Organization or Position.
"Importing Positions and Employees"	Revised this topic to update view navigation.
"Troubleshooting Import Processing Failures"	Revised title of this topic from "Evaluating Import Processing Failures" and updated the format of troubleshooting information provided.
"Parameters Used for Exports in Both the Header and Process Sections"	Added note in "EXPORT MATCHES Parameter" on page 99.

Table 1. What's New in Siebel Enterprise Integration Manager Administration Guide, Version 7.7, Rev. A

Topic	Description
"Parameters Used for Merges in Both the Header and Process Sections"	In "SET BASED LOGGING Parameter" on page 121, corrected a statement: the SET BASED LOGGING parameter must be set to FALSE to allow transaction logging for merge.
"Running an EIM Process Using the Graphical User Interface (GUI)"	Updated navigation in the task for how to run an EIM process using the GUI.
"Viewing the EIM Log File"	Renamed this topic (was "Viewing the Task Info Log") and revised information on trace flags and error flags, including task navigation and other changes within the following topics: ■ "Using Trace Flags, SQL Trace Flags, and Error Flags" on page 129 ■ "Setting Event Logging from the Graphical User Interface (GUI)" on page 131 ■ "Setting Event Logging from the Command-Line Interface" on page 131 ■ "Trace Flag Settings" on page 132.
"USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: EIM Criteria for Passing Indexes to the Database"	Added this topic.
"Oracle Databases"	Revised this topic to update the performance tuning information provided regarding extensions to Oracle databases in "Avoiding Excessive Table Fragmentation" on page 164.
Appendix A, EIM Error Messages	Removed this appendix.
"Example of Importing from Multiple EIM Tables in a Single .IFB File"	Added this topic.
"Example of Updating Columns When There Are Two Non-Target Base Tables Mapped to One EIM Table"	Added this topic.
"Example of Troubleshooting the Import of Extension Columns"	Added this topic.
"Example of Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts"	Added this topic.
"Example of Importing and Exporting Hierarchical LOVs"	Added this topic.

Table 1. What's New in Siebel Enterprise Integration Manager Administration Guide, Version 7.7, Rev. A

Topic	Description
"Examples of Resolving Foreign Keys"	Added this topic, which includes the following examples of resolving foreign keys: ■ "Example 1: Error Message "This is a foreign key value in the base table and the values in the interface table did not resolve to existing values."" on page 201 ■ "Example 2: Resolving the Foreign Key for Position Division" on page 203 ■ "Example 3: Resolving the Foreign Key Using a Special User Key" on page 203
"Example of Setting Explicit Primary Mappings"	Replaced the usage example in this topic with one that is relevant to both Siebel eBusiness Applications and Siebel Industry Applications (SIA).
"Foreign Key Column Values: NO MATCH ROW ID versus NULL versus a Valid ROW_ID"	Added note to this topic.

What's New in Siebel Enterprise Integration Manager Administration Guide, Version 7.7

Table 2 lists a change described in this version of the documentation to support Release 7.7 of the software.

Table 2. New Product Feature in Siebel Enterprise Integration Manager Administration Guide, Version 7.7

Topic	Description
EIM Table Mapping Report Generation See "Generating EIM Table Mapping Reports" on page 29	Previous versions of the *Siebel Interface Tables Reference* included a listing of the EIM tables in the repository. With this release, the *Siebel Interface Tables Reference* is no longer published because you can now generate this information yourself, from your own repository. You can produce your own report that contains every EIM table in your repository, or query to select EIM tables, including EIM tables you have added.

2 Siebel Enterprise Integration Manager: An Overview

Siebel Enterprise Integration Manager (EIM) is a server component in the Siebel eAI component group that transfers data between the Siebel database and other corporate data sources. This exchange of information is accomplished through intermediary tables called EIM tables. (In earlier releases, EIM tables were known as interface tables.) The EIM tables act as a staging area between the Siebel application database and other data sources.

EIM is your primary method of loading mass quantities of data into the Siebel database. You should use EIM to perform bulk imports, updates, merges, and deletes of data. Examples of each of the main EIM functions (import, export, update, and delete) are provided in "EIM Functions."

In the Siebel application database, there are application tables (known as base tables), which Siebel applications use. For data to come from other corporate data sources (external databases) into Siebel application tables, the data must go through EIM tables. So the data exchanges between the Siebel database and external databases occurs in two parts:

1. Load data into EIM tables.
2. Run Siebel EIM to import the data from the EIM tables into the Siebel base tables.

   NOTE: While the first part of this data-exchange process involves the intermediary tables that are called EIM tables, only the second part of the process involves the functionality of Siebel EIM.

When data is entered through the Siebel user interface, the application references properties set at the business component object type. However, when data is entered into Siebel base tables through EIM, EIM references properties set at the table object type.

NOTE: You must use EIM to perform bulk imports, exports, merges, and deletes, because Siebel Systems does *not* support using native SQL to load data directly into Siebel base tables (the tables targeted to receive the data). You should also be aware that EIM translates empty strings into NULL.

EIM Functions

This guide explains how to configure and use Siebel EIM to perform the functions described below. Each function is discussed separately in the chapters referenced.

Import New and Revised Data into Siebel Base Tables

The EIM import function can be used in several different ways:

- When initially implementing a Siebel application, load the Siebel database tables with data and file attachments created by external applications. For example, you can import information about product lines and products from an inventory control database into the Products entity in the Siebel database.

- As part of maintaining the Siebel database, you can use EIM for data archival. This not only provides customers with a Siebel database that is optimally using the resources available to it, but also streamlines the implementation of a corporate data archival strategy.
- As part of maintaining a non-Siebel database, you can update it with information from the Siebel database. For example, you might add new customers to an accounting database from the Siebel database.

Refer to Chapter 5, "Importing Data," for a detailed discussion of the import function.

Export Data from Siebel Base Tables

The data contained within a Siebel application is available for transfer to non-Siebel applications by using EIM. When implementing a non-Siebel application, you can export data from the Siebel database tables for use by that application. For example, you can export employee information to a corporate sales commission application. Refer to Chapter 6, "Exporting Data," for a detailed discussion of the export function.

Delete Data from Siebel Base Tables

As part of maintaining the Siebel database, you can identify rows to be deleted from a table and its associated child and intersection tables. For example, you might delete an obsolete product line and its associated products. Refer to Chapter 7, "Deleting Data," for a detailed discussion of the delete function.

Merge Data in Siebel Base Tables

In response to such external events as corporate mergers, you can merge two or more database rows into a single row. For example, you might merge the Frame, Inc. account information into the Adobe Corp. account. Refer to Chapter 8, "Merging Data," for a detailed discussion of the merge function.

Process Flow Between EIM and Other Databases

For each EIM process, you need to complete the following sequence of steps.

1. **Prepare the EIM tables.** For delete, merge, or import operations, the EIM tables require loading with representative data that allows EIM to identify the specific Siebel base table on which to operate. You can use either an SQL ancillary program utility or native SQL to perform this function. The structure of the EIM tables has the required mappings for the primary (or target) base table and other base tables that are serviced by the EIM table. The EIM export processes require minimal preparation of the EIM tables. When an export operation takes place, the EIM tables are populated with data from the Siebel base tables. Therefore, you can use either an SQL ancillary program or native SQL to transfer data from the Siebel application to a non-Siebel application. For more information, see Chapter 3, "Siebel EIM Tables."
2. **Edit the EIM configuration file.** An ASCII or Unicode (binary) text file of extension type .IFB that resides in the Siebel Server/admin directory allows you to define the type of EIM processes to be performed: export, delete, merge, or import. For more information, see Chapter 4, "EIM Configuration File."
3. **Run EIM.** EIM is submitted as a Siebel Server batch component task either from the Administration - Server Management views or from the Server Manager command line interface. For more information, see Chapter 9, "Running EIM."
4. **Check results.** The EIM component task produces a log file, which provides tracing information about the process. The tracing information produced is variable dependent upon the EIM component task parameters used and the Siebel Server event logging deployed for the EIM component. As always, during testing operations you should check the EIM processes using increased tracing information, and then reduce tracing when the process is deployed to production.

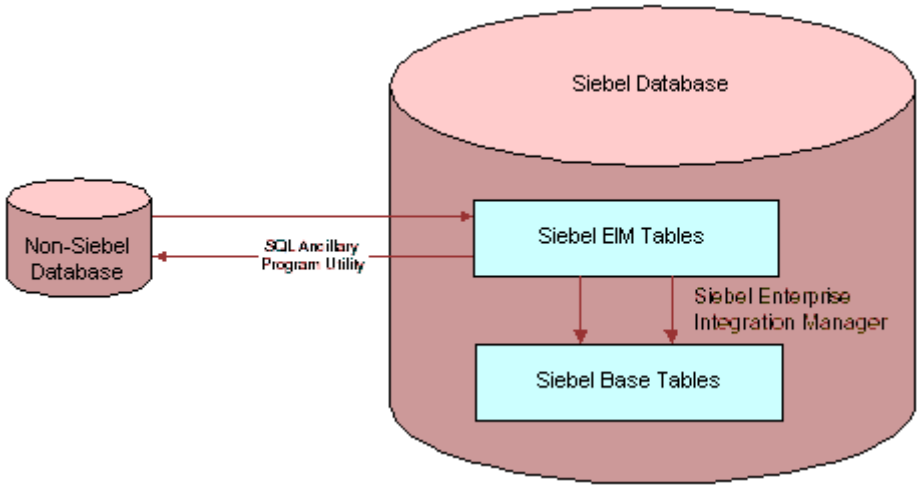


Figure 1. Process Flows Between Siebel Database and Other Databases

Figure 1 illustrates the following processes:

- How a non-Siebel database uses an SQL ancillary program utility to receive or send data to Siebel EIM tables.
- How Siebel EIM is used to move data between Siebel EIM tables and Siebel base tables.

Mobile Web Client Requirements

Due to the complexity of table relationships and Mobile Web Client requirements, you must use EIM to import data into Siebel base tables.

CAUTION: Do not attempt to modify data directly in the physical tables. Siebel Systems does *not* support performing this activity for the reasons that follow. The logical relationships that exist within the Siebel base tables are many and complex, as governed by the Siebel repository metadata. Direct modification of Siebel base tables is not supported because there is a high risk of data integrity corruption. EIM maintains data integrity and resolves foreign key relationships during the import process. In addition, EIM data inserts, updates, or deletes get routed to mobile users with Siebel Remote local databases or Siebel replicated nodes.

The only exception is when you are migrating the entire Siebel schema from one database to another. In this case, you may select to use a tool provided by the database vendor to migrate the data.

In other rare cases where EIM cannot be used, it may be possible to use Siebel Visual Basic (VB) to insert, update, or delete large amounts of data. For information on VB methods, see *Siebel VB Language Reference*.

For initial data loading, you should consider set-based operations for all EIM processes. To maximize performance, you should also consider running EIM processes in parallel.

For ongoing operations, if you are using Mobile Web Clients within your architecture, you should consider EIM in row-by-row operations for the data that is required of the Mobile Web Clients. Running large EIM processes and set-based operations usually requires performing a database extraction for Mobile Web Clients, if the data being manipulated affects them.

3 Siebel EIM Tables

This chapter discusses Siebel EIM tables (also known as interface tables) and how EIM uses them. Siebel EIM tables are intermediate database tables that act as a staging area between the base tables in the Siebel database and other databases. This chapter is organized into the following sections:

EIM Tables Overview

Siebel EIM tables are intermediate database tables that act as a staging area between the base tables in the Siebel database and other databases. EIM tables are designed to be simple and straightforward so they can be loaded or read by way of external programs.This section provides an overview of how EIM works with these EIM tables and how table names are derived.

Preparing EIM Tables for Merge, Update, or Import Processes

Before EIM can be used in a merge, update, or import process, a Siebel administrator or a database administrator must populate the EIM tables with data, using any method supported by the database. A Siebel administrator then invokes EIM to process this data. EIM makes multiple passes through the tables to complete the specified process.

Each EIM table usually supports a group of base tables that can be imported or exported in a single batch. Base tables are the tables within the Siebel database that contain your data. Base tables are the final destination of data imported into the Siebel database and the source of data exported from the Siebel database.

NOTE: If the Siebel administrator is importing into base tables that use the UTC (Universal Time Coordinated) time scale, the Siebel administrator or a database administrator must convert the local time in the data into UTC before loading data into the EIM tables.

EIM Table Naming Conventions

EIM tables in the Siebel database use a three-part naming convention; the syntax is: PREFIX_NAME_SUFFIX. These three parts are described as follows:

- **PREFIX.** All interface tables used by EIM have the prefix EIM_ (such as EIM_ACCOUNT). These EIM tables support Organizations, so they can be used for all EIM processes.

  NOTE: Previous versions of EIM used a different set of EIM (interface) tables, identified by the prefix S_ and the suffix _IF. These tables still appear in the Siebel database, but are inactive. These tables will *not* be included in the Siebel database in future versions. If you need these tables activated temporarily, contact Siebel Expert Services.

- **NAME.** A unique table name that is generally composed of an abbreviation of an entity type name. If more than one EIM table is required to fully support an entity, a sequential number may be added to the name of each table after the first one.

- **SUFFIX.** A supertype name may be followed by a suffix that indicates the types of data supported by the EIM table or to distinguish it as an EIM table.

For more information, see "Viewing EIM Table Mappings to Base Tables" on page 26.

EIM Table Columns

Running EIM is an iterative process, with each step accomplishing specific tasks and moving toward successful completion of the entire process. To process on a row-by-row basis, EIM uses several columns common to every EIM table. These columns are described in this section.

Several columns are mandatory. Others are conditionally mandatory, depending on the conditions of your import. To determine mandatory columns, use Siebel Tools to view each column in an EIM table and the EIM table's target base table columns.

By following the recommended import sequence, you make sure that the appropriate data dependencies are established.

NOTE: For import and merge processes, you must populate the ROW_ID, IF_ROW_STAT, and IF_ROW_BATCH_NUM columns in the EIM tables. This also must be done for delete processes when you run DELETE EXACT. For merge processes, you also need to populate the IF_ROW_MERGE_ID column. Do not populate these required columns with spaces because a space does *not* equal a NULL value.

Mandatory Columns for EIM Processing

ROW_ID. For an EIM table row to be eligible for processing, you must initialize its ROW_ID. The ROW_ID, in combination with the value of IF_ROW_BATCH_NUM, must yield a unique value. The ROW_ID values in the EIM tables are *not* the ROW_ID values that are assigned to the row when it is loaded into the base table. An EIM-generated ROW_ID has a ##-###-### format. A regular row ID that is assigned to the row has a #-## format.

IF_ROW_BATCH_NUM. You must set the values in this column to the same integer, greater than or equal to 0, as an identifying number for all rows to be processed as a batch. The maximum value is 2147483647. Use this column as the first key of any new indexes created on an EIM table.

IF_ROW_MERGE_ID. You can set this column to one of two values:

- NULL. This value identifies the surviving or merged-into-row.

- ROW_ID. This value identifies the ROW_ID number in the EIM table where the row will be merged.

  NOTE: This value is the ROW_ID of records in the EIM table, not the base tables.

IF_ROW_STAT. EIM updates this column after processing the row to indicate the status of the record. The IF_ROW_STAT column is not used by EIM when determining which rows to process. When populating the EIM tables, you can set this column to any value except NULL. You can initially set this value to FOR_IMPORT to indicate that the row has not been imported. After processing, if certain rows were not imported due to a data error, you should change:

- IF_ROW_BATCH_NUM value for the rows that require reimporting
- BATCH line in the configuration file

If EIM updates this column to NOT_ALLOWED after processing a row, EIM has attempted to insert a new row but the action is not allowed. In such cases, the INSERT ROWS parameter may have been set to FALSE.

IF_ROW_STAT_NUM. After processing, this column contains a zero (0) if a row was successfully processed to completion. If processing failed, this column contains the pass number where the pass failed.

Temporary columns. EIM uses temporary columns to manipulate data during processing. For example, EIM might store the ROW_ID value for a Siebel base table in a temporary column. These column names begin with T_ and indicate the table or column for which they are used. Because EIM uses these columns internally during processing, do not manipulate these columns in the EIM tables.

For detailed information about each EIM table, generate a table mapping report. See "Generating EIM Table Mapping Reports" on page 29.

File Attachment Columns

Three EIM table columns must be populated in order to import file attachments. Table 3 describes these columns and uses the attachment file budget99.doc as an example.

Table 3. File Attachment Columns

Column	Description	Example
FILE_NAME	This column requires the root filename of the file attachment.	FILE_NAME="budget99"
FILE_EXT	This column requires the extension type of the file attachment (DOC, XLS, or TXT).	FILE_EXT="doc"
FILE_SRC_TYPE	This column requires the value "FILE" or the rows cannot be imported.	FILE_SRC_TYPE="FILE"

You can also use these columns to define hyperlinks, as shown in Table 4.

Table 4. Defining Hyperlinks With File Attachment Columns

Column	Setting
FILE_NAME	Set to actual URL
FILE_EXT	NULL
FILE_SRC_TYPE	'URL'

Organization Columns

The EIM_ type interface tables use the xxx_BU/xxx_BI column pairs to map organizations. For example, the CON_BU/CON_BI column in the EIM_CONTACT interface table is mapped to the BU_ID column in the S_CONTACT base table.

In order for organizations to be resolved properly, you need to populate the xxx_BU column with the organization name and leave the xxx_BI column empty. Do not populate the xxx_BU column with the organization ROW_ID. EIM looks up the ROW_ID for the organization in xxx_BU and puts it in the corresponding xxx_BI column.

EIM Table and Column Mappings

EIM uses EIM table mappings to map columns from EIM tables to Siebel base tables. Siebel predefined EIM mappings are fixed and cannot be remapped. Using Siebel Tools, you can:

- View EIM table mappings to Siebel base tables
- View interface column mappings to Siebel base table columns
- View Siebel base table mappings to EIM tables
- Print EIM table reports

Some base tables may not be mapped to a corresponding EIM table. In such cases, use Siebel VB to load data into these base tables and inform Siebel Technical Services regarding the missing mapping. EIM does not interfere with Siebel VB code because Siebel VB works at the business object layer, and EIM works at the data object layer. You can also use the EIM Table Mapping Wizard to add missing mappings. For more information, see *Configuring Siebel eBusiness Applications*.

For information on using Siebel VB, see *Siebel VB Language Reference*.

Database Extensibility and EIM

If you have licensed Database Extensibility and created extensions, you can use the Column Mapping view to specify mappings to your new fields. Database Extensibility and EIM support mappings between columns in extension tables and EIM tables only if these columns share the same base table. To map EIM table extensions to base table extensions, you must specify which column the extended field will point to in the base table. For more information on Database Extensibility, see *Configuring Siebel eBusiness Applications*.

EIM Table Mappings Provided as Common Parents to Nontarget EIM Table Mappings

Some EIM table mappings (usually to the target base table) are provided only as a common parent to nontarget EIM table mappings. An example of this type of EIM table mapping is mapping from the EIM_OPTY_DTL interface table to the S_OPTY base table. These EIM table mappings have a comment in the Siebel repository, indicating that they do not support inserting or updating data.

In such EIM table mappings, only the user key columns are mapped. Except for updating the primary foreign key columns, EIM does not support inserting and updating rows using these EIM table mappings.

Parameters to Set

For stability of EIM when using these EIM tables, follow the template in the default.ifb file by including the following parameters for the relevant section in the EIM configuration file:

- INSERT ROWS = *optional parent_table*, FALSE
- UPDATE ROWS = *optional parent_table*, FALSE

CAUTION: If you do not include these parameters, the EIM process may fail or some exceptions may occur.

Exception to Recommended Parameter Settings

One exception to the recommendation provided above is when you want to update the primary foreign key columns in the parent table, in which case you do not want to include the following parameter in the EIM configuration file:

UPDATE ROWS = *parent_table*, FALSE

For example, EIM_ACCOUNT1 maps to the user key columns of S_ORG_EXT only. You can use EIM_ACCOUNT1 to update the primary foreign keys in S_ORG_EXT if the explicit primary mappings exist, such as S_ORG_EXT.PR_INDUST_ID, in the explicit primary mapping contained in the table mapping of S_ORG_INDUST. For more information, see "About Explicit Primary Mappings" on page 24.

In this case, you should use the default setting, UPDATE ROWS = S_ORG_EXT, TRUE in the EIM configuration file. If you do not need to update primary foreign keys in S_ORG_EXT, then you should set UPDATE ROWS = S_ORG_EXT, FALSE in the EIM configuration file.

Creating New EIM Table Mappings to Existing Base Tables

You can create new EIM table mappings from an EIM table into a base table if either of the following conditions is true:

- Mappings already exist from the EIM table to the base table.
- The base table is an extension table and mappings already exist from the EIM table to the corresponding base table.

For example, you could create a new column in EIM_ACCNT_DTL and map this either to a new extension column in S_ORG_EXT or to an existing column in the extension table S_ORG_EXT_X. These mappings are defined using Siebel Tools.

If you create an extension column to a base table, then run the EIM Table Mapping Wizard, the Wizard creates the following mappings:

- The mapping for the newly added extension column
- The mappings for all unmapped columns in the base table, including unmapped Siebel base columns

In general, manually creating mappings to an existing Siebel base column in Siebel Tools is not supported. Please contact Expert Services for further assistance.

About Explicit Primary Mappings

The Siebel Data Model uses primary foreign keys (or primaries) to point from a parent base table to a child base table. Primaries enable business logic in the Siebel Data Model, such as identifying the primary position for an account. Moreover, primaries improve performance by eliminating repeating subqueries when data from both the parent table and the primary child table are displayed. If you do not use primaries, then you must execute a new query to identify any child records each time a parent record is displayed.

For more information, see the following sections:

- "Setting Explicit Primary Mappings" on page 24
- "Setting Explicit Primaries for Many-to-Many Relationships" on page 25

Setting Explicit Primary Mappings

Primary foreign keys are columns that have names usually beginning with PR_ and are defined as primaries in the data model. If both the parent table and the primary child table of a primary foreign key are mapped to the same EIM table, then you should see an explicit primary mapping for this primary foreign key under the table mapping of the primary child table.

NOTE: Before you can create an explicit primary mapping, both the parent and the primary child table must be mapped to the same EIM table.

If an explicit primary mapping exists, you can use EIM to set the primary explicitly during import or update by setting the primary flag column in the EIM table. For an example of this, see "Example of Setting Explicit Primary Mappings" on page 204.

Setting Explicit Primaries for Many-to-Many Relationships

The example of setting a primary key in "Example of Setting Explicit Primary Mappings" on page 204 explains how to set an explicit primary for a one-to-many relationship. When setting a primary key for a many-to-many relationship, such as the relationship between Opportunities and Contacts, there is also an intersection table to consider.

For an example, see "Example of Setting Explicit Primary Mappings for Many-to-Many Relationships" on page 204.

About Viewing EIM Table Mappings

Before viewing EIM table mappings, you need to make sure your View settings are correct in Siebel Tools so that you can see the appropriate object types in the Object Explorer. For more information, see "Verifying Your Object Explorer View Settings" on page 25.

Information on viewing EIM table mappings is organized as follows:

- "Viewing EIM Table Mappings to Base Tables" on page 26
- "Viewing Interface Column Mappings to Base Tables" on page 27
- "Viewing Base Table Mappings to EIM Tables" on page 28
- "Generating EIM Table Mapping Reports" on page 29

Verifying Your Object Explorer View Settings

In order to be able to view all the EIM object types in the Siebel Tools Object Explorer, verify that your settings are correct.

To verify your view settings for the Object Explorer

1. Start Siebel Tools.
2. From the View menu, choose Options.
3. In the Development Tools Options dialog box, click the Object Explorer tab.
4. From the Object Explorer Hierarchy, find the EIM Interface Table object.

   The EIM Interface Table object may appear checked, but gray.

5 Uncheck the EIM Interface Table object, then check it again.

The EIM Interface Table object appears checked, and no longer gray.

6 In the Object Explorer Types tab, expand the EIM Interface Table object.

The rest of the EIM object types appear beneath the EIM Interface Table object type.

Viewing EIM Table Mappings to Base Tables

Use Siebel Tools to view EIM table mappings to base tables.

To view EIM table mappings to base tables

1 Start Siebel Tools.

2 In the Object Explorer, click the Types tab.

3 Click EIM Interface Table.

4 In the EIM Tables window, select the EIM table for which you want to view the mappings.

5 In the Object Explorer, expand EIM Interface Table.

6 Click EIM Table Mapping.

The EIM Table Mappings window displays all base table mappings for the selected EIM table.

You can view mappings for all interface columns, but you can only add or modify mappings for extended columns in the base schema to extended columns in the EIM tables.

Figure 2 shows an example of viewing the EIM table mappings for the EIM_ACCOUNT interface table. In the EIM Table Mappings list applet, you can find information about each base table that has been mapped to the selected EIM table. The Destination Table field contains the physical name of the mapped base table. You can also see which temporary columns (T_*) EIM is using when processing a mapped base table.

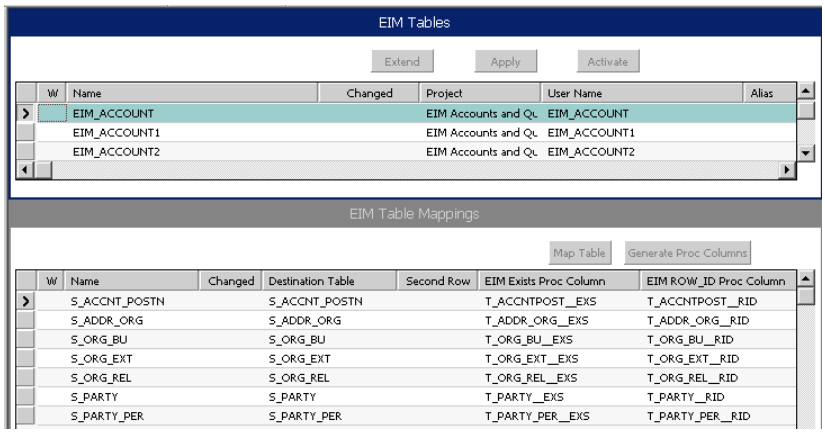


Figure 2. Viewing EIM Table Mappings to Base Tables

Viewing Interface Column Mappings to Base Tables

Use Siebel Tools to view interface column mappings to base table columns.

To view interface column mappings to base tables

1. Complete "To view EIM table mappings to base tables" on page 26.
2. In the EIM Table Mappings window, select a base table.
3. In the Object Explorer, expand EIM Table Mapping.
4. Click Attribute Mapping.

   The Attribute Mappings window displays column mappings for the selected base table.

Figure 3 shows an example of viewing column mappings for the S_ADDR_ORG base table. (This example is specific to Siebel eBusiness Applications rather than Siebel Industry Applications.) In the Attribute Mappings list applet, for a selected base table mapping, you can find information about the mapping that has been defined between the EIM table column and the base table column. For example, Figure 3 shows that the S_ADDR_ORG.ADDR_NAME column has been mapped to the ADDR_ADDR_NAME (EIM_ACCOUNT) interface column.

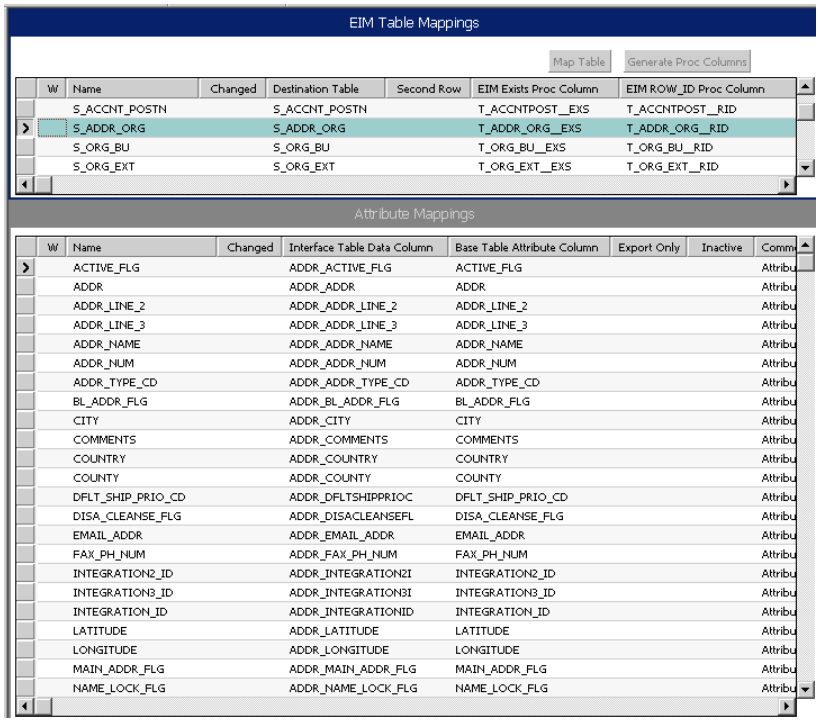


Figure 3. Viewing Interface Column Mappings to Base Tables

Viewing Base Table Mappings to EIM Tables

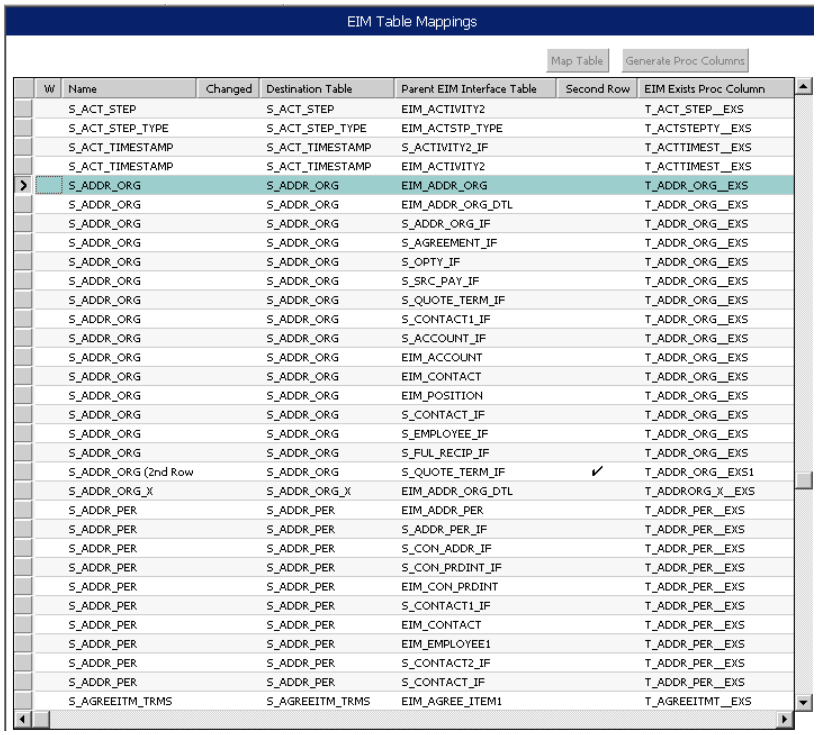
Use Siebel Tools to view base table mappings to EIM tables.

To search for an EIM table mapping to a specific base table

1. Start Siebel Tools.
2. In the Object Explorer, click the Types tab.
3. Expand EIM Interface Table, and click EIM Table Mapping.
4. Execute a query for a base table mapping, entering the name of the base table in the Destination Table field.

   The query returns all EIM tables that include a mapping to the base table. The EIM table to which the base table is mapped is shown in the Parent EIM Interface Table field. Some base tables may be mapped to more than one EIM table.

Figure 4 shows an example of viewing the EIM table mappings for the S_ADDR_ORG base table. (This example is specific to Siebel eBusiness Applications rather than Siebel Industry Applications.) Note that the S_ADDR_ORG base table maps to many EIM tables.



W	Name	Changed	Destination Table	Parent EIM Interface Table	Second Row	EIM Exists Proc Column
	S_ACT_STEP		S_ACT_STEP	EIM_ACTIVITY2		T_ACT_STEP__EXS
	S_ACT_STEP_TYPE		S_ACT_STEP_TYPE	EIM_ACTSTP_TYPE		T_ACTSTEPTY__EXS
	S_ACT_TIMESTAMP		S_ACT_TIMESTAMP	S_ACTIVITY2_IF		T_ACTTIMEST__EXS
	S_ACT_TIMESTAMP		S_ACT_TIMESTAMP	EIM_ACTIVITY2		T_ACTTIMEST__EXS
	S_ADDR_ORG		S_ADDR_ORG	EIM_ADDR_ORG		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	EIM_ADDR_ORG_DTL		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_ADDR_ORG_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_AGREEMENT_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_OPTY_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_SRC_PAY_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_QUOTE_TERM_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_CONTACT1_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_ACCOUNT_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	EIM_ACCOUNT		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	EIM_CONTACT		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	EIM_POSITION		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_CONTACT_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_EMPLOYEE_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG		S_ADDR_ORG	S_FUL_RECIP_IF		T_ADDR_ORG__EXS
	S_ADDR_ORG (2nd Row		S_ADDR_ORG	S_QUOTE_TERM_IF	✓	T_ADDR_ORG__EXS1
	S_ADDR_ORG_X		S_ADDR_ORG_X	EIM_ADDR_ORG_DTL		T_ADDRORG_X__EXS
	S_ADDR_PER		S_ADDR_PER	EIM_ADDR_PER		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_ADDR_PER_IF		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_CON_ADDR_IF		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_CON_PRDINT_IF		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	EIM_CON_PRDINT		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_CONTACT1_IF		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	EIM_CONTACT		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	EIM_EMPLOYEE1		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_CONTACT2_IF		T_ADDR_PER__EXS
	S_ADDR_PER		S_ADDR_PER	S_CONTACT_IF		T_ADDR_PER__EXS
	S_AGREEITM_TRMS		S_AGREEITM_TRMS	EIM_AGREE_ITEM1		T_AGREEITMT__EXS

Figure 4. Viewing Base Table Mappings to EIM Tables

Generating EIM Table Mapping Reports

Previous versions of the *Siebel Interface Tables Reference* included a listing of the EIM tables in the repository. You can produce your own report that contains every EIM table in your repository, or query to select EIM tables, including EIM tables you have added.

CAUTION: In the following procedure, unless you query for specific EIM tables, the default action is to print every EIM table in the repository. Because outputting a report for every EIM table takes a significant amount of time to prepare and print, it is recommended that you perform a query prior to submitting your report request.

To print an EIM table mapping report

1. Start Siebel Tools.
2. In the Object Explorer, click the Types tab.
3. Click EIM Interface Table.

   TIP: If the EIM Interface Table object is not visible in the Object Explorer, you must select it in the Development Tools Options dialog box. To verify your view settings and make this change, see "Verifying Your Object Explorer View Settings" on page 25.
4. (Optional) To print select EIM tables, create and run a query.
5. From the Reports menu, select EIM Interface Tables.

   The Siebel Report Viewer window opens with the Printer icon unavailable (gray). When the report is complete, the Printer icon becomes available.

   NOTE: The lower-left corner of the Siebel Report Viewer window indicates the status, which is either "Generating report ..." or "Report complete - xxxx page(s)".
6. Click the Printer icon.

   The following information appears in your EIM table report:

   - **Destination Table.** The base table.
   - **Destination Column.** The column name.
   - **UK.** The user key sequence, if any.
   - **FK Table.** The name of the table referenced by this column, if it is a foreign-key column.
   - **Source Column.** The name of the interface column that maps to the destination table.
   - **Comments.** This section provides a short description of the EIM table. It also includes information on any special restrictions and usages, such as in cases where the destination table has no user keys, and the special behavior of EIM for those tables.

About the Second Row Property on EIM Table Mapping Objects

The Second Row property is set when a base table is mapped for the second time in an EIM table. If a base table always has data row pairs, it is useful to map the base table twice in an EIM table such that one row in the EIM table will become two different rows in the base table. An example of this is the table mappings of S_INV_LGR_ENTRY in EIM_INV_TXN.

EIM Table Mappings to Base Tables Without User Keys

Some EIM tables contain table mappings to base tables without user keys. When using these EIM tables, you should note the EIM behavior for the relevant process type as described in "Process Issues for Base Tables Without User Keys" on page 31.

EIM Tables and Base Tables Without User Keys

Table 5 lists some examples of EIM tables containing table mappings to base tables without user keys.

Table 5. Example EIM Tables With Table Mappings to Base Tables Without User Keys

EIM Table	Target Base Table Without User Key
EIM_ACCNT_DTL	S_NOTE_ACCNT
EIM_ACCSRCPIDTL	S_NOTE_ACCSRCPI
EIM_ACC_SRC_DTL	S_NOTE_ACC_SRC
EIM_ACT_DTL	S_NOTE_ACT
EIM_ASGN_GRP	S_ASGN_RESULT
EIM_ASSET_DTL	S_NOTE_ASSET
EIM_BASELN_DTL	S_NOTE_BASELINE
EIM_CON_DTL	S_NOTE_CON
EIM_CONSUM_DTL	S_NOTE_CONSUME
EIM_CON_PI_DTL	S_NOTE_CON_PI
EIM_DCP_DTL	S_NOTE_DCP
EIM_DEFECT_DTL	S_NOTE_DEFECT
EIM_INVC_DTL	S_NOTE_INVOICE
EIM_NOTE	S_NOTE
EIM_OPTY_DTL	S_NOTE_OPTY
EIM_ORDER1	S_NOTE_ORDER

Table 5. Example EIM Tables With Table Mappings to Base Tables Without User Keys

EIM Table	Target Base Table Without User Key
EIM_ORDER_ITEM1	S_NOTE_ORDER_IT
EIM_GROUP_DTL	S_NOTE_ORGGROUP
EIM_PRDINT_DTL	S_NOTE_PROD_INT
EIM_PROJECTDTL	S_NOTE_PROJ
EIM_PROJITMDTL	S_NOTE_PROJITEM
EIM_PROJRSRCDTL	S_NOTE_PROJRSRC
EIM_QUOTE_DTL	S_NOTE_QUOTE
EIM_QUO_IT_DTL	S_NOTE_QUOTE_IT
EIM_PDSHIP_DTL	S_NOTE_SHIPMENT
EIM_SR_DTL	S_NOTE_SR
EIM_SRC_DTL	S_NOTE_SRC
EIM_TARGET_DTL	S_NOTE_TARGET
EIM_USR_MSG_DTL	S_NOTE_USR_MSG
EIM_WFM_ACTION	S_ACTION_ARG
EIM_WFM_RULE	S_ESCL_ACTION

Process Issues for Base Tables Without User Keys

This subsection describes issues that you should be aware of when performing EIM processes involving base tables without user keys.

Importing Data into Base Tables Without User Keys. Import works but EIM does not check and prevent duplicate records from being imported into the base tables without user keys. If an import batch is executed repeatedly, the same records are imported repeatedly because EIM cannot check whether the records to be imported already exist in the base table without user keys.

Updating Data in Base Tables Without User Keys. Update on base tables without user keys cannot work, because EIM cannot uniquely identify the record to update.

Exporting Data from Base Tables Without User Keys. Exporting data from base tables without user keys is done the same way as exporting data from base tables with user keys.

Deleting Data from Base Tables Without User Keys. DELETE ALL ROWS and DELETE MATCHES can be used to delete data in target base tables. If a table without a user key is the target table, then delete works as it does for base tables with user keys. In most cases, however, a table without a user key is a secondary table and its data can only be deleted with the table as a child of its parent table.

NOTE: EIM_NOTE_DEL and EIM_SKLI_DEL are special EIM tables used for deleting from the S_NOTE* and S_*SKILL_IT tables, which do not have the normal U1 user key.

Merging Data in Base Tables Without User Keys. Merge does not work on base tables without user keys.

Deleting EIM Table Rows

When you have successfully imported most of your EIM table rows, you can delete them. However, you might want to leave rows that were not fully imported in order to examine and correct them. If you want to do this, remember that each EIM table imports data into one or more target base tables. For example, EIM_ACCOUNT imports into S_PARTY, S_ORG_EXT, S_ORG_BU, S_PARTY_PER, S_ORG_REL, S_ACCNT_POSTN, S_ADDR_ORG, and S_CTLG_CAT_ORG.

- Each EIM table includes a separate temporary column that contains a status code for each base table into which it has imported data. The names of these columns are contractions of the target base table name.

  For example, T_ORG_EXT__STA. T_ indicates that this is a temporary column; ORG_EXT is the first three letters of each word in the target base table name (S_ORG_EXT), and __STA indicates that this is the status column. Note that the extension begins with two underscores.

- During import, a row's status column is set to 0 for those tables into which the row was successfully imported. The IF_ROW_STAT is set to IMPORTED if a row is successfully imported into all target base tables, or PARTIALLY IMPORTED if it is successfully imported into at least one target.

  - To delete rows that were successfully imported into all target base tables, you could use the following SQL statement:

    ```
    delete from EIM_ACCOUNT
    where (IF_ROW_STAT = 'IMPORTED')
    ```

  - To delete rows that were successfully imported into specific target base tables, you could use the following SQL statement:

    ```
    delete from EIM_ACCOUNT
    where (IF_ROW_STAT = 'PARTIALLY_IMPORTED' and
    T_ORG_EXT__STA = 0 and T_ADDORG__STA = 0)
    ```

  - You can also use ONLY BASE TABLES to limit processing.

Finding Differences in EIM Tables Between Repositories

The Siebel Data Model changes from release to release, and EIM mappings change accordingly. You can use the UTLEIMDIFF utility to find EIM mapping differences between two repositories for a list of EIM tables that you input. The results can be used to help you update your EIM data loading scripts, programs, and so on.

To use the UTLEIMDIFF utility

1 Create the view S_EIM_MAP_V in the database.

The database-platform-independent script for creating this view is called create_EIM_MAP_V.sql. This script can be found in the <dbsrvr>\common directory.

2 Find the executable UTLEIMDIFF.EXE in the <tools>\bin directory. Use the following switches for the program:

Switch	Entry	Description
/U	[username]	Siebel username
/P	[password]	Siebel password
/C	[connect string]	ODBC connect string
/D	[table owner]	Database table owner
/N	"[new Siebel repository]"	Required. Name of the new repository. **NOTE:** Enclose the repository name in quotation marks.
/O	"[old Siebel repository]"	Required. Name of the old repository. **NOTE:** Enclose the repository name in quotation marks.
/I	[input filename]	This file contains the list of EIM tables to be compared. The default input file (eim_tbl_lst.inp) is in the <tools>\bin directory. You can edit this file.
/M	[report filename]	Required. This is the output report. The default name is eim_diff.txt.
/L	[log filename]	The default name is eim_diff.log.

The program may run for several minutes, depending on the number of tables to be compared.

3 Interpret the three parts of the output file as follows:

- **Part 1 - Interface Table Difference.** Part 1 compares all the EIM tables in the two repositories.
- **Part 2 - Interface Table Mapping Difference.** Part 2 compares the EIM tables listed in the input file.
- **Part 3 - Interface Column Mapping Difference.** Part 3 compares the interface columns for the tables listed in the input file. "UK" means "User Key sequence." "Req'd" indicates that the column in the base table is required.

The first column of each part is the repository name. If there is an entry in one repository but not the other, then that means that the entry exists in one repository but not the other. If the same entry appears in both repositories, then that means that the entry has been modified.

4 EIM Configuration File

This chapter covers the generic use of EIM configuration files (referred to as .IFB files) and is organized into the following sections:

For specific parameter-level information that affects importing, deleting, merging, and exporting, refer to the chapters for those functions.

Using the EIM Configuration File to Define a Process

EIM reads a configuration file that specifies the EIM process to perform (import, update, merge, delete, or export) using the appropriate parameters. The EIM configuration file (the default file is default.ifb) is an ASCII text file of extension type .IFB that resides in the Siebel Server/admin directory. Before you can run an EIM process, you must edit the contents of the EIM configuration file to define the processes for EIM to perform.

NOTE: If you are planning to use Unicode in your implementation, then the EIM configuration file must be saved as a Unicode text file.

EIM then sets the process locale as specified during start up in the command line, the Server Manager graphical user interface (GUI), or the configuration file. You must specify the correct character set, such as Western European or UTF-8, for the target database in one of these locales. For information on locales and character sets, see *Global Deployment Guide*.

EIM accepts parameter values from three sources:

- The command line entered by the user that invokes the EIM process
- The Siebel Server Manager GUI
- The configuration file specified, or default.ifb if none is specified

Parameter value searches are performed according to a specific hierarchy: command line, component parameter, and configuration file. Command-line parameters thus override component parameters, and component parameters override configuration file parameters.

NOTE: If the batch number component parameter is set to 0, the batch number in the EIM configuration file (if any) is used. This is the only exception to the parameter hierarchy.

You can define multiple processes in the EIM configuration file and then invoke a specific process using the process parameters discussed later in this chapter. Alternatively, you can create multiple configuration files and specify which one EIM should use.

Defining EIM Configuration File Parameters

The EIM configuration file begins with a header section used to specify global parameters that apply to all process sections defined later in the file. Following the header section, there must be at least one process section with its associated parameters. Some process section parameters are generic for all EIM processes. Other process section parameters are specific to a particular EIM process, such as import.

This chapter describes only the header section and process section parameters that are generic to all EIM processes. For information on process-specific section parameters, see the relevant chapter for each process:

- For an import process, see "Editing the Configuration File for Import Processing" on page 64.
- For an export process, see "Editing the Configuration File for Export Processing" on page 97.
- For a delete process, see "Editing the Configuration File for Delete Processing" on page 108.
- For a merge process, see "Editing the Configuration File for Merge Processing" on page 120.

EIM Configuration File Parameters

You can find descriptions of all EIM configuration file parameters in this chapter and the chapters that follow. For information on inheritance rules, see "Inheritance Rules for Configuration Parameters" on page 42.

Each parameter is categorized by the specific type of EIM process in which it is used:

- **General Header Parameters.** Header parameters may be used in all EIM processes. See Table 6 on page 37 for a list of general header parameters.
- **General Process Parameters.** General process parameters may be used in all EIM processes. See Table 7 on page 39 for this list.
- **Import Process Parameters.** Import process parameters apply specifically to an import process. See Table 9 on page 65 and Table 10 on page 67.
- **Export Process Parameters.** Export process parameters apply specifically to an export process. See Table 15 on page 99.
- **Delete Process Parameters.** Delete process parameters apply specifically to a delete process. See Table 16 on page 110.
- **Merge Process Parameters.** Merge process parameters apply specifically to a merge process. See Table 18 on page 121.

You may want to refer to the default.ifb configuration file as you read the description of each parameter.

Header Section Parameters Generic to All EIM Processes

Header parameters are necessary at the beginning of the .IFB file. At a minimum, `[Siebel Interface Manager]` and `PROCESS` must be specified. Table 6 provides descriptions of header parameters.

Table 6. General Header Parameters for the EIM Configuration File

Parameter	Description
CONNECT	The ODBC source name for connecting to the database server.
LOG TRANSACTIONS TO FILE	This parameter must be in the header section and the default value is TRUE. Transactions can be logged in a file or a table. By default, EIM logs transactions into files. Log files are saved in the file system's eim directory. If you do not want transactions to be logged in files, then setting this parameter to FALSE logs transactions to a table. **NOTE:** If this parameter is set to TRUE, you must make sure that the Siebel Server can write to the file system's eim directory. During installation, the file system directory must be specified using the Uniform Naming Convention (UNC). For more information, see the *Siebel Installation Guide* for the operating system you are using.
PASSWORD	The database password for the process to be run. This parameter is inherited for the EIM component from the Gateway Name Server, so it should already be set. However, you can specify this in the .IFB file if you are running EIM from the Siebel application (not the command line) and if you have not already set this value in the EIM Server Component parameters. **NOTE:** If you start EIM from the command line, it uses the username and password you used to log into the srvrmgr. If you start EIM from the Siebel application, EIM looks for the username and password in the EIM Server Component parameters first, and if they are not specified, EIM then looks in the .IFB file. If EIM cannot find the username and password in those places, EIM cannot log into the database and it fails. If you do not want your username and password visible in the .IFB file, then specify them in the EIM Server Component parameters.
PROCESS	Identifies the specific process to run during this invocation of EIM. The named process must be defined in the process section of the .IFB file.
[Siebel Interface Manager]	Header section must use this reserved name.

Table 6. General Header Parameters for the EIM Configuration File

Parameter	Description
TABLEOWNER	The database logon name that owns the tables to be operated on; used as the prefix for table names; defined during installation.
USERNAME	The database/employee logon name for the process to be run. This parameter is inherited for the EIM component from the Gateway Name Server, so it should already be set. However, you can specify this in the .IFB file if you are running EIM from the Siebel application (not the command line), and if you have not already set this value in the EIM Server Component parameters. **NOTE:** If you start EIM from the command line, it uses the username and password you used to log into the srvrmgr. If you start EIM from the Siebel application, EIM looks for the username and password in the EIM Server Component parameters first, and if they are not specified, EIM then looks in the .IFB file. If EIM cannot find the username and password in those places, EIM cannot log into the database and it fails. If you do not want your username and password visible in the .IFB file, then specify them in the EIM Server Component parameters.

Process Section Parameters Generic to All EIM Processes

This section contains general process parameters generic to all EIM processes that appear in the process section of the EIM configuration file. Table 7 provides descriptions of these parameters.

NOTE: If your configuration file has more than one process section and you want a certain parameter to act on more than one process, you must include the parameter setting within each of the process sections that correspond to the processes on which you intend for the parameter to act.

Table 7. General Process Parameters for the EIM Configuration File

Parameter	Description

BATCH	Required. Specifies a required batch number for the process to be run. Use this batch number to identify the set of rows to load from the EIM tables for this specific process. This batch number corresponds to the value in the interface column IF_ROW_BATCH_NUM and must be a positive integer between 0 and 2147483647 (no commas). To specify multiple batches, use a range or list of batch numbers.
	
	To specify a range of batches, use the first_batch-last_batch format as shown in this example:
	
	`BATCH=100–120`
	
	To list batches, use the comma-delimited format as shown in this example:
	
	`BATCH=100,103,104`

COMMIT EACH PASS	Optional. Commit after each EIM pass; default is TRUE. **NOTE:** It is best not to use this parameter in delete processes. This is because if a commit occurs after each table or each pass in a delete process, then in case of errors causing exit from the process, you can be left with orphan records and dangling references. If the commit occurs for the whole batch, then in case of errors, you can roll back other table deletes.
COMMIT EACH TABLE	Optional. Commit after each base table; default is TRUE. **NOTE:** It is best not to use this parameter in delete processes. This is because if a commit occurs after each table or each pass in a delete process, then in case of errors causing exit from the process, you can be left with orphan records and dangling references. If the commit occurs for the whole batch, then in case of errors, you can roll back other table deletes.
IGNORE BASE TABLES	Optional. Do not process these tables.

Table 7. General Process Parameters for the EIM Configuration File

Parameter	Description
INCLUDE	Optional. Subprocess to execute. **NOTE:** This parameter can be used only in shell processes. A shell process uses the INCLUDE statement to invoke a sequence of processes in a single run. INCLUDE names a process to be included as part of this process. More than one process may be included in another process. All included processes execute before the process itself.
LOG TRANSACTIONS	Optional. Default value depends on system preference. Use this parameter to control the logging mode. If this parameter is set to TRUE, EIM logs changes when mobile clients synchronize. If this parameter is set to FALSE, changes are not logged. In general, when you load data into the HQ database for the first time, this parameter should be set to FALSE. `LOG TRANSACTIONS = TRUE` operates in row-by-row mode. `LOG TRANSACTIONS = FALSE` operates in set-based mode.
ONLY BASE TABLES	Optional. Process only base tables.
ROLLBACK ON ERROR	Optional. Error rollback behavior; default is FALSE.
SESSION SQL	Optional. Specifies a user-defined SQL statement to be sent to the database server before other SQL statements for this process. This string is sent directly to the database and must be a single SQL statement suitable for immediate processing. You can use the SESSION SQL parameter to set tracing for performance analysis. Only one SESSION SQL parameter can be used in each process section. **CAUTION:** *This parameter cannot be used to insert or update data in Siebel base tables.* EIM sends the SQL statement directly to the database and may cause data loss for Siebel Remote and Siebel Replication Manager.
SKIP BU_ID DEFAULT	Optional. Specifies whether the virtual null key is to be skipped for the BU_ID column. The default value is FALSE. Virtual null key sets the BU_ID column value to the default value defined in the repository. To use the default value defined in the repository for the BU_ID column, set this parameter to FALSE (the default). To skip the virtual null key and not use the default value defined in the repository for the BU_ID column, set this parameter to TRUE. This parameter applies to import, delete, and merge processes because the foreign key must be resolved before these processes can run.

Table 7. General Process Parameters for the EIM Configuration File

Parameter	Description

TABLE	Required. Specifies the name of an EIM table used in this process. Multiple TABLE parameters may be used to define a process using more than one table.
	
	Example:
	
	`TYPE = EXPORT`
	
	`BATCH = 101`
	
	`TABLE = EIM_ACCOUNT`
	
	`EXPORT MATCHES = S_ORG_EXT, (NAME > 'A')`
	
	NOTE: For performance reasons, you should limit the number of tables to export or merge in a single process section to five tables or fewer.

TRANSACTION SQL	Optional. Post-commit SQL statement. Specifies a user-defined SQL statement to be sent to the database before other SQL statements, and immediately after each commit or rollback operation during the process (including subprocesses). For more information about this parameter, see "TRANSACTION SQL Parameter" on page 45.
TYPE	Required. This parameter specifies the type of process being defined (possible values are IMPORT, EXPORT, DELETE, MERGE, SHELL). A shell process uses the INCLUDE statement to invoke a sequence of processes in a single run.
UPDATE STATISTICS	Optional. For DB2 databases only. Controls whether EIM dynamically updates the statistics of EIM tables. The default value is TRUE. For example, if you are running EIM on a DB2 database, the account under which EIM runs must have the DB2 CONTROL table privilege on the EIM tables. The database installer automatically grants this privilege when creating the tables. However, it may be necessary to regrant this privilege if the EIM tables have been modified or recreated. To regrant the CONTROL privilege, use the script named grantstat.sql in the database installer directory. **NOTE:** If you plan to run EIM processes in parallel on a DB2 database, this may cause a deadlock when multiple EIM processes access the same EIM table simultaneously. To avoid this potential problem, set the UPDATE STATISTICS parameter to FALSE.
USE ESSENTIAL INDEX HINTS	Optional. For MS SQL Server and Oracle databases only. The default value is TRUE. This parameter enables a subset of index hints for MS SQL Server.

Table 7. General Process Parameters for the EIM Configuration File

Parameter	Description
USE INDEX HINTS	Optional. For Oracle databases only. Controls whether EIM issues optimizer hints to the underlying database to improve performance and throughput. The default value is FALSE.
USING SYNONYMS	Optional. Controls the queries of account synonyms during import processing. When set to FALSE, this parameter saves processing time because queries that look up synonyms are not used. The default value is TRUE.

Inheritance Rules for Configuration Parameters

Some configuration parameters can only be used in a process section of a configuration file, not in the header section. The parameters TYPE and ONLY BASE TABLES are two examples of parameters in this category. Parameters that can be used only in a process section only affect that section, and only the process for which they appear.

Most configuration parameters are used in both the header section and the process section of the configuration file—the parameters USE INDEX HINTS and COMMIT EACH PASS are two examples. These parameters follow the inheritance rules that are listed below, using USE INDEX HINTS as an example:

- If you specify USE INDEX HINTS in a configuration file's header section—in [Siebel Interface Manager]—then it will be used for all processes in that configuration file.
- If you specify USE INDEX HINTS in a shell process, then USE INDEX HINTS affects all of the shell's subprocesses when running that shell process.
- If you specify USE INDEX HINTS in a shell process *and* in its subprocess, then the value from the subprocess will override the value from the shell process.
- If you specify USE INDEX HINTS in any other type of EIM process (import, export, delete, or merge), then USE INDEX HINTS will be used only for that process and not for any other processes that might be listed in the configuration file.
- If you specify USE INDEX HINTS in a configuration file's header section (in [Siebel Interface Manager]) *and* in the process section, the value from the process section will override the value from [Siebel Interface Manager].

Setting EIM Configuration Parameters

Table 6 on page 37 lists the general configuration parameters that can be set when using EIM.

Keep in mind the following points when working with the EIM configuration file:

- Lines in the default.ifb file that begin with a semicolon (;) are comment lines and are ignored.

- If you are continuing a parameter definition to multiple lines in the .IFB file, make certain that the backslash character (\) is the last character on the line. The backslash character denotes continuation. Do not combine comments (;) with new lines (/) because this format creates difficulties finding a comment in the middle of a line.

  CAUTION: When the backslash is followed by a space, EIM interprets the space character as "escaped," and the new line character then terminates the parameter definition. ***This can generate an error message indicating the parameter definition is incomplete.***

  If multiple lines have the backslash (continuation) character (\) at the end, this means they are a single parameter line. So, if a semi-colon (comment character) is placed among these lines, EIM ignores the column with the semi-colon.

  For example:

  ```
  ONLY BASE COLUMNS = S_PARTY.PARTY_TYPE_CD,\

  S_PARTY.PARTY_UID,\

  ; S_PARTY.ROOT_PARTY_FLG,\

  S_CONTACT_FNX.PAR_ROW_ID,\

  S_CONTACT_FNX.X_BATCH_ID
  ```

  These statements will cause EIM to comment off S_PARTY.ROOT_PARTY_FLG.

- PASSWORD and USERNAME values are generally not used for access authentication or as a security measure. EIM acquires access authentication from the component parameters.

  PASSWORD and USERNAME values in the .IFB file are only used if the parameters are not set at the enterprise or component level.

Setting EIM Configuration File Header Parameters

The first nonblank, noncomment line of the configuration file's header section must contain the exact information shown:

```
[Siebel Interface Manager]
```

Table 6 on page 37 lists the other general header parameters to set when using EIM.

Setting EIM Configuration File Process Parameters

This topic describes only the general process parameters, that is, the process parameters that are generic to all EIM processes and that appear in the process section of the EIM configuration file. The process-specific section parameters are described in the chapters that cover each specific EIM process.

Table 7 on page 39 lists the general process parameters to set when using EIM.

The first nonblank, noncomment line of each process section is a bracketed string that specifies the name of the process. This is the name used in the PROCESS argument, or in the RUN PROCESS parameter in the header section. The value between the square brackets ([and]) can contain alphanumeric characters, spaces, and the following punctuation marks:

```
# _ : - $ % / +
```

There are two types of keywords for process section parameters: required keywords and optional keywords.

Required Keywords for Process Parameters

Of the general configuration parameters listed in Table 7 on page 39, note that the following ones are required when using EIM:

- TYPE
- BATCH
- TABLE

Optional Keywords for Process Parameters

Of the general configuration parameters listed in Table 7 on page 39, note that the following ones are optional when using EIM:

- COMMIT EACH PASS
- COMMIT EACH TABLE
- IGNORE BASE TABLES
- INCLUDE
- LOG TRANSACTIONS
- ONLY BASE TABLES
- ROLLBACK ON ERROR
- SKIP BU_ID DEFAULT
- SESSION SQL
- TRANSACTION SQL
- UPDATE STATISTICS
- USE ESSENTIAL INDEX HINTS
- USE INDEX HINTS
- USING SYNONYMS

TRANSACTION SQL Parameter

This parameter specifies a user-defined SQL statement to be sent to the database before other SQL statements and immediately after each commit or rollback operation during the process (including subprocesses). Although a commit operation is processed first, this statement is emitted (for the first time) immediately after the SESSION SQL parameter. Only one TRANSACTION SQL parameter can be used in each process section.

You must define the rollback of the EIM process by doing either of the following:

- Add the TRANSACTION SQL parameter in the configuration file.
- Use the Server Manager to set the Database Rollback Segment Name parameter of the Enterprise Integration Mgr component at the component level.

To avoid errors, do not specify the rollback segment:

- When using the siebenv.bat file.
- At the task level.
- When using both the configuration file and the Server Manager.

NOTE: Do not use the TRANSACTION SQL parameter to insert or update data in Siebel base tables.

To define the rollback segment in the configuration file

- Add a line (as shown in the following example for an Oracle database) to the EIM configuration file.

```
TRANSACTION SQL = "set transaction use rollback segment rb_big"
```

To define the rollback segment using the Server Manager

1. Navigate to Administration - Server Configuration > Servers > Components > Parameters.
2. In the Components list, select Enterprise Integration Mgr.
3. Click the Component Parameters view tab.
4. In the Component Parameters list, select Database Rollback Segment Name.
5. In the Current Value field, type the name of the rollback segment to be used and click Save.

For more information on using the Server Manager, see *Siebel System Administration Guide*.

Setting Extended EIM Configuration Parameters

You can dynamically name and define extended parameters. This section explains how to use extended parameters in the EIM configuration file.

User-Defined Extended Parameters

Use extended parameters to create new parameter names and define values. You can define extended parameters using either the GUI or the command-line interface. User-defined extended parameters use the $*name*=*value* format inside the EIM configuration file, and the *name*=*value* format in the GUI or the command-line interface. The parameter can be a character string consisting of any alphanumeric characters; the underscore symbol (_) can also be used.

To define extended parameters using the GUI

1. Navigate to Administration - Server Configuration > Enterprises > Parameters.
2. Click the Component Requests view tab.
3. In the Component Requests form, click the menu button, and then click New Record.
4. In the Component/Job field, click the select button.
5. In the Component/Jobs window, select the Enterprise Integration Mgr component, and then click OK.

   If you want to use a component job based on EIM for your component request, you must first define the component job. For information on defining component jobs, see *Siebel System Administration Guide*.
6. Complete the rest of the fields and click Save.
7. In the Component Request Parameters list, click the menu button and then New Record.
8. In the Name field, click the Select button.
9. In the Job Parameters window, select Extended Parameters, and then click OK.
10. In the Value field, type in extended parameters using the comma-delimited format *name=value,name=value* as shown in the following example:

    ```
    ACCT_NAME=COMPAQ,ACCT_NUM=01101,ACCT_CONTACT=John Dove, CONTACT_PHONE=(987)123-4567
    ```

    If you are defining multiple values for an extended parameter, you need to enclose the values in double quotes preceded by a backslash as shown in the following example:

    ```
    \"BatchNum1=20001"
    ```
11. Click Save.
12. In the Component Requests form, click the menu button, and then click Submit Request.

Figure 5 shows an example of defining extended parameters as described.

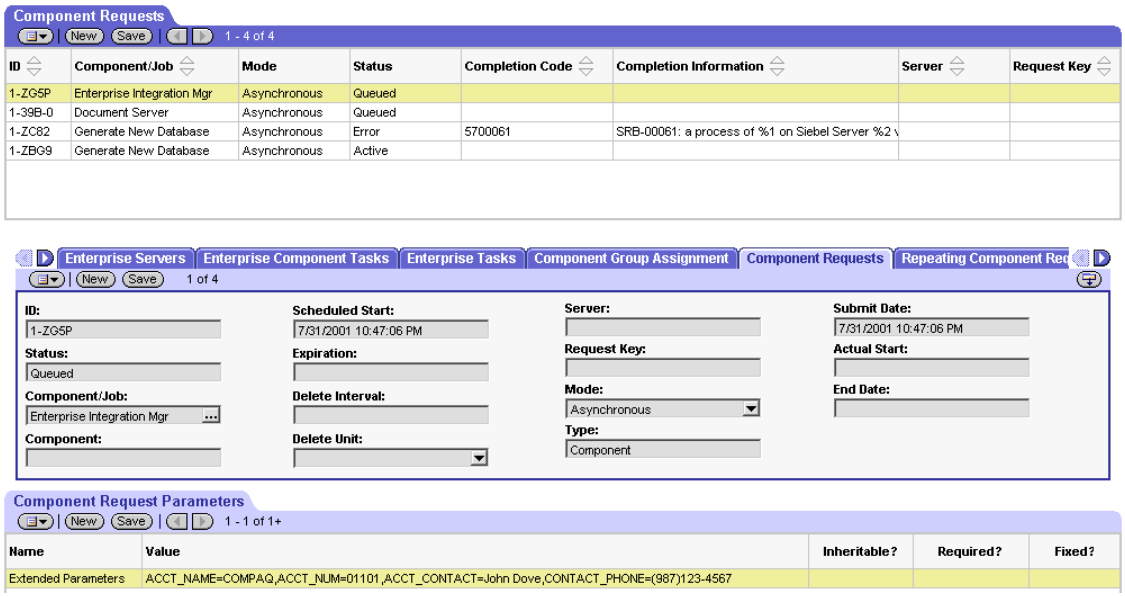


Figure 5. Defining Extended Parameters Using the GUI

To define extended parameters using the command-line interface

1 Use the reserved keyword ExtendedParams to define the *name=value* format as shown in the following example:

```
ExtendedParams="ACCT_NAME=COMPAQ,ACCT_NUM=01101, ACCT_CONTACT=John
Dove,CONTACT_PHONE=(987)123-4567"
```

NOTE: You must enter extended parameters in double quotes when using the Server Manager command-line interface.

2 Run EIM to test the extended parameters.

Predefined Extended Parameters

Some extended parameters are predefined in Siebel applications. These parameters also use the *name=value* format. Table 8 lists these predefined extended parameters.

Table 8. Predefined Extended Parameters

Parameter	Description	Example
CURRENT_USER	Logon name of current user	CURRENT_USER =Customer1
PASSWORD	Password of current user	PASSWORD=ABC

Table 8. Predefined Extended Parameters

Parameter	Description	Example
CURRENT_DATETIME	Current date and time information	`CURRENT_DATETIME=11/3/98_22:45`
ROOT_DIR	Home directory of Siebel server	`ROOT_DIR=Siebel`
SIEBEL_FILE_DIR	Siebel file system	`SIEBEL_FILE_DIR=Files`
LANGUAGE	Language of Siebel server installation	`LANGUAGE=English`
TABLE_OWNER	Name of tableowner	`TABLE_OWNER=ora22`
ODBC_DATA_SOURCE	Connect string for ODBC data source	`ODBC_DATA_SOURCE=sun1`
MAX_NEST_SUBST	Maximum level of nesting in parameter substitutions. The default value is 10.	`MAX_NEST_SUBST=10`
NUM_IFTABLE_LOAD_CUTOFF	When this parameter is enabled, EIM loads all schema mappings if the value is less than the number of EIM tables used in the run process. To enable, set the value to a positive number that is less than the number of EIM tables used in the run process. For example, if the EIM process is using one EIM table, then the setting should be `NUM_IFTABLE_LOAD_CUTOFF = 0`. When disabled, EIM loads only mappings for EIM tables used in the run process. This speeds up the dictionary loading process in EIM. To disable, set the value to -1. This feature is disabled by default. For more information, see "Example of Using the NUM_IFTABLE_LOAD_CUTOFF Parameter" on page 206.	`NUM_IFTABLE_LOAD_CUTOFF=-1`
IfbFileName	Name of the .IFB file where resolved parameters are stored.	`IfbFileName=TEST`
TraceFlags	Contains logs of various EIM operations. Available TraceFlags include 1, 2, 4, 8, and 32. For descriptions of available TraceFlags, see "Trace Flags" on page 130.	`TraceFlags=2`

Sample SQL Scripts

Use the following sample SQL scripts as a starting point for your own scripts. These scripts each provide an example of the data that is necessary when loading account and contact records. Sample scripts are provided for the following RDBMSs:

DB2 Sample SQL Script

```
insert into Siebel.EIM_ACCOUNT

(ROW_ID, IF_ROW_BATCH_NUM,IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD,
ROOT_PARTY_FLG,PARTY_NAME, NAME, MAIN_PH_NUM, LOC, ACCNT_BU, ACTIVE_FLG,
DISA_CLEANSE_FLG, EVT_LOC_FLG, FCST_ORG_FLG, INT_ORG_FLG, PROSPECT_FLG, PRTNR_FLG,
PRTNR_PUBLISH_FLG, RPLCD_WTH_CMPT_FLG, SKIP_PO_CRDCHK_FLG)

values

('100', '100','FOR_IMPORT', 'AUID1', 'ACD1', 'Y', 'Party1', 'Account1',
'6505511784','HQ', 'Default Organization', 'Y', 'Y','Y','Y','Y','Y',
'Y','Y','Y','Y');

insert into Siebel.EIM_CONTACT

(ROW_ID, IF_ROW_BATCH_NUM, IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD, ROOT_PARTY_FLG,
ADDR_NAME, DEPT_ACCNT_BU, DEPT_ACCNT_LOC, DEPT_ACCNT_NAME,  CON_PERSON_UID, CON_BU,
CON_ACTIVE_FLG, CON_DISACLEANSEFLG, CON_DISPIMGAUTHFLG, CON_EMAILSRUPD_FLG,
CON_EMP_FLG, CON_FST_NAME, CON_LAST_NAME, CON_PO_PAY_FLG, CON_PRIV_FLG,
CON_PROSPECT_FLG, CON_PTSHPCONTACTFL, CON_PTSHPKEYCONFLG, CON_SUPPRESSEMAILF,
CON_SUPPRESSFAXFLG, CLINT_ACCNT_BU, CLINT_ACCNT_LOC, CLINT_ACCNT_NAME,
PP_PARTY_TYPE_CD, PP_PARTY_UID, PP_REF_FLG, PP_START_DT)

values

('200', '200', 'FOR_IMPORT', 'CUID1', 'CCD1', 'Y', 'Address1', 'Default
Organization', 'HQ', 'Account1', 'CONUID1', 'Default Organization',
'Y','Y','Y','Y','Y','Tom','Hanks', 'Y','Y','Y','Y','Y','Y','Y','Default
Organization', 'CrossRoads', 'Account2', 'ACD1', 'AUID1', 'Y', '2000-05-17-
15.40.55.000000');
```

MS SQL Sample SQL Script

```
insert into dbo.EIM_ACCOUNT
```

```
(ROW_ID, IF_ROW_BATCH_NUM,IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD,
ROOT_PARTY_FLG,PARTY_NAME, NAME, MAIN_PH_NUM, LOC, ACCNT_BU, ACTIVE_FLG,
DISA_CLEANSE_FLG, EVT_LOC_FLG, FCST_ORG_FLG, INT_ORG_FLG, PROSPECT_FLG, PRTNR_FLG,
PRTNR_PUBLISH_FLG, RPLCD_WTH_CMPT_FLG, SKIP_PO_CRDCHK_FLG)

values

('100', '100','FOR_IMPORT', 'AUID1', 'ACD1', 'Y', 'Party1', 'Account1',
'6505511784','HQ', 'Default Organization', 'Y', 'Y','Y','Y','Y','Y',
'Y','Y','Y','Y')

insert into dbo.EIM_CONTACT

(ROW_ID, IF_ROW_BATCH_NUM, IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD, ROOT_PARTY_FLG,
ADDR_NAME, DEPT_ACCNT_BU, DEPT_ACCNT_LOC, DEPT_ACCNT_NAME,  CON_PERSON_UID, CON_BU,
CON_ACTIVE_FLG, CON_DISACLEANSEFLG, CON_DISPIMGAUTHFLG, CON_EMAILSRUPD_FLG,
CON_EMP_FLG, CON_FST_NAME, CON_LAST_NAME, CON_PO_PAY_FLG, CON_PRIV_FLG,
CON_PROSPECT_FLG, CON_PTSHPCONTACTFL, CON_PTSHPKEYCONFLG, CON_SUPPRESSEMAILF,
CON_SUPPRESSFAXFLG, CLINT_ACCNT_BU, CLINT_ACCNT_LOC, CLINT_ACCNT_NAME,
PP_PARTY_TYPE_CD, PP_PARTY_UID, PP_REF_FLG, PP_START_DT)

values

('200', '200', 'FOR_IMPORT', 'CUID1', 'CCD1', 'Y', 'Address1', 'Default
Organization', 'HQ', 'Account1', 'CONUID1', 'Default Organization',
'Y','Y','Y','Y','Y','Tom','Hanks', 'Y','Y','Y','Y','Y','Y','Y','Default
Organization', 'CrossRoads', 'Account2', 'ACD1', 'AUID1', 'Y', '02-FEB-2002')

insert into EIM_ACCOUNT

(ROW_ID, IF_ROW_BATCH_NUM,IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD,
ROOT_PARTY_FLG,PARTY_NAME, NAME, MAIN_PH_NUM, LOC, ACCNT_BU, ACTIVE_FLG,
DISA_CLEANSE_FLG, EVT_LOC_FLG, FCST_ORG_FLG, INT_ORG_FLG, PROSPECT_FLG, PRTNR_FLG,
PRTNR_PUBLISH_FLG, RPLCD_WTH_CMPT_FLG, SKIP_PO_CRDCHK_FLG)

values

('100', '100','FOR_IMPORT', 'AUID1', 'ACD1', 'Y', 'Party1', 'Account1',
'6505511784','HQ', 'Default Organization', 'Y', 'Y','Y','Y','Y','Y',
'Y','Y','Y','Y');

insert into EIM_CONTACT

(ROW_ID, IF_ROW_BATCH_NUM, IF_ROW_STAT, PARTY_UID, PARTY_TYPE_CD, ROOT_PARTY_FLG,
ADDR_NAME, DEPT_ACCNT_BU, DEPT_ACCNT_LOC, DEPT_ACCNT_NAME,  CON_PERSON_UID, CON_BU,
CON_ACTIVE_FLG, CON_DISACLEANSEFLG, CON_DISPIMGAUTHFLG, CON_EMAILSRUPD_FLG,
CON_EMP_FLG, CON_FST_NAME, CON_LAST_NAME, CON_PO_PAY_FLG, CON_PRIV_FLG,
CON_PROSPECT_FLG, CON_PTSHPCONTACTFL, CON_PTSHPKEYCONFLG, CON_SUPPRESSEMAILF,
CON_SUPPRESSFAXFLG, CLINT_ACCNT_BU, CLINT_ACCNT_LOC, CLINT_ACCNT_NAME,
PP_PARTY_TYPE_CD, PP_PARTY_UID, PP_REF_FLG, PP_START_DT)

values
```

```
('200', '200', 'FOR_IMPORT', 'CUID1', 'CCD1', 'Y', 'Address1', 'Default
Organization', 'HQ', 'Account1', 'CONUID1', 'Default Organization',
'Y','Y','Y','Y','Y','Tom','Hanks', 'Y','Y','Y','Y','Y','Y','Y','Default
Organization', 'CrossRoads', 'Account2', 'ACD1', 'AUID1', 'Y', '02-FEB-2002');
```

5 Importing Data

Importing data into Siebel base tables is a multistep process that requires significant effort. You must first load data from an external database into the EIM tables. Subsequently, you need to run an EIM process to read the data in these EIM tables and import them into the appropriate Siebel base tables.

This chapter is organized into the following sections:

EIM Import Process

To import tables of data, EIM performs a sequence of tasks. Each task involves multiple passes; at least one pass is required for each EIM table included in the process. Depending on the type of import process, EIM may repeat several tasks.

This section describes the general tasks that EIM performs to import data into the Siebel database using EIM. To see the general steps that *you* take when using EIM to import data, see "Import Data Process Flow" on page 55.

To import data from EIM tables, EIM performs the following steps:

1. EIM initializes any temporary columns:
   - It compares values in IF_ROW_BATCH_NUM with the batch number provided by the Component task that initiated this import process. For information on IF_ROW_BATCH_NUM, see "Mandatory Columns for EIM Processing" on page 20.
   - It sets all temporary columns to NULL and counts the rows to be processed.

   NOTE: If there are rows where required columns contain only blanks, the complete EIM process will fail at this step. Rows will not be imported or updated.
2. EIM applies any DEFAULT_COLUMN and FIXED_COLUMN values defined for this import process. For information on DEFAULT_COLUMN and FIXED_COLUMN, see "Parameters Used for Imports in Both the Header and Process Sections" on page 67.

3 EIM applies any filter queries defined for this import process. If a row fails the filter query, EIM eliminates the row from further processing.

4 EIM generates foreign key references for rows with corresponding existing rows in the Siebel base tables. It writes these foreign key values into EIM table temporary columns.

   If foreign keys fail for required columns, EIM eliminates these rows from further processing. It also validates bounded picklist values against the List of Values table (S_LST_OF_VAL). For this validation to occur, the List of Values must be specified at the table level, and not just at the business component level. For more information on bounded and unbounded picklists, see *Configuring Siebel eBusiness Applications*.

5 EIM writes the appropriate ROW_ID values in the EIM table rows' temporary columns, for rows with corresponding base table rows. For information on ROW_ID, see "Mandatory Columns for EIM Processing" on page 20.

6 EIM creates a ROW_ID with a unique value in the base table for each EIM table row without a corresponding row in the base tables.

7 EIM eliminates rows with invalid values for user keys from further processing.

   NOTE: You can use EIM to update only non-user key columns; EIM does not support modification of existing user key columns. To update user key columns in S_ORG_EXT and S_PROD_INT tables use EIM_ORG_EXT_UK and EIM_PROD_INT_UK. For more information, see "Fields That Cannot Be Updated" on page 62.

   It then generates foreign key references for rows without corresponding rows in the Siebel database tables, and writes these foreign key values into EIM table temporary columns:

   - If foreign keys fail for required columns, EIM eliminates these rows from further processing.
   - For EIM table rows with data that will reside in multiple destination tables, EIM fails rows with foreign keys that cannot be generated.

8 EIM updates contents of existing base table rows with contents from corresponding EIM table rows that have successfully passed all earlier steps:

   - If any rows contain content that differs from the existing base table row, EIM writes these rows to the Master Transaction Log (if Docking: Transaction Logging is enabled).
   - If multiple EIM table rows have the same user primary key for a base table, EIM uses only the first EIM table row to update the base table, and ignores the data in other rows.

9 EIM inserts any new EIM table rows that have successfully passed all earlier steps in the Siebel database tables:

   - It writes new rows to the Master Transaction Log (if Docking: Transaction Logging is enabled).
   - If multiple EIM table rows use the same user primary key for a base table, EIM uses only the first EIM table row to update the base table, and ignores the data in other rows.

10 EIM updates primary child relationships in the Siebel database tables as necessary. EIM populates all primary child columns with Primary Child Col property set to TRUE. For information on primary child relationships, see "About Explicit Primary Mappings" on page 24.

CAUTION: You may want to use the UPDATE ROWS = FALSE statement to preserve existing information. Suppressing updates prevents updating primaries in this step of the import process, so this setting should be used with caution. For more information, see "Suppressing Updates" on page 75.

11 Finally, EIM runs optional miscellaneous SQL statements. For more information, see the section on the MISC SQL parameter in "Parameters Used for Imports in Both the Header and Process Sections" on page 67.

Import Data Process Flow

This section describes the general process flow that you must follow to import data into the Siebel database using EIM.

NOTE: Running an import process can be a substantial effort that may require the time of key personnel, as well as significant resources.

1 **Identify and validate the data to be imported.** To perform this task, you must:

- Determine the data to load and whether it already exists in another database. You should review existing data for completeness. For example, the Siebel database may require both an area code and a telephone number, while your existing database may not.
- Determine the number of opportunities, contacts, and accounts you plan to import. This information assists you in estimating the time and resources required to import, process, and store your data.

NOTE: If the data exists in a database that uses a different character set, the import process does not work properly until you recreate the database.

2 **Identify the column mappings and user key columns of the data to be imported.** To perform this task, you must:

- Identify the mapping between the data and Siebel base columns. For information on Siebel base table columns, see *Siebel Data Model Reference*.
- Identify the EIM table columns that map to these base table columns. To view mappings between EIM table columns and base table columns, see "EIM Table and Column Mappings" on page 22.
- Identify the user key columns and make sure they are populated uniquely. For information on user key columns, see *Siebel Data Model Reference*.

3 **Make sure that your hardware and software environments are ready.** Before you use Siebel EIM tables to import data, the Siebel application must be properly installed.

Work with your Siebel representative and MIS personnel to verify that the required hardware and software resources are available. For information about resource requirements, see "Importing Large Databases" on page 60.

4 **Back up your existing database.** Before undertaking any significant change—such as installing a new application, importing data, or upgrading an installed application—you should first perform a comprehensive backup of your database. This facilitates an easy recovery if problems occur.

5 **Copy file attachments to the Siebel server subdirectory named "input."** If you want to import file attachments, you can:

- Copy the files to the input subdirectory under the Siebel server root directory.
- Store file attachments in the location specified in the ATTACHMENT DIRECTORY .IFB file header parameter.

Siebel EIM tables support all file attachment formats, including common file types such as Word documents (.doc), Excel spreadsheets (.xls), and text files (.txt). For information on file attachment columns, see "File Attachment Columns" on page 21.

6 **Load and verify the EIM tables.** Your database administrator can use a database tool provided with your RDBMS (such as SQL*Loader, Bulk Copy Utility, or dbload) to copy data from your existing database to the Siebel EIM tables.

NOTE: Siebel EIM tables contain several special columns that must be populated before rows can be imported. For more information, see "EIM Table Columns" on page 20.

- After the EIM tables are loaded, check the number of loaded rows against your existing database to make sure that the appropriate rows were loaded.
- Check the contents of several rows to make sure that the tables are ready for the import process.

For information on preparing the EIM tables for data import, see "Preparing the EIM Tables for Import Processing" on page 62.

7 **Edit the EIM configuration file (default.ifb).** This file customizes the behavior of EIM by defining the data you will import and identifying the batch number to use.

For information on editing the EIM configuration file for data import, see "Using the EIM Configuration File to Define a Process" on page 35.

8 **Test your import process**. Run a small test batch (perhaps 100 records) to verify that the EIM tables load correctly, and that the correct parameters are set in the configuration file and on the srvrmgr command line.

For information on testing your import process, see "Testing EIM Processes" on page 145.

9 **Run the import process**. Although your batch sizes depend on the volume of data you must import, consider using multiple smaller batches (1,000 to 5,000 rows) rather than one large batch. Smaller batches place fewer demands on resources. Also, when using smaller batches, the fixing of problems is simpler. If a batch is not imported correctly, it is easier to isolate the condition, correct it, and rerun the batch.

For more information on this step, see "Running an Import Process" on page 89.

10 **Verify results.** EIM provides several diagnostic tools that let you verify the success of import processing. For information on these tools, see "Checking Import Results" on page 89.

You must test and run the import process and verify the results for each batch you are importing. If an import process failure occurs, see "Troubleshooting Import Processing Failures" on page 91 for descriptions of problems that can cause failures.

EIM provides comprehensive status information about each import process. When a process ends, you should review the information as described in "Checking Import Results" on page 89.

Importing Legacy Data

This section describes the general concepts and procedures for importing legacy data into the Siebel database using EIM.

Recommended Import Order for Importing Legacy Data

The order in which legacy data is imported is critical to make sure that relationships between dependent data elements are established correctly. Siebel EIM tables do not map one-to-one with Siebel target database tables.

NOTE: The recommended import order that follows is a general guideline. Your own data import process may require a different order.

To make sure that the necessary data is present to establish relationships between data entities, use the following sequence to import data:

1. Administrative

   NOTE: An example of administrative data would be a List of Values for Currency or Zip Code.

2. Business Unit
3. Positions
4. Accounts
5. Contacts
6. Employees
7. Products
8. Opportunities
9. Personal Accounts
10. Quotes
11. Documents
12. Forecasts
13. Fulfillment
14. Marketing Campaigns
15. CPG Promotion Management

16 CPG Product Movement

17 Service Requests

18 Product Defects

19 Activities and Appointments

20 Notes

21 File Attachments

This import order reflects most import processes. In some cases, the import order for your import process may vary slightly depending on your requirements.

NOTE: Your Siebel application provides a sample configuration file named default.ifb. You can also use the import sequence in this sample file in your configuration file.

While the import order is most critical when performing the initial import of legacy data, this recommended order should be followed for all subsequent data imports as well.

NOTE: Some tables cannot be used to import all data necessary for the imported data to be visible in the GUI. For example, the interface table EIM_FCSTOPTYPRD can be used to export forecast data but it cannot be used for importing. The import runs successfully, but the imported data cannot be seen in the GUI because EIM does not populate the table that would make the data visible.

Importing an Initial Batch of Legacy Data

When you are importing an initial batch of legacy data, you need to complete the following procedure.

To import initial batches of data

1. In the EIM table, assign a unique batch number to each batch of data in the IF_ROW_BATCH_NUM column.
2. Disable the Docking: Transaction Logging preference.

   NOTE: Typically, initial data loads require transaction logging to be turned off. Siebel Mobile Web Clients will receive their updates during this initial data load.

   a. Navigate to Administration - Application > System Preferences.

   b. Select the Docking: Transaction Logging system preference.

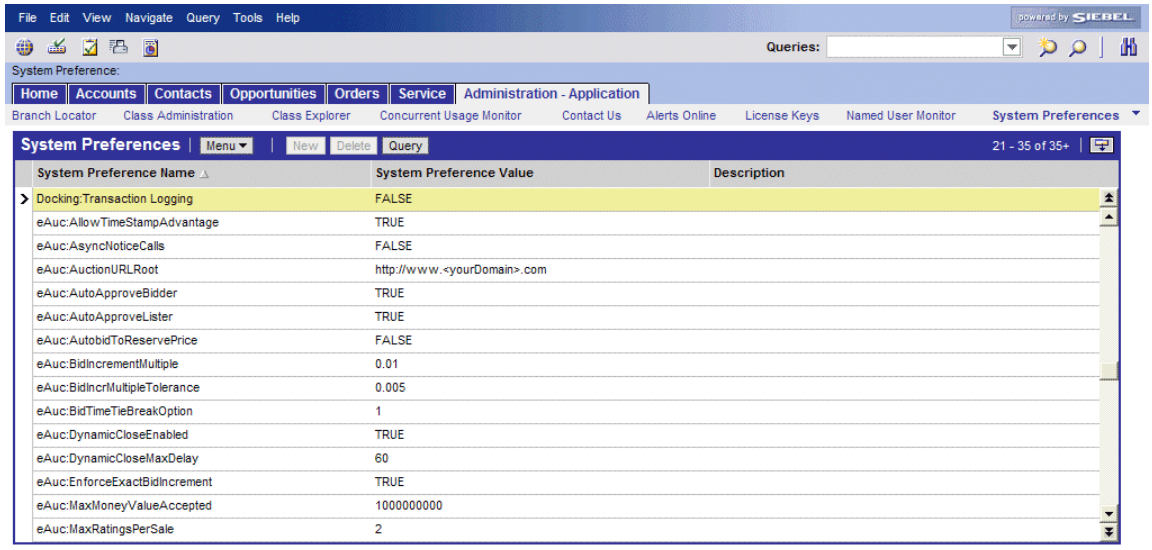
   c. In the System Preference Value field, type FALSE.

      Do not change this value to TRUE until after you import all the initial data.

d Click Save.

You can also change the transaction logging preference by changing the LOG TRANSACTIONS parameter in the EIM configuration file. For more information, see "Process Section Parameters Generic to All EIM Processes" on page 39.

The following figure shows an example of disabling the Docking: Transaction Logging Preference in the System Preferences view.



3 Start an EIM task for each batch number.

For information on running an EIM process, see "Running an Import Process" on page 89.

4 Review your import processes by using the log file produced by EIM (EIM_task#.log).

This file contains comprehensive status and diagnostic information about the import processes. By default, this file is located in the Siebel server log directory.

Using ACT! for Legacy Data Imports

One of the options for importing bulk data from a legacy system into the Siebel database is to use ACT!

- ACT! 2.0 and ACT! 3.0 are the only versions that have File/Import functionality for data import into Siebel eBusiness applications.
- You can use "Exporter for ACT!" to export ACT! 4.0 or 2000Contacts, Notes/History, Activity, Group, Sales and E-Mail data into comma-delimited files.

For information on ACT! products, visit their official Web site.

Importing Large Databases

Before importing a large database, such as a legacy database, you should thoroughly test your import processes. Once the test batches are loaded correctly and any data discrepancies that may affect other batches are resolved, you may want to consider importing large batches for the remaining data. Before doing so, first make sure that the Siebel database is capable of storing the volume of data, and that your resources are adequate to support the processing.

Memory Resources Needed for EIM

To achieve and maintain high performance, the database memory area needs to be large enough to hold most of the frequently accessed data in the cache. Because a very large EIM batch may flush all the data from the cache and cause performance degradation, limit EIM batch sizes so the most frequently accessed data can remain in memory.

Database Resources Needed for EIM

EIM uses database server space for the EIM tables, target base tables, secondary tables, and work areas. To make sure that an import process runs smoothly to completion, you must anticipate and plan for these space requirements. Actual requirements vary based on the RDBMS you are using and the size of the database you are populating. Work with your Siebel representative and database administrator to develop a database blueprint that addresses the following resource requirements:

- **Base tables and indexes.** When establishing appropriate sizes for the Siebel base tables and indexes, consider not only current size, but also reasonable growth. You should plan for future changes that may affect the database, such as organization expansion, new product lines, and company acquisitions. For more information on table sizing, see the documentation for your RDBMS.
- **Secondary tables.** You may be importing data from a single EIM table into multiple destination tables. For each EIM table (except EIM_NOTE), there is a primary, or target, Siebel base table. In addition, there may be one or more secondary tables associated with the target table. Data from the EIM table may ultimately reside in one of these secondary tables.
- **Database manager transaction logging area.** The database manager uses a disk area to log its transactions. If you fail to set an adequately sized logging area for this operation, the database manager halts when the area runs out of space.
- **Transaction rollback areas.** Database resources are temporarily allocated to store intermediate results used to recover the original database state if a transaction is rolled back or aborted. Each RDBMS may use a different implementation. The amount of data processed in a transaction determines the amount of database resources required for rollback areas. Make sure that you allocate sufficient resources, or use smaller batch sizes, to handle the rollback requirements. Your database administrator can configure your database to allocate adequate transaction rollback areas.

After working with small batches to make sure that your import processes run smoothly, you may want to initiate an unattended session in which EIM runs multiple import processes to load a large database.

Updating the Siebel Database

After you have completed the initial import of enterprise data, you can periodically use EIM to update the Siebel database. For example, if you add a new product line, it may be efficient to load the data into your enterprise inventory management database and then import it into the Siebel database. Use the steps described in "Import Data Process Flow" on page 55, although the scope of the update import is usually significantly smaller than that of an initial data import.

CAUTION: If you have active mobile Web clients, do *not* disable the Docking: Transaction Logging system preference. If you disable this system preference, the server database and mobile Web client databases will not be synchronized after the import.

By default, when importing information, EIM performs both inserts and updates based on the content of the batch set. EIM first examines the set of information to determine which rows in the batch already exist in the Siebel database:

- Batch rows matching existing base rows are used to update the database.
- Batch rows that do not match base rows are used to perform inserts.

See "INSERT ROWS and UPDATE ROWS Parameters" on page 73 for further information.

In some circumstances, you may need to suppress inserts and updates. For more information on adjusting parameters to suppress an insert or update, see "Suppressing Data When Updating Existing Databases" on page 74.

NOTE: You can use EIM to update only non-user key columns; EIM does not support modification of existing user key columns. To update user key columns in the S_ORG_EXT and S_PROD_INT tables, use EIM_ORG_EXT_UK and EIM_PROD_INT_UK. For more information, see "Fields That Cannot Be Updated" on page 62.

Updating Siebel Database for Batches with Both an Insert and Update to the Same Record

You may need to update the Siebel database with a batch that contains a record to be inserted as well as an update to that same row. When you use EIM to do this, a record will be inserted, but the update will be flagged as a duplicate.

EIM processes a record once for each batch, so for each record, MIN(ROW_ID) is processed, and the other record is marked as a duplicate (IF_ROW_STAT is set to DUP_RECORD_IN_EIM_TBL for the duplicate record). If you enter the user key of a record with different attributes twice in the EIM table, only the record with the MIN(ROW_ID) will be imported or updated. The duplicate will be ignored.

To avoid this situation, analyze the input records before beginning the EIM task. If you find duplicate records, you can either combine them into one record, or specify a different batch number for the duplicate record so as to process the update in a separate batch. For more information, see "Separating EIM Processes by Operation" on page 148.

Fields That Cannot Be Updated

You cannot update system fields. All Siebel system fields are fields reserved only for use by Siebel Systems Inc. for internal Siebel processes. They are not to be populated with customer data.

The following are reserved system fields that cannot be updated:

- CONFLICT_ID
- CREATED
- CREATED_BY
- LAST_UPD
- LAST_UPD_BY
- MODIFICATION_NUM
- ROW_ID

Preparing the EIM Tables for Import Processing

This section explains how to prepare the EIM tables for a subsequent import into a Siebel database. To import data, EIM reads data in the EIM tables and writes data in the appropriate Siebel base tables by making multiple passes through the EIM tables to:

- Set initial values for some columns in the EIM tables
  - When importing new data, make sure to populate the columns marked Required in the EIM table.
  - When updating existing records you do not need to populate the Required columns, but the user key columns must be populated.

  To find which columns are required, and which columns are user keys, generate a table mapping report. See "Generating EIM Table Mapping Reports" on page 29.
- Apply filter logic to select rows for importing
- Generate foreign key references and internal values
- Add or update relevant Siebel database rows
- Update each EIM table row to indicate its import status

For general information on EIM tables, see Chapter 3, "Siebel EIM Tables."

Required Initial Values for Special Columns

Each row to be imported must contain the data you want to import and the appropriate values in the following columns:

ROW_ID. This value, in combination with the nonempty contents of IF_ROW_BATCH_NUM, must yield a unique value.

IF_ROW_BATCH_NUM. Set this value to an identifying number for all rows to be processed as a batch.

IF_ROW_STAT. In each row to be imported, set this column to FOR_IMPORT to indicate that the row has not been imported. After processing, if certain rows were not imported due to a data error, you should change:

- IF_ROW_BATCH_NUM value for the rows that require reimporting
- BATCH parameter in the configuration file

For more information on special columns, see "EIM Table Columns" on page 20.

Required Initial Values for File Attachment Columns

Each file attachment row must contain the filename reference to the files you want to import and the appropriate values in the following columns:

FILE_NAME. Set this column to the root filename of the file attachment.

FILE_EXT. Set this column to the extension type of the file attachment (such as DOC, XLS, or TXT).

FILE_SRC_TYPE. This column must be set to FILE.

For more information on file attachment columns, see "File Attachment Columns" on page 21.

Adjusting the Case of Values

EIM supports various case values defined for base table columns in Siebel Tools. EIM adjusts the case value of an EIM table column according to the Force Case property of the corresponding base table column.

NOTE: The case values supported by EIM are listed in the Force Case property of the Column object in Siebel Tools. Force Case is a protected property that you cannot change.

Prior to importing data into base table columns, EIM also adjusts the case of values in EIM table columns as defined in the list of values. The available case modes include:

- Upper (Makes all letters uppercase)
- Lower (Makes all letters lowercase)
- FirstUpper (Makes the first letter of each word uppercase and leaves other letters unchanged)
- None (Has no effect)

NOTE: Letters are defined as *A* through *Z* (ASCII only). Words are defined as groups of letters separated by spaces (not punctuation).

If a requested case mode is not supported by the database, EIM performs a row-by-row pass through the EIM table to adjust the case of column values and update the row accordingly. If this occurs, you should expect slower import processing.

NOTE: To change the case mode, consult Siebel Expert Services because this requires changing read-only properties defined at the table level.

Editing the Configuration File for Import Processing

This section describes the header and process sections that you need in the EIM configuration file to properly configure EIM for an import process. For general information on the EIM configuration file, see Chapter 4, "EIM Configuration File."

Before import processing begins, you must change the configuration file to support this function. Such changes include:

- Editing the header and process sections and parameters
- Adjusting settings in the configuration file for various purposes. See "Special Considerations for Imports" on page 73.

CAUTION: To prepare for recovery in the event of an unexpected problem, back up your existing database before you begin an import process.

Header Section Parameters Used for Imports

Parameters in the header section generally apply to all types of processes. For a description of the necessary contents in the header section, see "Header Section Parameters Generic to All EIM Processes" on page 37.

Process Section Parameters Used for Imports

Parameters in the process section apply only to that specific process and override any corresponding value in the header section for the specific process. This section describes the parameters used in the process section that are specific to an import process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

Table 9 lists the parameters specific to an import process that appear in the process section of the EIM configuration file. (For the parameters specific to an import process that can appear in both the process section and the header section of the EIM configuration file, see Table 10 on page 67.)

Table 9. Import Process Parameters for the EIM Configuration File - Process Section

Parameter	Description
COMMIT OPERATIONS	Docking Log row commit frequency; default is 0.
FILTER QUERY	SQL preprocess filter query fragment. Example: `FILTER QUERY=(ACCNT_NUM = "1500")` This parameter names a query that runs before the import process. The query prescreens certain rows in the import batch, using data values in the EIM tables. Rows that do not meet the filter criteria are eliminated. The query expression should be a self-contained WHERE clause expression (without the WHERE keyword) and should use only unqualified column names from the EIM table or literal values (such as name is not null). By default, the FILTER QUERY parameter is not used.
IGNORE BASE COLUMNS	Specifies base table columns to be ignored by the import process. Use commas to separate column names, which can be qualified with base table names. Required and user key columns cannot be ignored. Use this parameter to improve performance when updating all but a few columns. The default is to not ignore any base table columns.
IGNORE BASE TABLES	Specifies base tables to be ignored by the import process. Use commas to separate table names. Target tables for EIM tables cannot be ignored. The default is to not ignore any base tables. Use this parameter to improve performance when updating all but a few tables. This parameter affects all EIM tables used in the import process.
ONLY BASE COLUMNS	Specifies and restricts base table columns for the import process. Use commas to separate column names, which can be qualified with base table names. Include all user key columns and required columns. Use this parameter to improve performance when updating many rows but few columns. The default is to process all interface columns mapped to the base table. Example: `ONLY BASE COLUMNS = S_ORG_EXT.NAME, S_ORG_EXT.LOC, S_ORG_EXT.BU_ID`

Table 9. Import Process Parameters for the EIM Configuration File - Process Section

Parameter	Description
ONLY BASE TABLES	Specifies and restricts selected base tables for the import process. Use commas to separate table names. Target tables for EIM tables must be included. The default is to process all base tables into rows that can be imported from the EIM tables. Use this parameter to improve performance when updating only a few tables. This parameter affects all EIM tables used in the import process. Example: `ONLY BASE TABLES = S_CONTACT, S_ORG_EXT`
UPDATE ROWS	Optional base table, TRUE/FALSE toggle; default is TRUE. For more information on the UPDATE ROWS parameter, see "INSERT ROWS and UPDATE ROWS Parameters" on page 73.

NOTE: The ONLY BASE TABLES, IGNORE BASE TABLES, ONLY BASE COLUMNS, and IGNORE BASE COLUMNS parameters can be used to improve EIM performance.

Parameters Used for Imports in Both the Header and Process Sections

Table 10 describes the parameters that can appear in either the header section or a process section, and are specific to an import process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39. (Table 9 on page 65 lists the parameters specific to an import process that appear in only the process section of the EIM configuration file.)

Table 10. Import Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
ATTACHMENT DIRECTORY	(Default = SIEBEL_HOME\INPUT) Specifies the directory to be used for importing attachments. Before specifying a directory, make sure the directory exists on a Siebel Server machine and you have read and write access to the directory. Example: `ATTACHMENT DIRECTORY = SIEBEL_HOME\INPUT`
COMMIT EACH PASS	Specifies whether a separate transaction should be used for each EIM pass through each EIM table. The default value is TRUE, which invokes commits after each pass. This setting helps to reduce the database resources required for the import process and provides a checkpoint to which you can return in the event of unexpected results. **NOTE:** COMMIT EACH PASS works cumulatively with COMMIT EACH TABLE. If you set both COMMIT EACH PASS and COMMIT EACH TABLE to TRUE, a commit will occur at the end of each pass *and* at the end of each table.
COMMIT EACH TABLE	Specifies whether a separate transaction should be used for each EIM table. The default value is TRUE, which invokes commits after each table. This setting helps to reduce the database resources required for the import process. **NOTE:** COMMIT EACH TABLE works cumulatively with COMMIT EACH PASS. If you set both COMMIT EACH PASS and COMMIT EACH TABLE to TRUE, a commit will occur at the end of each pass *and* at the end of each table.
COMMIT OPERATIONS	(Import only.) Specifies the number of insert and update operations to be performed before a commit is invoked. The value for this parameter, an integer greater than zero, prevents the transaction rollback space from overflowing when large data sets are imported. The default for COMMIT OPERATIONS is not set; a commit is thus invoked only at the end of the import by default. This setting is ignored if you have turned off Docking: Transaction Logging. **NOTE:** This parameter is useful only for row-by-row processing (with transaction logging on). It is not used for set-based processing operations.

Table 10. Import Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
DEFAULT COLUMN	(Import only) Specifies a default value for an EIM table column. The syntax is column name, value. Example: `DEFAULT COLUMN = CURCY_CD , "USD"` The given value will be used only if the column is null in the EIM table.
FIXED COLUMN	(Import only.) Specifies the value for an EIM table column. The syntax is the same as for DEFAULT COLUMN. Example: `FIXED COLUMN=ORG_CD, "Commercial"` The given value will be loaded into the Siebel base table, overriding the value in the EIM table column.
INSERT ROWS	Specifies that nonexistent rows in the EIM table be inserted into the Siebel base table. The default value is TRUE. A table name can be specified with insert rows as the first value, separated by a comma. Example: `INSERT ROWS = EIM_ACCOUNT, FALSE` If the named table is an EIM table, as in the example, the setting applies to all Siebel base tables imported from this EIM table. If the named table is a Siebel base table, the setting is applied when data is imported from any EIM table. **NOTE:** The INSERT ROWS parameter must be set to FALSE for any table with an EIM table that does not have mappings to all its required columns, such as S_ORDER for EIM_ORDER_DTL. In this example, when EIM is not able to resolve the EIM_ORDER_DTL row to an existing S_ORDER record, it attempts to insert it as a new S_ORDER record. Since EIM_ORDER_DTL does not have mappings to all the S_ORDER required columns, the process fails with a "Cannot insert null" error. For more information on the INSERT ROWS parameter, see "INSERT ROWS and UPDATE ROWS Parameters" on page 73.

Table 10. Import Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
MISC SQL	Sets specific explicit or implicit primaries, as mentioned in Step 11 on page 55 of the import process. Explicit is when you have specific values to set as primaries. Implicit is when any of a group of values is acceptable. For example, you are importing one account with nine addresses. If any of the addresses is acceptable as being the primary, then set primary to implicit. EIM then selects one of the addresses as primary. If a specific address should be the primary, then set primary to explicit and indicate the primary account by setting its flag column (EIM_ACCOUNT.ACC_PR_ADDR) to Y. **NOTE:** MISC SQL is intended for initial data loading only (with DOCKING TRANSACTIONS = FALSE), because when using MISC SQL to set primary child foreign keys, NO transactions are logged for mobile users. For a list of fields that can be set using the MISC SQL parameter, see "MISC SQL Parameter" on page 71.
NET CHANGE	(Import only.) Specifies the handling of null (non-user key) column values when importing a row that already exists in the Siebel database table. If NET CHANGE = TRUE, the null value will be ignored; otherwise, the column in the base table will be updated with NULL. This parameter is ignored if UPDATE ROWS = FALSE. The default value is TRUE; null attribute values will thus be ignored for existing rows by default. For more information on this parameter, see "NET CHANGE Parameter" on page 69.
ROLLBACK ON ERROR	Specifies whether the current transaction should be rolled back (aborted) when an error, such as an SQL database failure, is encountered. The default value is FALSE. If you set this parameter to TRUE, you should also set COMMIT EACH PASS and COMMIT EACH TABLE to FALSE, and make sure that the database transaction space is large.
TRIM SPACES	(Import only.) Specifies whether the character columns in the EIM tables should have trailing spaces removed before importing. The default value is TRUE.

NET CHANGE Parameter

By default, EIM does not update non-user key columns—that is, columns with a null value. The NET CHANGE parameter specifies the handling of null (non-user key) column values when importing a row that already exists in the Siebel database table. If NET CHANGE = TRUE, the null value will be ignored. If NET CHANGE = FALSE, the column in the base table will be updated with NULL.

NOTE: NET CHANGE = TRUE does not work for long columns. If you want to update a long column, you must use NET CHANGE = FALSE.

Effect of NET CHANGE = FALSE on IF_ROW_STAT

When NET CHANGE = FALSE, there are three possible outcomes:

- For a null value, EIM updates the base table column to NULL and sets the EIM table's IF_ROW_STAT to IMPORTED.
- For a non-null value that is a duplicate, nothing is done to the base table column and the EIM table's IF_ROW_STAT is set to DUP_RECORD_EXISTS.
- For a non-null value that is not a duplicate, EIM updates the base table column with the value in the EIM table and sets IF_ROW_STAT to IMPORTED.

EIM only updates the non-user key columns with NULL if you set the NET CHANGE parameter to FALSE. Also note that when EIM updates non-user key columns with NULL for the columns that had a non-null value beforehand, then the status of IF_ROW_STAT becomes IMPORTED. This is because EIM has performed the update transaction for this table.

The second case mentioned above shows, however, that if a column had a null value beforehand, and EIM has performed the update with all the same records (including this NULL column), then in effect, EIM has ignored this null value and has not performed an update transaction for this NULL column (regardless of whether NET CHANGE is set to FALSE). So in this case, EIM populates IF_ROW_STAT with DUP_RECORD_EXISTS.

If in cases like this you want to update certain columns with NULL, then you can specify the ONLY BASE COLUMNS parameter in the .IFB file.

Example of Using the NET CHANGE Parameter

The following example is part of a sample .IFB file that uses the NET CHANGE parameter:

```
[Siebel Interface Manager]

   USER NAME = "SADMIN"

   PASSWORD = "SADMIN"

   PROCESS = IMPORT ACCOUNT

[IMPORT ACCOUNT]

   TYPE = IMPORT

   BATCH = 1

   TABLE = EIM_ACCOUNT

   NET CHANGE = FALSE
```

MISC SQL Parameter

Table 11 lists the EIM tables that can be used with the MISC SQL parameter, as well as the values that can be set. The table lists the values of the MISC SQL parameter when you want to set a field explicitly. If you want to set the field implicitly, replace the letters EXPR (EXplicit PRimary) with IMPR (IMplicit PRimary). Note that all separators for values are underscores. Tables and values marked "SIA-specific" are only applicable to Siebel Industry Applications.

Table 11. Primaries Supported by the MISC SQL Parameter

Table and Primary Child Foreign Key	MISC SQL Parameter Value for Explicit Primary	Corresponding EIM Table	Comments
S_PROJ.PR_OU_ADDR_ID	EXPR_S_PROJ_PR_OU_ADDR_ID	EIM_PROJECT	No implicit primary
S_OPTY.PR_OU_ADDR_ID	EXPR_S_OPTY_PR_OU_ADDR_ID	EIM_OPTY	No implicit primary
S_OPTY.PR_OU_INDUST_ID	EXPR_S_OPTY_PR_OU_INDUST_ID	EIM_OPTY	
S_CONTACT.PR_HELD_POSTN_ID	EXPR_S_CONTACT_PR_HELD_POSTN_ID	EIM_EMPLOYEE	
S_CONTACT.PR_USERROLE_ID	EXPR_S_CONTACT_PR_USERROLE_ID	EIM_USER	
S_CONTACT.PR_OU_ADDR_ID	EXPR_S_CONTACT_PR_OU_ADDR_ID	EIM_CONTACT	
S_POSTN.PR_POSTN_ADDR_ID	EXPR_S_POSTN_PR_POSTN_ADDR_ID	EIM_POSITION	
S_POSTN.PR_EMP_ID	EXPR_S_POSTN_PR_EMP_ID	EIM_POSITION	
S_ORG_EXT.PR_BL_PER_ID	EXPR_S_ORG_EXT_PR_BL_PER_ID	EIM_ACCOUNT	
S_ORG_EXT.PR_SHIP_PER_ID	EXPR_S_ORG_EXT_PR_SHIP_PER_ID	EIM_ACCOUNT	
S_CONTACT.PR_AFFL_ID	EXPR_S_CONTACT_PR_AFFL_ID	EIM_CONTACT	SIA-specific
S_ORG_EXT.PR_BL_PER_ID	EXPR_SIS_S_ORG_EXT_PR_BL_PER_ID	EIM_ACCNT_CUT	SIA-specific
S_ORG_EXT.PR_SHIP_PER_ID	EXPR_SIS_S_ORG_EXT_PR_SHIP_PER_ID	EIM_ACCNT_CUT	SIA-specific
S_ORG_EXT.PR_CON_ID	EXPR_S_ORG_EXT_PR_CON_ID	EIM_ACCNT_CUT	SIA-specific

Table 11. Primaries Supported by the MISC SQL Parameter

Table and Primary Child Foreign Key	MISC SQL Parameter Value for Explicit Primary	Corresponding EIM Table	Comments
S_POSTN_CON.PR_ADDR_ID	EXPR_S_POSTN_CON_PR_ADDR_ID	EIM_CONTACT1	SIA-specific
S_ORG_EXT.PR_BL_PER_ID	EXPR_FINS_S_ORG_EXT_PR_BL_PER_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_SHIP_PER_ID	EXPR_FINS_S_ORG_EXT_PR_SHIP_PER_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_CON_ID	EXPR_FINS_S_ORG_EXT_PR_CON_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_BL_OU_ID	EXPR_S_ORG_EXT_PR_BL_OU_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_SHIP_OU_ID	EXPR_S_ORG_EXT_PR_SHIP_OU_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_PAY_OU_ID	EXPR_S_ORG_EXT_PR_PAY_OU_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_COMPETITOR_ID	EXPR_S_ORG_EXT_PR_COMPETITOR_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_PRTNR_OU_ID	EXPR_S_ORG_EXT_PR_PRTNR_OU_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_EXT.PR_EMP_REL_ID	EXPR_FINS_S_ORG_EXT_PR_EMP_REL_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_BU.PR_BL_PER_ID	EXPR_S_ORG_BU_PR_BL_PER_ID	EIM_FN_ACCNT1	SIA-specific
S_ORG_BU.PR_SHIP_PER_ID	EXPR_S_ORG_BU_PR_SHIP_PER_ID	EIM_FN_ACCNT1	SIA-specific
S_CONTACT.PR_HELD_POSTN_ID	EXPR_FINS_S_CONTACT_PR_HELD_POSTN_ID	EIM_FN_CONTACT1	SIA-specific
S_ASSET.PR_ASSET_ID	EXPR_S_ASSET_PR_ASSET_ID	EIM_FN_ASSET1	SIA-specific
S_ORG_GROUP.PR_ADDR_PER_ID	EXPR_S_ORG_GROUP_PR_ADDR_PER_ID	EIM_FN_ORGGRP	SIA-specific
S_PROD_INT_TNTX.PR_CATEGORY_ID	EXPR_S_PROD_INT_TNTX_PR_CATEGORY_ID	EIM_PRDINT_TNT	SIA-specific
S_QUOTE_TNTX.PR_ORDER_ID	EXPR_S_QUOTE_TNTX_PR_ORDER_ID	EIM_QUOTE_TNT	SIA-specific

If you always want to use explicit primaries, follow this syntax:

```
MISC SQL = EXPR_S_CONTACT_PR_OU_ADDR_ID
```

If you always want to use implicit primaries, follow this syntax:

```
MISC SQL = IMPR_S_CONTACT_PR_OU_ADDR_ID
```

The most flexible method is to use explicit primaries on the records for which you have specified a primary, and to automatically use implicit primaries on the records where you have not specified a primary. The following example shows this syntax:

```
MISC SQL = EXPR_S_CONTACT_PR_OU_ADDR_ID, IMPR_S_CONTACT_PR_OU_ADDR_ID
```

For more information on how to use the MISC SQL parameter, see the sample default.ifb file located in the `Siebel Server/admin` directory.

INSERT ROWS and UPDATE ROWS Parameters

The INSERT ROWS and UPDATE ROWS parameters have optional elements of their syntax. For both parameters, the default value is TRUE. To change this for all tables, use this syntax:

```
INSERT ROWS = FALSE
```

To change only one table, specify the table name as follows:

```
UPDATE ROWS = S_CONTACT, FALSE
```

To change multiple tables, specify each table in a separate line, as follows:

```
INSERT ROWS = S_CONTACT, FALSE
INSERT ROWS = S_ADDR_ORG, FALSE
```

If you need the parameter to be FALSE for most tables, and TRUE for only a few, use this method:

```
UPDATE ROWS = FALSE
UPDATE ROWS = S_CONTACT, TRUE
UPDATE ROWS = S_ADDR_ORG, TRUE
```

Special Considerations for Imports

There are several issues you should be aware of when running import processes. These issues include the following:

Suppressing Data When Updating Existing Databases

By default, when importing information, EIM performs both inserts and updates based on the content of the batch set. However, situations may arise in which you want to perform only inserts or only updates.

Suppressing Inserts

When the batch is a superset of an existing table, you should suppress inserts. For example, you may have a batch set of employee information that includes every individual in your organization. However, your Siebel database contains only members of the sales organization. To ignore batch entries for nonsales personnel in this case, you may want to run the entire batch using this setting to perform updates to existing rows only. If EIM attempts to insert a new row with this setting, the IF_ROW_STAT column is updated to NOT_ALLOWED. This means that EIM has attempted to insert a new row, but the action is not allowed.

To suppress insertions

- Set the INSERT ROWS parameter in the EIM configuration file to FALSE.

  The following example shows how to suppress insertions of unmatched rows from the EIM_ACCOUNT table to the S_ORG_EXT base table.

```
[Import Accounts Details]

          TYPE = IMPORT

          BATCH = 1

          TABLE = EIM_ACCOUNT

          INSERT ROWS = S_ORG_EXT, FALSE
```

Suppressing Updates

When the information in your database is already accurate and current, you should suppress updates. For example, opportunities and associated contacts might appear as a batch feed from an external application on a regular basis. You may only be interested in adding new opportunities while preserving the information in existing opportunities. Use the UPDATE ROWS = FALSE statement to preserve existing information.

CAUTION: Because suppressing updates prevents updating primaries in Step 10 on page 55, this setting should be used with caution.

To suppress updates to existing rows

- Set the UPDATE ROWS parameter in the EIM configuration file to FALSE.

  The following example shows how to suppress updates to existing rows in the S_ORG_EXT base table.

```
[Import Accounts Details]

          TYPE = IMPORT

          BATCH = 1

          TABLE = S_ACCOUNT_DTLIF

          UPDATE ROWS = S_ORG_EXT, FALSE
```

Importing Customizable Products

If your data includes customizable products built in Siebel eConfigurator, you must use XML to load them. Customizable products cannot be loaded using EIM. Customizable products have rules, scripts, and resources associated with them, so in order to migrate customizable products, you must use XML import and export functionality. For information on exporting and importing products, see *Product Administration Guide*.

Importing Opportunities and Revenues

When importing opportunities and revenues, it is important to note that S_OPTY has some columns that are denormalized from S_REVN—the columns named SUM_*. These columns are not defined as type Denormalized, but nevertheless they need to be maintained as denormalized columns.

Maintaining Denormalized Columns

When updating columns that are the source of denormalized columns in other tables, you must find the records related to the columns being updated and load them as well, in the same batch.

As an example, you are updating the S_SRC table using EIM_SRC. EIM_SRC maps to S_SRC, S_SRC_BU, and S_SRC_POSTN, among others. S_SRC_BU and S_SRC_POSTN both contain the column SRC_NAME, which is denormalized from S_SRC.NAME. So, S_SRC_BU.SRC_NAME and S_SRC_POSTN.SRC_NAME should match S_SRC.NAME.

You have a record in S_SRC, and you want to update its NAME to something else using EIM_SRC. When you load the data of this record with its new NAME into EIM_SRC and then run EIM to update the NAME, EIM does not automatically update the SRC_NAME in the records within S_SRC_BU and S_SRC_POSTN. In order for the EIM engine to update S_SRC_BU.SRC_NAME and S_SRC_POSTN.SRC_NAME with these related records, you must find these related records in S_SRC_BU and S_SRC_POSTN and load them into EIM_SRC as well. The batch number must be the same. Only the user key column data needs to be loaded for these related records.

Importing Marketing Responses

In 6.x and later versions, you need to populate the CAMP_MEDIA_ID column in the S_COMMUNICATION base table with valid values from the S_SRC_DCP base table in order for the rows to be displayed in the Response views. You also need to do this if you are upgrading from version 5.x.

Importing Contacts

This topic provides information organized as follows:

- "ASGN_* Flags" on page 77
- "S_POSTN_CON.ROW_STATUS Flag" on page 77

For more information related to the importing of contacts, see:

- "Importing Private Contacts" on page 77
- "Importing Contacts to Make Them Visible in the Contact List" on page 77
- "Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts" on page 77

ASGN_* Flags

When you import contacts and set positions using EIM, the flags ASGN_MANL_FLG, ASGN_DNRM_FLG, and ASGN_SYS_FLG are set so that the intersection records are not routed to remote users. The Contacts view on the local database will display fewer contacts than the same view for the same user on the server database.

S_POSTN_CON.ROW_STATUS Flag

The column S_POSTN_CON.ROW_STATUS is a flag that can have value Y or N. When a contact is imported with value Y for this column, the contact shows in the user interface with an asterisk [*] in the New column, which means it is a new contact.

Importing Private Contacts

Siebel applications do not support importing private contacts using EIM. The default.ifb file contains a section that sets the CON_PRIV_FLG column to a constant N to make sure that only public contacts are imported. Because EIM does not support importing private contacts, do not change the value of the PRIV_FLG column. Do not remove this section of the .IFB file either—to import contacts, you must have the CON_PRIV_FLG section in the EIM configuration file.

Importing Contacts to Make Them Visible in the Contact List

You need to use EIM_CONTACT to import into S_PARTY, S_CONTACT, and S_POSTN_CON. Make sure S_POSTN_CON.POSTN_ID references valid positions and that there is at least one employee associated with each position. S_POSTN_CON.POSTN_ID is mapped by PC_POSTN_NAME, PC_POSTN_DIVN, PC_POSTN_LOC, and PC_POSTN_BU in EIM_CONTACT. PC_POSTN_BU does not map to S_POSTN.BU_ID and BU_ID is not among the user key columns of S_POSTN. Instead, PC_POSTN_BU together with PC_POSTN_DIVN and PC_POSTN_LOC are used to resolve the S_POSTN.OU_ID, which refers to the divisions the positions belong to.

Divisions are stored in S_ORG_EXT with user key columns NAME, LOC, and BU_ID. For divisions, S_ORG_EXT.BU_ID references Default Organization; therefore, PC_POSTN_BU should be populated with Default Organization.

Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts

This topic documents the causes, diagnostic steps, and solutions for troubleshooting the unique constraint error received when importing data through EIM.

NOTE: The error message and cause are the same for both contact data import and account data import, but there are separate diagnostic steps and solutions for each type of import data.

To resolve the problem, look for it in the list of Symptoms/Error Messages in Table 14.

Table 12. Resolving the Unique Constraint Error When Importing Accounts or Contacts

Symptom/Error Message	Diagnostic Steps/Cause	Solution

When importing Account or Contact data using EIM, the batch fails with the following error:
 	
`EIM-00107: ODBC (SQL) error.`
 	
The log file displays an error message similar to the following error shown below for an Oracle database:
 	
`ODBC error 23000: [MERANT][ODBC Oracle 8 driver][Oracle 8]ORA-00001: unique constraint (SIEBEL.S_CONTACT_U1) violated`
 	
`ODBC error 23000: [MERANT][ODBC Oracle 8 driver][Oracle 8]ORA-00001: unique constraint (SIEBEL.S_ORG_EXT_U1) violated`
	This unique constraint error usually occurs due to inconsistent data in the base tables or incorrect data populated in the interface tables.
	
	The inconsistent data may result when two different server tasks, such as Siebel eAI processes and an EIM process, are run at the same time to import the same data.
	
	For an example of this, see "Example of Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts" on page 186.
		See "Example of Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts" on page 186.

	Import of EIM contact data into the S_CONTACT table fails with the above error. For diagnostic steps, see "Example of Troubleshooting the Import of EIM Contact Data into the S_CONTACT Table" on page 187.	See "Example of Troubleshooting the Import of EIM Contact Data into the S_CONTACT Table" on page 187.
	Import of EIM account data into the S_ORG_EXT table. For diagnostic steps, see "Example of Troubleshooting the Import of EIM Account Data into the S_ORG_EXT Table" on page 189.	See "Example of Troubleshooting the Import of EIM Account Data into the S_ORG_EXT Table" on page 189.

Importing Party Records

There are columns in the S_PARTY table that must be populated when importing party records such as Contacts, Positions, and so on. The following are the required columns, with their possible values:

- **PARTY_TYPE_CD.** Indicates the type of party data that is being imported. The PARTY_TYPE_CD column can have the following values:

Value	Description
Person	For Contact, User, Employee, or Partner.
Organization	For Organization, Division, or Account.
Household	For a Household (or Group). A Household is comprised of a collection of Contacts, independent of Account affiliations.
Position	For an Internal Division Position.
AccessGroup (OR)	For bundling of Party entities. Relates a person to groups indirectly (through Positions, Organizations, Accounts, and so on). An Access Group can have Organizations, Accounts, Positions, and User Lists.
UserList	A User List contains Siebel persons as its members. User Lists are created on an ad-hoc basis, not restricted to the Organizations to which the persons belong or to the Positions they hold.

NOTE: No custom values are allowed in the PARTY_TYPE_CD column. This column must contain one of the values listed above.

- **PARTY_UID.** PARTY_UID is populated by default through the Siebel upgrade process and the application UI with the ROW_ID of the party record (for example, Contact or Position) that is being created, but you maintain the value for this column. The value does not have to remain identical with the ROW_ID.

  With EIM, the PARTY_UID gets populated with the value specified in the EIM table for this column. PARTY_UID may have a calculated value with logic, such as a combination of email and other data. For this reason, PARTY_UID is defined as VARCHAR100.

- **ROOT_PARTY_FLG.** ROOT_PARTY_FLG supports performance for Oracle. The following are possible queries to get top-level Positions, Organizations, or Access Groups. Try using the first query before the second one:

  - **WHERE ROOT_PARTY_FLG='Y'.** ROOT_PARTY_FLG is set to 'Y' for top-level Positions, Organizations, and Access Groups as it applies only to these party subtypes. It is set to 'N' for other party subtypes.
  - **WHERE PAR_PARTY_ID IS NULL.** Oracle cannot use an indexed access path because there are no index entries for NULL, so ROOT_PARTY_FLG was added.

NOTE: The PAR_PARTY_ID field needs to be populated only when the PARTY_TYPE_CD is set to Organization or Position. For Positions, if the record is a position that is the child of another position, then PAR_PARTY_ID needs to be populated with the ROW_ID of the parent position. In the case of Organizations, this field applies only to internal organizations. Similarly to Positions, the PAR_PARTY_ID needs to be populated with the parent organization if it has one.

Also note that Divisions and Accounts have PARTY_TYPE_CD set to Organization well, but it is not necessary to populate the PAR_PARTY_ID field.

Importing Solutions

The Solution business component has the following Search Specification property: [Solution Item='Y']. For imported records of this type to be visible following an import process, you must import data from the EIM_SOLUTION interface table to the S_RESITEM base table with the value in the SOLUTION_ITEM_FLG column set to 'Y.'

When importing into the S_RESITEM base table, you need to include the following columns in the ONLY BASE COLUMNS parameter in the EIM configuration file:

- FILE_NAME
- FILE_EXT
- FILE_SRC_TYPE

If these columns are not included in the ONLY BASE COLUMNS parameter, a low-level error will be generated.

Another requirement is that the Internal Publish flag (INTR_PUBLISH_FLG) must be set in the parent record for imports to be visible in the Solution/Resolution Documents view.

Importing Call Lists

When importing into the S_CALL_LST base table, you need to include the following columns in the ONLY BASE COLUMNS parameter in the EIM configuration file:

- FILE_NAME
- FILE_EXT
- FILE_SRC_TYPE

If these columns are not included in the ONLY BASE COLUMNS parameter, a low-level error will be generated.

Importing Positions and Employees

The Administration - Group views automatically maintain the internal organization hierarchy incrementally as you change your organization's position hierarchy, minimizing transaction volume and therefore improving the performance of Siebel Remote. For more information on using the Administration - Group views for working with positions, see *Security Guide for Siebel eBusiness Applications*.

When using EIM to import or update positions, you must generate reporting relationships after running EIM to maintain organization relationships. If you do not generate reporting relationships, then incomplete or inaccurate data will be displayed in views involving employees or positions. For example, the My Team View will fail to display all positions on the team.

NOTE: When importing or updating positions, you must check for duplicate reporting relationships. Make sure that no positions report directly to themselves (PAR_POSTN_ID=ROW_ID). Before importing, search for this condition and correct it. If you import a record with this condition, you will get an error when you click Generate Reporting Relationships after the import.

To activate position hierarchy, see "Activating Position Hierarchy" on page 82. To generate reporting relationships, see "Generating Reporting Relationships" on page 83.

NOTE: EIM does not support importing Multiple Organization Visibility organizations. You cannot import this type of organization using the EIM_ORG_INT interface table or S_ORG_INT base table. EIM does support importing divisions that are not Multiple Organization Visibility Organizations.

To import employees and positions

1. Before importing employees and positions, make sure that the Position and Department columns in the Employee table contain the correct data, as follows:
   - Data from the Hire Date column in the Employee table matches the data from the Emp_Start_Date column in the Position table.
   - Data from the Position Start Date column in the Employee table matches the data from the Position Start Date column in the Position table.
   - Position table contains the logons of all employees.
   - Data from the Employee Hire Date column in the Position table matches the data from the Hire Date column in the Employee table.
2. Import the Employee table.

   You should import the Employee table first, because EIM searches for the foreign key of the Position table during its import and update of the Employee table.

   NOTE: If you are importing employees and positions with S_CONTACT.PR_HELD_POSTN_ID and S_POSTN.PR_EMP_ID set as primary columns, import the Position table first. See "To import employees and positions with S_CONTACT.PR_HELD_POSTN_ID and S_POSTN.PR_EMP_ID as primary columns" on page 82.
3. Import the Position table.

   If you want to import employees and positions using EIM and you also want to set the following primary columns:
   - S_CONTACT.PR_HELD_POSTN_ID
   - S_POSTN.PR_EMP_ID

   Then you will have to run the import twice for the EIM_POSITIONS table.

To import employees and positions with S_CONTACT.PR_HELD_POSTN_ID and S_POSTN.PR_EMP_ID as primary columns

1. Import the Position table using the EIM_POSITION interface table.
2. Import the Employee table, associate positions, and set the primary held position (S_CONTACT.PR_HELD_POSTN_ID) with the use of the MISC SQL parameter.
3. Set the primary employee of Position (S_POSTN.PR_EMP_ID) by using the EIM_POSITION table and the MISC SQL parameter.

Activating Position Hierarchy

After importing or merging positions using EIM, or after merging positions through the user interface, it is necessary to generate reporting relationships to populate or rebuild S_POSTN_RPT_REL (for versions prior to 6.x) or S_PARTY_PER (for version 7.x or later). This happens automatically when you insert positions using the user interface.

NOTE: For customers using the Siebel Financial Services application, the relationship of party entities is stored in S_PARTY_RPT_REL.

From Siebel 98 (Version 4) and later, the Generate Reporting Relationships button is no longer exposed by default. To expose this button, follow the instructions in "Exposing the Generate Reporting Relationships Button for Versions Prior to 6.x" on page 82 or "Exposing the Generate Reporting Relationships Button for Versions 7.x and Later" on page 83, depending on the version number of the application you are using.

Exposing the Generate Reporting Relationships Button for Versions Prior to 6.x

In Siebel Tools, there are two places where you can expose the Generate Reporting Relationships button:

- View=Internal Division, Project=Division
- View=Organization Chart, Project=OrgChart

To expose the Generate Reporting Relationships button

1. Log in to Siebel Tools.
2. Open the Siebel repository.
3. Select the View QBE.
4. Select Internal Division or Organization Chart (depending on which place you chose to expose this button).
5. Lock the project.
6. Populate sectors 6 and 7 with Position List Applet (for Internal Division) or sectors 4, 5, 6, and 7 (if you chose Organization Chart).
7. Compile the locked project and distribute new SRF files to users who need to perform this function.

After exposing the Generate Reporting Relationships button, you can test it by generating reporting relationships. See "Generating Reporting Relationships" on page 83.

Exposing the Generate Reporting Relationships Button for Versions 7.x and Later

The Generate Reporting Relationships process needs to be executed after upgrading to version 7.0. For more information, see the section on post-upgrade tasks for the production environment in the *Upgrade Guide* for the operating system you are using. You also need to execute this process whenever the denormalized hierarchy structure (S_PARTY_RPT_REL) becomes unsynchronized with the data in the normalized tables (S_PARTY).

The following situations can cause these tables to become unsynchronized:

- After upgrading to version 7.0, the organizational hierarchy (even if there is only one organization) must be established to maintain appropriate visibility in the views mentioned previously.
- When you use EIM to import or update any of the hierarchies (positions, organizations, or access groups).

Generating Reporting Relationships

If you want to modify your organization structure by importing or updating positions using EIM, you must generate reporting relationships after running EIM to maintain organization relationships. Before generating reporting relationships, you must first activate position hierarchy by completing the procedure in "Activating Position Hierarchy" on page 82.

For best performance, complete all organization changes before generating reporting relationships, because this operation generates a high number of transactions for mobile users. This operation generates reporting relationships for all organizations and divisions regardless of the organization or division you have selected in the GUI. For more information on organization administration, see *Security Guide for Siebel eBusiness Applications*.

NOTE: If you have mobile users, stop the Transaction Processor before clicking Generate Reporting Relationships. This is necessary because generating the reporting relationships can cause a large number of Siebel Remote transactions to also be generated.

To generate reporting relationships

1. Navigate to Administration - Group > Positions.
2. In the Positions list applet, click Generate Reporting Relationships.
3. Click OK.

Importing Data with Parent and Child Relationships

Siebel applications support multilevel hierarchies for defining accounts, products, and product lines. For example, a product's bill of materials may involve levels for components, assemblies, and sub-assemblies. Similarly, a parent account may have multiple child accounts for company divisions and wholly-owned subsidiaries. These child accounts may be further organized into subaccounts such as regions and offices.

Siebel applications support an unlimited number of levels within account, product, and product line structures. For a child entity to be successfully imported, its parent must first be successfully imported in a prior batch or in the same batch.For more information, see "Example of Importing and Exporting Hierarchical LOVs" on page 191.

Importing Industry Codes

Siebel applications support the use of Standard Industrial Classification (SIC) codes. For example, a company may want to categorize its customers by industry type using SIC codes. In Siebel applications, the SIC field holds values that map to specific industries. If you want to use SIC codes, you can import data from a third-party database that supports SIC codes using EIM.

NOTE: SIC codes are valid only for the United States and Canada. If you want to implement industry codes for other countries, you need to create custom industry codes for your company and map these codes accordingly in EIM.

Importing File Attachments

EIM can import file attachments in all formats, including common file types such as Word documents (.doc), Excel spreadsheets (.xls), and text files (.txt). EIM does not place a limit on the number or the total size of files that can be imported.

To import file attachments into Siebel database tables

1. Using Windows Explorer, navigate to the Siebel Server directory.

   The default is c:\siebel.

2. Verify that the Siebel directory contains a directory named input.

   If the directory does not exist, create it by choosing File > New > Folder and entering `input`.

3. Copy all file attachments to the input directory.

   Siebel EIM tables support all file attachment formats.

4. Populate EIM tables with rows matching the file attachments.

5 Run EIM.

NOTE: All three file attachment columns (FILE_NAME, FILE_EXT, FILE_SRC_TYPE) must be populated in order to import file attachments. The FILE_SRC_TYPE column must be set to FILE. Although these columns can be listed as nullable in the EIM tables, the import process will return errors if you leave any of these columns as NULL.

Updating File Attachments

You can also update file attachments that have already been imported into the Siebel database.

In order to update file attachments, EIM deletes the old row pointing to the existing file attachment and then imports the new file attachment. After all file attachments have been updated, use the Siebel File System Maintenance Utility named sfscleanup.exe (during hours when the network is least laden) to clean the file attachment directory of any unused file attachments.

To update file attachments

1 Update the file attachment by completing the steps in "Importing File Attachments" on page 84.

2 Once all file attachments have been updated, run the Siebel File System Maintenance Utility named sfscleanup.exe to clean up the file attachment directory.

   For information on using sfscleanup.exe, see *Siebel System Administration Guide*.

NOTE: EIM does not support merging of file attachments.

Importing Organizations That Contain the BU_ID Column

Base tables in the Siebel Data Model that are enabled for multiple organizations contain the BU_ID foreign key column. This column points to a business organization defined in the S_BU base table. Examples of such base tables include S_PROD_INT, S_PRI_LST, and S_DOC_AGREE.

NOTE: For more information on multi-org, see the section on access control in *Security Guide for Siebel eBusiness Applications*.

During the import process, if the value supplied in the EIM table does not resolve to a valid business organization, EIM by default will continue to import the record with the BU_ID set to the default value defined in the base table. If you want EIM to report import failures for such instances, set the parameter SKIP BU_ID DEFAULT parameter to TRUE in the .IFB file (the default value for this parameter is FALSE).

If you have not implemented multi-org capability or if you will not be using organizations, then use the Default Organization, a predefined organization in the S_BU base table.

Importing Accounts Containing Multiple Team Members

You can import multiple team members for accounts using EIM_ACCOUNT. Accounts and team members are related through S_ACCNT_POSTN. You can import multiple team members for accounts at the same time and specify the primary positions by setting ACC_PR_POSTN to Y.

Importing Multiline Fields

When importing multiline fields, such as addresses, you should use CHR(13) and CHR(10) for the field to be displayed as a multiline field. Otherwise, the following warning may be displayed in the GUI:

```
You have tried to modify a group of fields that may have more than one value. To
edit or add field values in this group, please open the first field in the group by
clicking on the multivalue field control.
```

Importing Exported Rows Into Target and Secondary Tables

If user keys from the secondary tables are made up of foreign keys referencing the target table and additional user keys of nonrequired columns, note that:

- If you export rows from both target and secondary base tables, one EIM table row will be created for every target table row, and a separate EIM table row will be created for every related secondary table row.
- If you reimport the exported batch rows into both the target and secondary base tables, the exported target table rows will be imported into the secondary table as well. This is because the exported target table rows have NULL values in the secondary table interface columns, and the secondary table's additional user keys allow NULL values to be imported. Additional rows will thus be mistakenly imported into the secondary base table.

To avoid this problem, after exporting the target and secondary base tables rows, you should split the secondary table rows out from the exported batch into another batch, and then import the target and secondary table rows separately.

Importing International Phone Numbers Using EIM

To import international phone numbers, the phone number must be prefixed with a plus (+) sign and the country code. For example, an international phone number with a country code of 44 should have the following format: +44123456789.

Any phone number without a preceding plus sign in the database is treated as a US phone number. This leads to the display of +1 in front of the phone number, and the use of the corresponding PHONE_FORMAT if the regional settings of the client are different.

Importing URL Links Into the S_LIT Base Table

To import records as URL links into the S_LIT base table, the FILE_NAME column must not be NULL and the FILE_EXT column must be NULL for URLs.

Importing LOV and MLOV Data

When importing List of Values (LOV) data, whether into an LOV column or a multilingual LOV (MLOV) column, you must populate the EIM table column with the display value of a specific language. The difference between the two cases is the following:

- When importing into an LOV column, the EIM engine puts the display value directly into the column.
- When importing into an MLOV column, EIM translates MLOV values during the import process. The EIM engine looks up the Language Independent Code (LIC) of the display value in the EIM table column and populates the LIC into the MLOV column.

EIM runs in the same language as that of the Siebel server installation. For example, if the Siebel server installation is in German, the LANGUAGE parameter setting defaults to German. In this example, the following takes place:

- To import into an MLOV column, you enter a German display value in the EIM table column. You can enter “Aktiv” to indicate an account status that is active. The EIM engine puts the corresponding LIC, "Active," into the MLOV column.
- To import into an LOV column, the EIM engine puts "Activ" into the LOV column.

NOTE: You must always populate EIM table columns that are mapped to LOV bounded base table columns with values that correspond to S_LST_OF_VAL.VAL, even when MLOV are used.

To find the specific steps for importing LOV data, see the example in “To import data into an LOV table” on page 88.

LOV Validation

When importing data from EIM tables, you may encounter the following error message in your trace file:

```
[ERR00] Interface table:
[ERR00] S_XXXX_XMIF (Interface for XXXX Built-In M:1 Extension Table)
[ERR00] --------------
[ERR00]
[ERR00] Base table:
[ERR00] S_XXXX_XM (Account M:1 Extension)
[ERR00] ------------
[ERR00] TYPE (Type)
[ERR00] This column contains a bounded picklist value and the value given does not
[ERR00] correspond to a value in the list-of-values table for the given picklist
type.
```

This error message indicates that either a picklist has not been created for this column (TYPE) or the value in your EIM table for this column (TYPE) does not correspond to one of the values in the picklist for this column. To resolve this issue, you need to make sure that:

- A picklist already exists for this column.
- The value you are importing for this column corresponds to one of the values in the picklist.

The following procedure explains how to import data into an LOV table, using the S_ORG_EXT_XM table as an example.

To import data into an LOV table

1. To find the LOV type for a column in the S_ORG_EXT_XM TABLE, perform the following actions:
   a. In Siebel Tools, select Types.
   b. Click Table.
   c. Select S_ORG_EXT_XM.
   d. With the S_ORG_EXT_XM table highlighted, expand Column tree control, and find the Type column.
   e. With the Type column highlighted, find the following two attributes in the Properties window:
      - Lov Bounded: TRUE
      - Lov Type: ORG_EXT_XM_TYPE

      The TYPE column should contain the value as the VAL column in the S_LST_OF_VAL table.
2. Using the Siebel client, find S_ORG_EXT_XM_TYPE.
   a. Navigate to the List of Values screen.
   b. Query the Display Value column for ORG_EXT_XM_TYPE to make sure that the picklist already exists.
3. Using the Siebel client or EIM, add values for this bounded picklist.

   If you are using the Siebel client:

   a. In the List of Values view, create a new record.
   b. In the Type column, type ORG_EXT_XM_TYPE.
   c. In the Display value column, insert any value you want to use for this type.
   d. Repeat Step c until you have created records for all values you want to have in this picklist.

   If you are using EIM:

   e. Populate the EIM_LST_OF_VAL table, set the TYPE column to ORG_EXT_XM_TYPE, and set the VAL column to any value you want to use for this type. Make sure to populate all the required fields in the EIM_LST_OF_VAL table.
   f. Repeat Step e until you have inserted all records into the table for all values you want to have in this picklist.

g Import data from EIM_LST_OF_VAL to S_LST_OF_VAL using EIM.

The VAL column in the S_LST_OF_VAL table should contain the same value as the TYPE column in the S_ORG_EXT_XM table.

EIM and Audit Trail

EIM is a tool for performing bulk updates to data. EIM runs directly on the Siebel database object layer, so it has no direct knowledge of Siebel business objects or UI objects. Because Siebel Audit Trail functions on the business object layer, EIM does not record audit trail information, even if the data being updated is owned by business components that have auditing switched on.

The bulk updates that EIM performs are directly controlled by the .IFB file and the population of the EIM table in use. For this reason, the .IFB file and the EIM table provide information that you can use for audit trail purposes. You can also use the EIM log file for audit trail purposes, because it shows how many records have been manipulated.

If the use of Audit Trail is a requirement in your Siebel implementation, use Siebel Business Process Designer to design batch updates. These batch updates will then operate on the business component layer, so they will update the audit trail.

Running an Import Process

You can run an import process when you have:

- Identified the data for import processing
- Prepared the related EIM tables
- Modified the EIM configuration file accordingly

Run the import process by completing the procedures in Chapter 9, "Running EIM."

Checking Import Results

When an import process ends, you should carefully check the results to verify that data was successfully imported. During each import process, EIM writes comprehensive status and diagnostic information to multiple destinations. This section explains how to use this information to determine the results of the import process and is organized as follows:

Viewing a List of Imported Rows

The first task you should perform to check the results of the import process is to view a list of the imported rows.

To view a list of imported rows

- Query the appropriate EIM tables for rows whose IF_ROW_BATCH_NUM equals the batch number for the import.

These columns in each EIM table indicate whether a row was imported successfully, and they identify the pass number on which a row failed. During various passes of import processing, EIM sets the IF_ROW_STAT value to one of the values shown in Table 13 on page 90.

If error flags, SQL trace flags, or trace flags were activated for the EIM process, you can also use the trace file to view the results of the EIM process. For more information on viewing the trace file, see "Viewing the EIM Log File" on page 128.

Table 13. IF_ROW_STAT Values

Value	Comment
AMBIGUOUS	There are two rows in the base table that have the same user key but different conflict IDs. EIM cannot distinguish these rows.
DUP_RECORD_EXISTS	The row exactly matches rows that already exist in the destination tables. This error occurs in Step 8 on page 54. Note that a row may have a duplicate in the target base table, but not in other destination base tables. In this situation, EIM adds the new relation (a child or intersection table) in the other destination base tables, and does not mark the EIM table row as a duplicate.
DUP_RECORD_IN_EIM_TBL	The row was eliminated because it is a duplicate (has the same user key) of another row in the EIM table with the same batch number. In this case, MIN(ROW_ID) is the record processed, and the other records with the same user key are marked as DUP_RECORD_IN_EIM_TBL. Do not confuse DUP_RECORD_IN_EIM_TBL with DUP_RECORD_EXISTS. DUP_RECORD_EXISTS status indicates that the same record already exists in the base table, while DUP_RECORD_IN_EIM_TBL status indicates that there are two or more EIM table records having the same user key values.
FOREIGN_KEY	A required foreign key column in the target table could not be resolved. This error occurs in Step 4 on page 54.
IMPORTED	The row was successfully processed against all its destination base tables. This status is set after the import has been completed. You can check the import status by using database commands to query the appropriate EIM tables for rows whose IF_ROW_STAT value is not equal to IMPORTED. The result is a list of rows that were not successfully imported.
IMPORT_REJECTED	A user-specified filter query failed for this row. This error occurs in Step 3 on page 54 if the user has specified FILTER QUERY expressions.
IN_PROGRESS	In Step 1 on page 53, EIM sets IF_ROW_STAT to this initial value for all rows in the batch. If rows still have this status value after EIM exits, a failure occurred that aborted processing for this table.

Table 13. IF_ROW_STAT Values

Value	Comment
NON_UNIQUE_UKEYS	The user key was not unique in all the user key specifications on the table.
PARTIALLY_IMPORTED	The row did not fail for the target table (although it may have been a duplicate), but did fail during processing of a secondary base table. This status is set after the import has completed.
PICKLIST_VALUE	A required picklist value in the target table could not be resolved. This error occurs for NULL or invalid bounded picklist values in Step 4 on page 54.
REQUIRED_COLS	One or more required columns for the target table were NULL. This error occurs for missing user key columns in Step 7 on page 54, or when inserting new rows in Step 9 on page 54.
ROLLBACK	EIM encountered an error, such as an SQL database failure, and rolled back the transaction. This status is only used when ROLLBACK ON ERROR = TRUE.
SQL_ERROR	An SQL error occurred during an attempt to import this row. This error occurs for rows processed when Docking: Transaction Logging is set to TRUE.

Troubleshooting Import Processing Failures

EIM is designed to import large volumes of data. Most failures are caused by data errors. It is usually faster and easier to correct the data errors and resubmit the corrected rows as part of a subsequent batch than to reprocess an entire batch. EIM does not stop when failures occur.

Failures can occur at several steps during the "EIM Import Process" on page 53; each type of failure has a different cause. See the causes listed in Table 14 on page 92.

This section provides guidelines for resolving import processing problems. To resolve the problem, look for it in the list of Symptoms/Error Messages in Table 14.

Table 14. Resolving Import Processing Problems

Symptom/Error Message	Diagnostic Steps/Cause	Solution
Step 4 Failures	Step 4 processes foreign keys and bounded picklists. A row fails this step if the foreign key developed from values in the EIM table columns does not correspond to an existing row in the target Siebel database table. For example, a Step 4 failure on ACCNT_NAME indicates that the value in the ACCNT_NAME column of that row did not correspond to an existing name (S_ORG_EXT.NAME) or synonym name (S_ORG_SYN.NAME).	Correct the data errors and resubmit the corrected rows as part of a subsequent batch.
Step 6 Failures	Step 6 failures generally indicate invalid user key values. For example, a contact with a NULL value for the LAST_NAME column will fail because this is a required user key. All user keys are required except MID_NAME for contacts (S_CONTACT.MID_NAME) and LOC (location) for accounts (S_ORG_EXT.LOC).	Correct the data errors and resubmit the corrected rows as part of a subsequent batch.
Step 7 Failures	Step 7 evaluates the foreign key relative to the data being imported (whereas Step 4 evaluates it relative to existing data). If the foreign key references a table that is imported from the same EIM table, Step 7 resolves foreign keys into the data to be imported.	Correct the data errors and resubmit the corrected rows as part of a subsequent batch.
Step 8 and Step 9 Failures	Failures for Step 8 and Step 9 indicate columns that have NULL values for fields that are required but are not part of the user key.	Correct the data errors and resubmit the corrected rows as part of a subsequent batch.

Table 14. Resolving Import Processing Problems

Symptom/Error Message	Diagnostic Steps/Cause	Solution
Data not visible after import	If you find that, after an EIM import, the data is not visible in some views or applets, it is probably because values required for a particular view or applet to display imported data may not have been imported. For example, the Sales Order Line Items applet's product picklist will only display products with S_PROD_INT.SALES_SRVC_FLG value set to N.	To determine which values need to be imported for a particular view or applet, do a client-side spooling and check the SQL conditions when selecting the record.
Unable to edit quotes after import	Users are unable to edit their quotes after importing quote information.	Make sure that the APPROVED_FLG field is set to N or left blank for each quote. Setting APPROVED_FLG to Y makes the quote read only and not editable by the user.

6 Exporting Data

This chapter explains how to export data from the Siebel base tables into the EIM tables. This chapter is organized into the following sections:

Overview of EIM Export Processing

To export data, EIM reads the data in the Siebel database tables and places the information in the appropriate EIM tables. You can then copy data from the EIM tables into another database. The export process generally populates the applicable EIM table with a row for every Siebel base table row encountered. As a consequence, where EIM tables have mappings to multiple Siebel base tables, one export operation can generate multiple rows within the EIM table governing the rows encountered within the Siebel base tables.

During its multiple passes through the EIM tables, EIM performs the following tasks:

- EIM initializes the EIM tables for export.
- It applies filter logic to select rows for exporting.
- EIM updates EIM table rows to indicate the export status.

EIM then provides comprehensive status information about each export process. When the process ends, you should review this information. See "EIM Export Process" on page 96 for more details on how EIM functions in the export process.

The following tasks comprise an EIM export process:

- "Preparing the EIM Tables for Export Processing" on page 96
- "Editing the Configuration File for Export Processing" on page 97
- "Running an Export Process" on page 102
- "Checking Export Results" on page 102
- "Extracting Data from the EIM Tables" on page 103

Upon completion of the EIM process, your database administrator can access the EIM tables and extract the data for use in a non-Siebel application.

EIM Export Process

To export tables of data, EIM performs a sequence of tasks. Each task involves multiple passes; at least one pass is required for each EIM table included in the process.

To export data to EIM tables, EIM performs the following steps:

1. EIM initializes EIM tables for export.

   If CLEAR INTERFACE TABLE in the configuration file is TRUE, all rows with the specified batch number are deleted. Otherwise, a warning is issued if rows already exist with the specified batch number. The default configuration file is default.ifb.

2. It uses export parameter expressions in the configuration file to locate and export table rows:

   - If EXPORT ALL ROWS is TRUE, ignore any EXPORT MATCHES parameters and export all rows.
   - If EXPORT ALL ROWS is FALSE, use EXPORT MATCHES parameters to locate specific rows.

   Set IF_ROW_STAT to EXPORTED for rows that are successfully exported.

3. For parent tables, EIM locates child table rows and exports them to their corresponding EIM tables.

NOTE: Transaction logging does not occur during export operations because Siebel base table values are not modified.

Preparing the EIM Tables for Export Processing

Unlike other Open Interfaces processes, an export process requires minimal preparation of the EIM tables. During the first step of export processing, EIM inspects each EIM table involved in the process. If EIM finds a row whose IF_ROW_BATCH_NUM matches the batch number for this export process, it does one of the following:

- Clear the row if the CLEAR INTERFACE TABLES parameter is set to TRUE in the EIM configuration file
- Issue a warning if the CLEAR INTERFACE TABLES parameter is set to FALSE in the EIM configuration file

For information on the CLEAR INTERFACE TABLES parameter, see "Parameters Used for Exports in Both the Header and Process Sections" on page 99.

Checking Existing Rows Batch Numbers

Before you initiate an export process, you should verify that rows do not contain an IF_ROW_BATCH_NUM matching the batch number you plan to use. If such rows do exist, you should either make sure that they do not contain data you need to preserve, or change the batch number for the export process. In each row that you are exporting, you may also want to set the IF_ROW_STAT column to FOR_EXPORT.

Preserved Column Values

The values for the LAST_UPD and CREATED columns in the EIM tables always contain the values for the LAST_UPD and CREATED columns from the target base table. For example, if you use the EIM_CONTACT interface table to export data from the S_CONTACT and S_ADDR_PER base tables, the values of the EIM_CONTACT.LAST_UPD and EIM_CONTACT.CREATED columns contain the data from the S_CONTACT.LAST_UPD and S_CONTACT.CREATED columns, respectively.

EIM Tables Not Supported for Export Processes

Due to the complexity of the associated base tables, EIM export processes to the following interface tables are not supported:

- EIM_ACCSRCPIDTL
- EIM_CRSE_TSTRUN
- EIM_IC_CALC
- EIM_IC_PERF_HST
- EIM_MDF

For more information on special columns, see "EIM Table Columns" on page 20. For general information on EIM tables, see Chapter 3, "Siebel EIM Tables."

Editing the Configuration File for Export Processing

This section describes the header and process sections that you need in the EIM configuration file to properly configure EIM for an export process. For general information on the EIM configuration file, see Chapter 4, "EIM Configuration File."

Before export processing begins, you must change the configuration file to support this function. Such changes include:

- Editing the configuration file header and process sections using the parameters specific to export processes. For general information on the EIM configuration file, see Chapter 4, "EIM Configuration File."
- Altering configuration file settings for the following purposes:
  - "Exporting All Data Rows" on page 101
  - "Exporting Selected Data Rows" on page 101
  - "Exporting Recursive Relationships" on page 101
  - "Exporting LOV and MLOV Data" on page 101

Header Section Parameters Used for Exports

Parameters in the header section generally apply to all types of processes. For a description of the necessary contents in the header section, see "Header Section Parameters Generic to All EIM Processes" on page 37.

Process Section Parameters Used for Exports

Parameters in the process section apply only to that specific process and override any corresponding value in the header section for the specific process. This section describes the parameters used in the process section that are specific to an export process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

To export data, you must define at least one process with TYPE = EXPORT. The following example contains lines that may be used in the EIM configuration file to define an export process from the S_PARTY table and its extension tables.

```
[Export Accounts]
  TYPE = EXPORT
  BATCH = 2
  TABLE = EIM_ACCOUNT
  EXPORT ALL ROWS = TRUE
```

NOTE: For performance reasons, you should limit the number of tables to export in a single process section to five or less.

Parameters Used for Exports in Both the Header and Process Sections

Table 15 describes the parameters that can appear in either the header section or a process section, and are specific to an export process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

Table 15. Export Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
ATTACHMENT DIRECTORY	(Default is SIEBEL_HOME\OUTPUT) Specifies the directory to be used for exporting attachments. Before specifying a directory, make sure the directory exists on a Siebel Server machine and you have read and write access to it. Example: `ATTACHMENT DIRECTORY = SIEBEL_HOME\OUTPUT` If the export of an attachment fails, the export process continues and EIM writes a message in the trace file.
CLEAR INTERFACE TABLE	Specifies whether existing rows in the EIM table for the given batch number should be deleted. The default value is TRUE.
EXPORT ALL ROWS	Specifies that all rows in the target base table and secondary tables are to be exported. The default value is FALSE. Existing values in the EIM table and export matches expressions are ignored. For all columns to export using an EIM table (both data from the base table and data from related child tables), you need to make sure this parameter is set to TRUE (you may need to add this line if it does not currently exist) in the .IFB file. Note: Rows from child tables of related child tables are not exported until they have been mapped.
EXPORT MATCHES	WHERE clause fragment. Example: `EXPORT MATCHES = (NAME LIKE "GEN%")` For more information on the EXPORT MATCHES parameter, see "EXPORT MATCHES Parameter" on page 99.

EXPORT MATCHES Parameter

The EXPORT MATCHES parameter specifies a WHERE clause expression for filtering base table rows. The value is in two parts: the Siebel EIM table name and the filter expression that goes against the target base table. The expression is applied against the target base table for the EIM table.

The expression is a self-contained WHERE clause expression (without the WHERE) and should use only literal values or unqualified column names from the base table. There must also be a space separating the operator from the operand.

NOTE: Complex SQL WHERE clauses like subqueries are not supported.

EXPORT MATCHES can be used only against a target base table, or against a non-target base table that is an extension table of S_PARTY when the target table is S_PARTY. For more information, see "To check whether a base table is of Siebel extension type" on page 100.

The syntax to use with the EXPORT MATCHES parameter depends on whether the target base table is S_PARTY or not.

NOTE: The column names included in the criteria (that is, in "(...criteria...)" below) must be columns from the target base table or the table that is specified for the EXPORT MATCHES parameter.

Syntax for EXPORT MATCHES with S_PARTY as the Target Base Table

The syntax listed below is for use with the EXPORT MATCHES parameter if the EIM table's target table is S_PARTY.

Allowed syntax includes the following:

```
EXPORT MATCHES = S_PARTY, (...criteria...)

EXPORT MATCHES = <non-target base table name of Siebel Extension type>,
(...criteria...)
```

> **NOTE:** When using the EXPORT MATCHES parameter against a non-target base table, you must still include the target table in the export.

The following syntax is *not* allowed:

```
EXPORT MATCHES = <EIM table name>, (...criteria...)

EXPORT MATCHES = (...criteria...)
```

Syntax for EXPORT MATCHES with Target Base Tables Other Than S_PARTY

The syntax listed below is for use with the EXPORT MATCHES parameter if the EIM table's target table is not S_PARTY.

Allowed syntax includes the following:

```
EXPORT MATCHES = <EIM table name>, (...criteria...)

EXPORT MATCHES = <target base table name>, (...criteria...)

EXPORT MATCHES = (...criteria...)
```

The following syntax is *not* allowed:

```
EXPORT MATCHES = <non-target base table name>, (...criteria...)
```

To check whether a base table is of Siebel extension type

1 In Siebel Tools, navigate to the Table control and query a table name.

Check the Type property value. If the Type property value contains 'Extension (Siebel),' then the table is a Siebel extension type table.

Exporting All Data Rows

To export all rows from the tables that are mapped in an EIM table, set the EXPORT ALL ROWS parameter for the file to TRUE in the specific export batch section of the EIM configuration file. The following example contains lines that may be used in the EIM configuration file to export all data rows from the accounts table.

```
[Export Accounts]
  TYPE = EXPORT
  BATCH = 2
  TABLE = EIM_ACCOUNT
  EXPORT ALL ROWS = TRUE
```

Prior to exporting, make sure that your database administrator has allocated enough space for the EIM table into which data will be exported.

Exporting Selected Data Rows

To export selected rows from base tables, set the EXPORT ALL ROWS parameter as follows:

```
EXPORT ALL ROWS = FALSE
```

Specify one or more EXPORT MATCHES expressions to define the rows you want exported in this batch.

Exporting Recursive Relationships

Siebel applications support multilevel hierarchies for defining accounts, products, and product lines. For example, a product's bill of materials may involve levels for components, assemblies, and sub-assemblies. Similarly, a parent account may have multiple child accounts for company divisions and wholly owned subsidiaries. These child accounts may be further organized into subaccounts such as regions and offices. Siebel applications support an unlimited number of levels within account, product, and product line structures.

Exporting LOV and MLOV Data

When exporting List of Values (LOV) data, whether from an LOV column or a multilingual LOV (MLOV) column, the EIM engine populates the EIM table column with the display value of a specific language. The difference between the two cases is the following:

- When exporting from an LOV column, the EIM engine exports the display value stored in the column.

- When exporting from an MLOV column, EIM translates MLOV values during the export process. You do not need to populate the EIM table columns prior to the export. The EIM engine looks up the language-specific display value for the Language Independent Code (LIC) stored in the MLOV column, and puts the display value in the EIM table column.

  NOTE: If you are exporting from an MLOV, you must set the LIC parameter to the appropriate language first. EIM exports the display value for the language specified.

For more information on how EIM processes LOV and MLOV data, see "Importing LOV and MLOV Data" on page 87.

Running an Export Process

You may run an export process once you have:

- Identified the data for export processing
- Prepared the related EIM tables
- Modified the EIM configuration file accordingly

Run the export process by completing the procedures in Chapter 9, "Running EIM."

If you are exporting data that pertains to organizations and divisions, it may be necessary to run additional SQL statements against the EIM table to complete the export of names from the S_BU base table (used for organizations).

To populate the BU columns from the S_BU base table

1. In the Admin directory within the Siebel Server root directory, open the file named eim_export_lookup_bu_name.sql.
2. Locate the appropriate SQL statement for the base table that you are exporting.
3. Modify this SQL statement if necessary and run it against the EIM table to populate the BU columns from the S_BU base table.

Checking Export Results

When an export process ends, you should carefully check the results to verify that data was successfully exported. During each export process, EIM writes comprehensive status and diagnostic information to several destinations.

Viewing a List of Exported Rows

You can verify export results by checking a list of exported rows, as described in the following procedure.

To view a list of exported rows

- Query the appropriate EIM tables for rows whose IF_ROW_BATCH_NUM equals the batch number for the export.

  The value of IF_ROW_STAT should be EXPORTED.

If error flags, SQL trace flags, or trace flags were activated for the EIM process, you can also use the trace file to view the results of the EIM process. For more information on viewing the trace file, see "Viewing the EIM Log File" on page 128.

Extracting Data from the EIM Tables

Upon completion of an export process, the database administrator can use appropriate tools (such as native SQL) to extract data from the EIM tables for subsequent use by an external application. The following examples illustrate when to perform this process:

- If you have exported employee information for transfer to a human resources application.
- If you want to load customer information for a specific accounting application. Begin by exporting your customer information from the Siebel database.

7 Deleting Data

This chapter covers the process of deleting selected data from the Siebel database. This chapter is organized into the following sections:

EIM Delete Process

EIM reads information from the EIM tables and the EIM configuration file to identify rows to delete from the Siebel base tables.

During its multiple passes through the EIM tables, EIM performs the following tasks:

- EIM initializes the EIM tables for deletion.
- It applies filter logic to do one of the following:
  - Select rows for deleting
  - Insert EIM tables rows that correspond to matching base table rows
  - Select rows with matching user keys in the EIM tables
- EIM updates other tables with rows containing foreign keys that point to newly deleted rows.

EIM provides comprehensive status information about each delete process. When the process ends, you should review this information. For further details, see "EIM Delete Process" on page 105.

The EIM delete function requires you to perform the following tasks:

- "Preparing the EIM Tables for Delete Processing" on page 107
- "Editing the Configuration File for Delete Processing" on page 108
- "Running a Delete Process" on page 116
- "Checking Delete Results" on page 116

The delete process performed by EIM is called a *cascade delete.* When a cascade delete is performed, all of the contents of a data structure, including all of its substructures, are deleted. In other words, the data deleted is not restricted to the base tables mapped to the EIM table that you specified in the delete process, but all child records as well. To delete data, EIM performs a sequence of tasks. Each task involves multiple passes; at least one pass is required for each EIM table included in the process. You should be very careful and specific when specifying delete criteria. For example, using the criteria "DELETE MATCHES = S_PARTY, (CREATED > *xxxxx*)" causes all records of S_PARTY that match this criteria to be deleted from the database.

Deletion Methods Supported

EIM uses a combination of EIM table row contents and configuration file parameter values to determine the method for selecting rows to be deleted. The following methods are supported:

- Delete rows in a Siebel base table with user key values specified in the corresponding EIM table.
- Delete rows in the base table where the contents of a named column match those specified by a WHERE clause expression in the configuration file.
- Delete all rows in the base table regardless of EIM table row contents or configuration file WHERE clause expressions.

  CAUTION: Do not use EIM to delete organizations. Using EIM to delete data from the Products base tables is also not recommended and can lead to inadvertent data integrity loss.

Delete Process Flow

Preparing for an EIM delete process requires a thorough understanding of the parameter settings that specify delete criteria. You should be very careful and specific when setting delete-criteria parameters to avoid unintentional data loss. The EIM parameters mentioned in the following process flow are discussed in depth in "Parameters Used for Deletes in Both the Header and Process Sections" on page 109.

To delete data, EIM performs the following steps.

1. EIM initializes EIM tables for delete.

   If CLEAR INTERFACE TABLE in the configuration file is TRUE, all rows with the specified batch number are deleted. CLEAR INTERFACE TABLE must be FALSE for a delete process that uses EIM table values to identify rows for deletion.

2. EIM deletes rows.

   a. If the DELETE EXACT parameter in the configuration file is set to TRUE, EIM deletes the rows from the table that match the user key defined in the EIM table.

   b. If the DELETE MATCHES parameter in the configuration file is set to a base table, EIM deletes the rows from the target base table that match the predicate specified in the parameter.

c If the DELETE ALL ROWS parameter in the configuration file is set to TRUE, EIM deletes all rows from the target base table.

For information on configuration file parameters to use in a delete process, see "Parameters Used for Deletes in Both the Header and Process Sections" on page 109.

3 EIM sets IF_ROW_STAT to DELETED for rows that are successfully processed.

- When a foreign key column that references the deleted record is a required one, the record with the foreign key is deleted. Otherwise, the foreign key column is cleared.

  NOTE: If the record to be deleted is a parent, the child records are affected as described above. However, if a non-required foreign key is part of the user key and clearing it will create a conflict, then the record will be deleted.

- EIM deletion of a parent row causes cascade deletion of child rows only if the foreign key column in the child table is a mandatory column. Otherwise a cascade clear is performed.

NOTE: Because the delete process affects the contents of base tables, transaction logging should be in effect during delete operations if you have active mobile Web clients, so that the appropriate transactions are captured for later docking.

Preparing the EIM Tables for Delete Processing

This section provides assistance in loading the EIM tables with data used to control deletion of rows from Siebel base tables.

You must make sure that each EIM table row to be processed contains both data that correctly identifies the exact base table rows to delete and the appropriate values in the following columns.

ROW_ID. This value in combination with the nonempty contents of IF_ROW_BATCH_NUM must yield a unique value.

IF_ROW_BATCH_NUM. Set this to an identifying number for all EIM table rows to be processed as a batch.

IF_ROW_STAT. In each row to be deleted, set this column to FOR_DELETE to indicate that the row has not been deleted. After processing, if certain rows were not deleted due to a data error, you should change:

- IF_ROW_BATCH_NUM value for the rows that require redeleting.
- BATCH NUMBER line in the configuration file.

It is not possible to delete rows that have the same primary user key and different conflict IDs using EIM, because EIM relies on user keys to identify rows in base tables. If there are two rows in the base table that have the same user key but different conflict IDs, EIM cannot distinguish these rows. In such case, the IF_ROW_STAT field of the row in the EIM table will be marked as AMBIGUOUS.

NOTE: When you are deleting records based on user keys, specify the parameter DELETE EXACT in the .IFB file.

For more information on special columns, see "EIM Table Columns" on page 20. For general information on EIM tables, see Chapter 3, "Siebel EIM Tables."

Editing the Configuration File for Delete Processing

This section describes the header and process sections that you need in the EIM configuration file to properly configure EIM for a delete process. It also discusses the parameters in the configuration file that must be adjusted for the delete process. For general information on the EIM configuration file, see Chapter 4, "EIM Configuration File."

Before delete processing begins, you must change the configuration file to support this function. Such changes include:

- Editing the header and process sections and parameters
- Adjusting settings in the configuration file for the following purposes:
  - "Deleting All Data Rows" on page 113
  - "Deleting Data Rows Identified by User Key Values" on page 113
  - "Deleting from Base Tables Other Than the Target Base Table" on page 114
  - "Deleting Rows from Extension Tables" on page 115
  - "Deleting File Attachments" on page 115
  - "Handling Aborts of EIM Delete Processing" on page 116

Header Section Parameters Used for Deletes

Parameters in the header section generally apply to all types of processes. For a description of the necessary contents in the header section, see "Header Section Parameters Generic to All EIM Processes" on page 37.

Process Section Parameters Used for Deletes

Parameters in the process section apply only to that specific process and override any corresponding value in the header section for the specific process. This section describes the parameters used in the process section that are specific to a delete process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

To delete data, you must define at least one process with TYPE = DELETE.

If the process is defined with TYPE = DELETE, the DELETE ROWS parameter will be automatically set to TRUE. In some cases, you may not want to delete data from a nontarget base table as a result of cascade action. In this case, use the DELETE ROWS parameter to prevent deletion of rows from a specified table. The following example contains lines that can be used in the EIM configuration file to define a delete process for the accounts table while preventing rows from being deleted in the S_ADDR_ORG table.

```
[Delete Accounts]
  TYPE = DELETE
  BATCH = 200
  TABLE = EIM_ACCOUNT
  DELETE ROWS = S_ADDR_ORG, FALSE
  DELETE EXACT = TRUE
  ONLY BASE TABLES = S_ORG_EXT
```

Parameters Used for Deletes in Both the Header and Process Sections

This section describes the parameters that can appear in either the header section or a process section and are specific to a delete process. For generic parameters that can be used in all EIM processes, see "Header Section Parameters Generic to All EIM Processes" on page 37 and "Process Section Parameters Generic to All EIM Processes" on page 39.

Table 16 provides descriptions of the parameters that can appear in the header and process sections of the EIM configuration file, and which are specific to delete processes.

Table 16. Delete Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
CASCADE DELETE ONLY	(Default = FALSE). Set this parameter to TRUE to delete child records with nullable foreign keys when the parent record is deleted. If FALSE, then when EIM deletes a parent record, it sets the foreign keys of the child records to NULL.
CLEAR INTERFACE TABLE	This parameter specifies whether existing rows in the EIM table for the given batch number should be deleted. Valid values are true (the default unless DELETE EXACT = TRUE) and false (the default if DELETE EXACT = FALSE).
DELETE ALL ROWS	Used for deleting all rows in table; default is FALSE. **NOTE:** Use this parameter with caution. For more information on this parameter, see "DELETE ALL ROWS Parameter" on page 112.
DELETE EXACT	Delete using user key matching algorithm with rows in EIM table; default is FALSE. For more information on this parameter, see "DELETE EXACT Parameter" on page 111.
DELETE SKIP PRIMARY	This parameter specifies whether EIM should perform a cascade update to the primary child column. The default value is TRUE.
DELETE MATCHES	SQL WHERE fragment deletion criteria. Example: DELETE MATCHES = EIM_ACCOUNT, (NAME LIKE "TST_ACCT%").
DELETE ROWS	This parameter specifies whether rows from the target base table can be deleted. Valid values are TRUE (the default) and FALSE. This parameter can prevent deletions from one table while allowing them in others. For example, the following parameter setting prevents deletion of rows from the S_ADDR_ORG table: `DELETE ROWS=S_ADDR_ORG, FALSE` **NOTE:** Use the FALSE setting for DELETE ROWS carefully. Inappropriate use can result in dangling foreign key pointers.

Table 16. Delete Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
IGNORE BASE COLUMNS	Specifies base table columns to be ignored by the import process. Use commas to separate column names, which can be qualified with base table names. Required and user key columns cannot be ignored. Use this parameter to improve performance when updating all but a few columns. The default is to not ignore any base table columns.
UPDATE ROWS	Specifies whether foreign key references can be updated. This parameter can be used to prevent the updating of foreign key references with a setting of FALSE. The default value is TRUE, which affects all tables. To affect only specific tables, you can specify a table name. For example: `UPDATE ROWS = S_CONTACT, TRUE` The UPDATE ROWS parameter also prevents updates in one table while allowing them in others. If this parameter is set to FALSE, EIM does not update rows in the specified base table. If you need to specify multiple tables, use one UPDATE ROWS statement for each table. **NOTE:** Use the FALSE setting for UPDATE ROWS carefully. Inappropriate use can result in dangling foreign key pointers.

NOTE: You *must* use one of the following DELETE parameters described in this section: DELETE EXACT, DELETE MATCHES, or DELETE ALL ROWS.

DELETE EXACT Parameter

This parameter specifies the base table rows to delete by using user key values specified in the EIM table. By default, DELETE EXACT = FALSE. If DELETE EXACT is set to TRUE, you must use the ONLY BASE TABLES parameter in conjunction with this parameter to identify the base tables.

NOTE: Do not use ONLY BASE TABLES with the target base table *and* nontarget base tables, because the EIM table record cannot specify just one record to be deleted.

Although this parameter can be used to delete rows from both target and nontarget base tables, use the DELETE EXACT parameter to delete only nontarget base tables *containing user keys*. Rows in nontarget base tables that do not contain user keys will not be deleted. For example, you cannot use the DELETE EXACT parameter to update the S_ACTION_ARG table and the S_ESCL_ACTION table because there are no user keys defined for these tables.

As another example, you can use DELETE EXACT to delete any of the nontarget base tables such as S_ADDR_PER and S_ACCNT_POSTN using the EIM_ACCOUNT table. In this case, the EIM_ACCOUNT table would need to be loaded with records that would singularly identify the S_ACCNT_POSTN or the S_ADDR_PER record to be deleted.

To use the DELETE EXACT parameter to delete data from base tables other than the target base table, specify the user key columns only for a single base table for each row in the EIM table. When specifying rows for exact deletion, make sure any columns not necessary to specify the row to be deleted are NULL to avoid problems with deleting from the wrong base table. EIM tries to enforce this behavior by requiring other user key columns to be NULL. If a row cannot be identified as clearly referring to a row in a single base table, that row will fail to be deleted.

"Deleting from Base Tables Other Than the Target Base Table" on page 114 explains how to delete data from base tables other than the target base table using the DELETE EXACT parameter with the following scenario as an example. In this example, EIM_ACCOUNT is mapped to base tables including S_ORG_EXT, S_ORG_PROD, and S_ORG_INDUST. You should delete data only from S_ORG_PROD, and you should not delete data from S_ORG_EXT or any other base tables.

DELETE MATCHES Parameter

This parameter specifies a WHERE clause expression for filtering base table rows. The value is in two parts: the Siebel base table name and the filter expression that goes against the target base table. An example would be:

```
DELETE MATCHES = S_ORG_EXT, (LAST_UPD > '2000-06-22' AND LAST_UPD < '2000-06-23')
```

The expression is a self-contained WHERE clause expression (without the WHERE) and should use only literal values or column names (optionally prefixed with the base table name). There must also be a space separating the operator from the operand in this expression (a space must be added between > and '). When deleting rows for a specific date, you should use date ranges as shown in the example instead of setting the date equal to a specific date. By default, DELETE MATCHES expressions are not used.

This parameter will only write the user keys values of the deleted target table rows to the EIM table columns. It will not write values of nonuser keys columns or nontarget table rows column values to the EIM table. The deleted rows cannot be reimported using the EIM table rows written by the EIM delete process, because they will not contain all the original information.

Only use this parameter to delete rows from target base tables. Rows will be deleted from the target base table even if the DELETE ROWS parameter is set to FALSE for that table.

CAUTION: Do not use the DELETE MATCHES parameter to delete rows from S_PARTY based tables. For example, using the criteria "DELETE MATCHES = S_PARTY, (CREATED > xxxxx)" will cause all records of S_PARTY that matches this criteria to be deleted from the database.

DELETE ALL ROWS Parameter

This parameter specifies that all rows in the target base table are to be deleted. Valid values are TRUE and FALSE (the default). Existing values in the EIM table and DELETE MATCHES expressions are ignored.

This parameter will only write the user keys values of the deleted target table rows to the EIM table columns. It will not write values of nonuser keys columns or nontarget table rows column values to the EIM table. The deleted rows cannot be reimported using the EIM table rows written by the EIM delete process, because they will not contain all the original information.

CAUTION: ***Use the DELETE ALL ROWS = TRUE setting with extreme caution.*** It will delete all rows in the named base table including any seed data. Do not remove unnecessary seed data by deleting all rows from the S_LST_OF_VAL base table. If you do so, you will not be able to reimport "clean"data and you will be forced to rebuild the seed data or restore from backup. To selectively delete rows, use the DELETE EXACT or DELETE MATCHES expressions.

Deleting All Data Rows

If you want to delete all data rows in a target base table, you must perform the following procedure. Typically, this would only be performed in a test environment.

To delete all rows in a target base table

- Set the DELETE ALL ROWS parameter in the EIM configuration file to TRUE; its default value is FALSE.

  The following example contains lines that can be used in the EIM configuration file to delete all rows from the accounts table:

```
[Delete Accounts]
  TYPE = DELETE
  BATCH = 200
  TABLE = EIM_ACCOUNT
  DELETE ALL ROWS = TRUE
```

CAUTION: Use the DELETE ALL ROWS = TRUE setting with extreme caution. It will indeed delete all rows in the target base table.

Deleting Data Rows Identified by User Key Values

You must complete the following procedure to delete rows identified by user key values.

To delete rows with user key values appearing in the EIM tables

1. Set the DELETE EXACT parameter in the EIM configuration file to TRUE; its default value is FALSE.
2. Add the ONLY BASE TABLES parameter and set this parameter to the name of the base table you want to delete.

   The following example contains lines that can be used in the EIM configuration file to delete rows with user key values in the EIM tables from the Accounts table:

```
TYPE = DELETE
BATCH = 200
TABLE = EIM_ACCOUNT
DELETE EXACT = TRUE
ONLY BASE TABLES = S_ACCNT_POSTN
```

NOTE: Although you can use the DELETE EXACT parameter to delete rows from both target and nontarget base table, you should only use it to delete nontarget base tables that contain user keys. Rows in nontarget base tables that do not contain user keys will not be deleted.

Rows from the following tables do not have primary user keys and thus cannot be deleted using this parameter:

- Notes
- Territory Items
- Fulfillment Items

Deleting from Base Tables Other Than the Target Base Table

To use the DELETE EXACT parameter to delete data from base tables other than the target base table, specify the user key columns only for a single base table for each row in the EIM table. When specifying rows for exact deletion, make sure any columns that are not necessary to specify the row to be deleted are NULL to avoid problems with deleting from the wrong base table. EIM tries to enforce this behavior by requiring other user key columns to be NULL. If a row cannot be identified as clearly referring to a row in a single base table that row will fail to be deleted.

The following procedure explains how to delete data from base tables other than the target base table using the DELETE EXACT parameter with the following scenario as an example. In this example, EIM_ACCOUNT is mapped to base tables including S_ORG_EXT, S_ORG_PROD, and S_ORG_INDUST. If you want to delete data only from S_ORG_PROD, and not delete data from S_ORG_EXT or any other base tables, complete the following procedure.

To delete data from base tables other than the target base table

1. Populate the following columns in the EIM table (such as user keys for the S_ORG_PROD table and all the special interface columns):
   - ACCNT_NAME
   - ACCNT_LOC
   - INS_PROD_NAME
   - INS_PROD_VENDR
   - INS_PROD_VENDR_LOC
   - INS_DT, ROW_ID
   - IF_ROW_BATCH_NUM

- IF_ROW_STAT
- ROW_ID

2 Add or modify the following process section in your .IFB file:

TYPE = DELETE

BATCH NUMBER = *<number used to populate IF_ROW_BATCH_NUM column>*

TABLE = EIM_ACCOUNT

ONLY BASE TABLES = S_ORG_PROD

DELETE EXACT=TRUE

3 Run EIM.

This deletes all rows from the S_ORG_PROD table that have user keys that match the rows in your EIM table.

Deleting Rows from Extension Tables

You cannot delete a row from one-to-one extension tables (*_X type) without removing its parent row. For example, to remove a row from S_CONTACT_X, you must drop the parent row from S_CONTACT.

If you have to get rid of data in an extension column, update it with NULL by setting NET CHANGE = FALSE in the configuration file, and if necessary, use ONLY BASE COLUMNS.

Deleting File Attachments

You can also delete file attachments that have previously been imported into the Siebel database.

In order to delete file attachments, EIM deletes the row pointing to the file attachment. After all file attachments have been deleted, use the Siebel File System Maintenance Utility named sfscleanup.exe during hours when the network is least laden to clean the file attachment directory of any unused file attachments.

To delete file attachments

1 Run an EIM delete process for all file attachments that you want to delete.

2 After all file attachments have been deleted, run the Siebel File System Maintenance Utility named sfscleanup.exe to clean up the file attachment directory.

For information on using sfscleanup.exe, see *Siebel System Administration Guide*.

Handling Aborts of EIM Delete Processing

If an EIM delete process is aborted, base tables associated with deleted rows may not be updated. Orphans rows may be created because foreign keys may not have been updated. This may cause critical data integrity issues.

To avoid this problem, you should set the following parameters in the .IFB file to make sure that the EIM delete process performs only one commit and rollback when aborted:

COMMIT EACH TABLE = FALSE

COMMIT EACH PASS = FALSE

ROLLBACK ON ERROR = TRUE

Running a Delete Process

You may run a delete process after you have:

- Identified the data for delete processing
- Prepared the related EIM tables
- Modified the EIM configuration file accordingly

Run the delete process by completing the procedures in Chapter 9, "Running EIM."

Checking Delete Results

When a delete process ends, you should carefully check the results to verify that data was successfully deleted. During each process, EIM writes comprehensive status and diagnostic information to several destinations.

EIM uses a special column named T_DELETED_ROW_ID in the EIM tables. EIM writes the ROW_ID of each deleted base table row to this column.

To view a list of deleted base table rows

- Query the appropriate EIM table for rows whose IF_ROW_BATCH_NUM equals the batch number for the delete.

  The value of T_DELETED_ROW_ID identifies deleted rows.

If error flags, SQL trace flags, or trace flags were activated for the EIM process, you can also use the trace file to view the results of the EIM process. For more information on viewing the trace file, see "Viewing the EIM Log File" on page 128.

8 Merging Data

This chapter covers the process of merging data into the Siebel database. This chapter is organized into the following sections:

Overview of EIM Merge Processing

EIM uses a combination of EIM table row contents and configuration file parameter values to control the merge process. A merge process deletes one or more existing rows from the base table and makes sure that intersecting table rows are adjusted to refer to the remaining rows. Data from the record you select as the surviving record is preserved. Data from the other records is lost. If there are other records associated with the records you merge, those records—with the exception of duplicates—are associated with the surviving record.

Duplicate child records of the deleted rows will have CONFLICT_ID updated during the merge process. For example, when merging two Accounts (parent), the user keys of the Contacts (child) will be compared, and if the same Contact belongs to both Accounts, the Contact of the deleted Account will have its CONFLICT_ID updated.

You can only merge records that have primary user keys. Because records in the following tables do not have primary user keys, these records cannot be merged:

- Notes
- Territory Items
- Fulfillment Items

  CAUTION: Using EIM to merge data in the Products and Positions base tables is not recommended and can lead to inadvertent data integrity loss.

It is not possible to merge rows that have the same primary user key and different conflict IDs using EIM, because EIM relies on user keys to identify rows in base tables. If there are two rows in the base table that have the same user key but different conflict IDs, EIM cannot distinguish between these rows. In such cases, the IF_ROW_STAT field of the row in the EIM table will be marked as AMBIGUOUS.

EIM can only be used to merge rows from target base tables and not secondary tables. For example, the target base table for EIM_ASSET is S_ASSET. EIM can only be used to merge two or more S_ASSET rows into single S_ASSET rows. You cannot use EIM to merge two or more S_ASSET_CON rows into single S_ASSET_CON rows.

EIM Merge Process

During its multiple passes through the EIM tables, EIM completes the following tasks within a merge process:

- Initialize the EIM tables for merge.
- Select for merge the rows with matching user keys in the EIM tables.
- Merge child rows into the replacement rows. EIM then deletes rows from the target base table that are specified in the EIM table.
  - For deleted rows, EIM sets T_MERGED_ROW_ID to the ROW_ID of the row that was merged into (the surviving row).
  - EIM sets T_DELETED_ROW_ID to the ROW_ID of the deleted base table row.
- Update child rows containing foreign keys that point to newly deleted rows. For base tables that have foreign keys in newly deleted rows, EIM updates the foreign keys to point to surviving rows (depending on the value for UPDATE ROWS in the configuration file).

EIM provides comprehensive status information about each merge process. When the process ends, you should review this information. For more information, see "Checking Merge Results" on page 123.

Each task involves multiple passes; at least one pass is required for each EIM table included in the process.

NOTE: Because the merge process affects the contents of base tables, transaction logging should be enabled during merge operations if you have active mobile Web clients, so that the appropriate transactions are captured for later synchronization. For more information, see "Enabling Transaction Logging for Merge Processing" on page 122.

Running through the EIM merge process requires that you perform the following steps, which are discussed in the remaining sections of this chapter:

1. "Preparing the EIM Tables for Merge Processing" on page 119.
2. "Editing the Configuration File for Merge Processing" on page 120.
3. "Running a Merge Process" on page 122.
4. "Checking Merge Results" on page 123.

Preparing the EIM Tables for Merge Processing

This section provides assistance in loading the EIM tables with data used to control the process of merging rows in Siebel applications base tables. Your database administrator can use the loading tool provided by your database.

You must make sure that each EIM table row to be processed contains the appropriate values in the following columns. Table 17 shows a merge example for special columns.

Table 17. EIM Merge Example for Special Columns

IF_ROW_BATCH_NUM	NAME	ROW_ID	IF_ROW_MERGE_ID
1	IBM	100	NULL
1	IBM Japan	101	100
1	IBM Europe	102	100

IF_ROW_BATCH_NUM. Set this to an identifying number for all EIM table rows to be processed as a batch.

ROW_ID. This value in combination with the nonempty contents of IF_ROW_BATCH_NUM must yield a unique value.

IF_ROW_MERGE_ID. Set this value to one of two values. For an EIM table row whose ROW_ID and IF_ROW_BATCH_NUM columns identify the surviving or merged-into row, set this value to NULL. For EIM table rows whose ROW_ID and IF_ROW_BATCH_NUM columns identify a row to be merged (and subsequently deleted), set this value to the ROW_ID where this row will be merged. Upon completion of the merge process, the first row survives and the remaining rows are deleted. All child and intersection table rows that previously pointed to ROW_IDs 101 and 102 now point to 100.

IF_ROW_STAT. In each row to be merged, set this column to FOR_MERGE to indicate that the row has not been merged. After processing, if certain rows were not merged due to a data error, you should change:

- IF_ROW_BATCH_NUM value for the rows that require remerging.
- BATCH NUMBER line in the configuration file.

NOTE: In addition to populating these columns, user key information for each row to be merged must be loaded into the EIM table.

If you do not correctly populate all the user key columns, the merge process will fail and the IF_ROW_STAT column in the EIM table will be set to the value NO_SUCH_RECORD. This indicates that EIM cannot find the appropriate rows to merge using the specified user keys.

For more information on special columns, see "EIM Table Columns" on page 20. For general information on EIM tables, see Chapter 3, "Siebel EIM Tables."

Editing the Configuration File for Merge Processing

This section describes the header and process sections that you need in the EIM configuration file to properly configure EIM for a merge process. For general information on the EIM configuration file, see Chapter 4, "EIM Configuration File."

Before merge processing begins, you must change the configuration file to support this function. Such changes include:

- Editing the header and process sections and parameters
- Adjusting settings in the configuration file in the following ways:
  - "Updating Affected Rows" on page 121
  - "Avoiding Aborts of EIM Merge Processing" on page 121
  - "Enabling Transaction Logging for Merge Processing" on page 122
  - "Specifying Survivor Records for Merge Processes" on page 122

Header Section Parameters Used for Merges

Parameters in the header section generally apply to all types of processes. For a description of the necessary contents in this section, see "Header Section Parameters Generic to All EIM Processes" on page 37.

Process Section Parameters Used for Merges

Parameters in the process section apply only to that specific process and override any corresponding value in the header section for the specific process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

To merge data, you must define at least one process with TYPE = MERGE. The following example contains lines that can be used in the EIM configuration file to define a merge process for the Accounts table.

```
[Merge Accounts]
  TYPE = MERGE
  BATCH = 1
  TABLE = EIM_ACCOUNT
  UPDATE ROWS = TRUE
```

NOTE: For performance reasons, you should limit the number of tables to merge in a single process section to five or less.

Parameters Used for Merges in Both the Header and Process Sections

Table 18 describes the parameters that can appear in either the header section or a process section, and are specific to a merge process. For generic parameters that can be used in all EIM processes, see "Process Section Parameters Generic to All EIM Processes" on page 39.

Table 18. Merge Process Parameters for the EIM Configuration File - Header and Process Sections

Parameter	Description
SET BASED LOGGING	Specifies whether set-based logging is enabled. The default value is TRUE. **NOTE:** EIM will ignore this parameter if Docking: Transaction Logging is set to FALSE in the System Preferences view. For more information on this parameter, see "SET BASED LOGGING Parameter" on page 121.
UPDATE ROWS	Specifies whether the foreign key (or keys) that reference the merged rows in the named table need to be adjusted. Valid values are TRUE (the default) and FALSE. **NOTE:** Use the `UPDATE ROWS = Table_Name, FALSE` setting carefully. Inappropriate use can result in dangling foreign key pointers.

SET BASED LOGGING Parameter

When set-based logging is enabled, a separate log entry is generated for all rows in each table affected by EIM. This allows greater performance improvement because EIM can perform the operations as set operations in SQL, without resorting to row-by-row processing to support the transaction log. Set-based transaction logging is most useful when a table is read-only to mobile Web clients. Set-based logging is always the default for merge. The SET BASED LOGGING parameter must be set to FALSE to allow transaction logging for merge.

Updating Affected Rows

During a merge operation, a specific base table may have some rows deleted and others updated. You can use the UPDATE ROWS parameter to prevent updates to one base table while allowing updates to another. By default, UPDATE ROWS = TRUE.

Avoiding Aborts of EIM Merge Processing

If an EIM merge process is aborted, base tables associated with merged rows may not be updated. Orphan rows may be created because foreign keys may not have been updated. This may cause critical data integrity issues.

To avoid this problem, set the following parameters in the .IFB file so the EIM merge process performs only one commit or rollback when aborted:

COMMIT EACH TABLE = FALSE

COMMIT EACH PASS = FALSE

ROLLBACK ON ERROR = TRUE

Enabling Transaction Logging for Merge Processing

To enable transaction logging for an EIM merge process, set the following parameters in the .IFB file so the EIM merge process runs in ongoing (row-by-row) mode:

LOG TRANSACTIONS= TRUE

SET BASED LOGGING = FALSE

For information on the LOG TRANSACTIONS parameter, see "Optional Keywords for Process Parameters" on page 44. For information on the SET BASED LOGGING parameter, see "Process Section Parameters Used for Merges" on page 120.

Specifying Survivor Records for Merge Processes

In a merge process, data from the record you select as the surviving record is preserved, while data from the other records is lost. Do not specify the same record as both the survivor and the victim or it will be deleted. You should also make sure that a record is specified as a survivor only once in a batch.

NOTE: EIM behavior, whether executed from the GUI or through an EIM run, does not merge data in the base record. It simply repoints the foreign keys in the dependent child records. This applies to all columns in the base table. This could lead to unintended data loss in an extension column. For more information, see "Example of Running a Merge with Custom Columns" on page 196.

Running a Merge Process

You can run a merge process after you have:

- Identified the data for merge processing
- Prepared the related EIM tables
- Modified the EIM configuration file accordingly

Run the merge process by completing the procedures in Chapter 9, "Running EIM."

Checking Merge Results

When a merge process ends, you should carefully check the results to verify that data was successfully merged. During each process, EIM writes comprehensive status and diagnostic information to several destinations.

During a merge process, EIM writes the following values to two special columns in the EIM tables:

- T_DELETED_ROW_ID contains the ROW_ID of the deleted base table row.
- T_MERGED_ROW_ID contains the ROW_ID of the surviving base table row.

To view the results of a merge

1. Query the appropriate EIM table for rows whose IF_ROW_BATCH_NUM equals the batch number for the merge process.
2. Inspect the values of T_DELETED_ROW_ID and T_MERGED_ROW_ID.

If error flags, SQL trace flags, or trace flags were activated for the EIM process, you can also use the trace file to view the results of the EIM process. For more information on viewing the trace file, see "Viewing the EIM Log File" on page 128.

9 Running EIM

This chapter covers how to run an EIM process and check the results. This chapter is organized into the following sections:

Preparing to Run an EIM Process

You can run an EIM process (import, export, delete, or merge) once you have:

- Identified the data for EIM processing
- Prepared the related EIM tables
- Modified the EIM configuration file accordingly

You can start an EIM process by running a server task for the Enterprise Integration Manager component. You can run the server task using either the GUI or the command-line interface. For more information on running server tasks, see *Siebel System Administration Guide*.

Running an EIM Process

On each pass, EIM processes one EIM table and performs a particular action on all rows in that table for that batch. Most passes affect only the EIM table's temporary columns; for example, resolving foreign keys. There are two methods for running an EIM process:

Running an EIM Process Using the Graphical User Interface (GUI)

The most common method for starting an EIM server task is to use the graphical user interface (GUI). When performing this procedure, be aware that passes in Step 8 on page 54 (update), Step 9 on page 54 (insert), and Step 10 on page 55 (primary keys) affect the base tables. All steps are performed for all columns used in the import process.

CAUTION: If you are running EIM on a DB2 database, then set the database configuration parameters as described in the *Siebel Installation Guide* for the operating system you are using, or EIM will not run successfully. You should also run the `updatestats.sql` script (located in dbserver_home\db2) each time before running EIM, or performance issues may be encountered when loading the dictionary. For more information, see Chapter 10, "EIM Performance Tuning."

To run an EIM process using the GUI

1. Navigate to Administration - Server Management > Jobs.
2. In the Jobs list, click New.

   The component job status field changes to Creating.
3. In the Component/Job field, click the drop-down list and select the Enterprise Integration Mgr component.

   If you want to use a component job template based on EIM for your component request, you must first define the component job template. For information on defining component job templates, see *Siebel System Administration Guide*.
4. In the Job Detail view, enter data in other appropriate fields as described in the table that follows.

Field	Description
Scheduled Start	The scheduled start date and time of the component job.
Expiration	The date at which the component job is no longer valid.
Requested Server	Set if you want to target a server component on a specific Siebel Server.
Request Key	Set if you want to target a component or repeating component job to a specific instance of the server component identified by the request key. In all other situations, keep this field blank.
Delete Interval	Set with Delete Unit field, this field determines the length of time before the component job is deleted. If not updated, this field defaults to 1.
Delete Unit	Set with Delete Interval field, this field determines the length of time before the component job is deleted. If not updated, this field defaults to Weeks.
Retry on Error	Check this box to retry the component job in case of error.

Field	Description
Sleep Time	This field is available when the Retry on Error check box is true and determines the amount of time before the component job is retried.
Number of Retries	This field is available when the Retry on Error check box is true and determines the number of times the component job is retried.

5 Click the menu button, and then click Save Record.

6 In the Job Parameters list, add or change any component job parameters for the EIM process:

   a Click the New button.

   b In the Name field, click the Select button.

      The Job Parameters dialog box appears. The parameters that appear in the Job Parameters dialog box vary depending on the server component you selected in Step 3.

   c Select a parameter in the Component Parameters dialog box, and modify its value.

   d Click the menu button and then click Save Record.

7 In the Jobs list, click the Start button.

   The Status field changes from Creating to Queued.

   CAUTION: EIM is a multistep process. ***Once the EIM process is running, do not stop or pause the task***. Otherwise, some steps may not roll back correctly.

Running an EIM Process Using the Command-Line Interface

You can also start the EIM server task through the command-line interface. For example, if you are using a UNIX operating system or if you have experienced the EIM server task being QUEUED when the job was submitted by the GUI, use the command-line interface to run an EIM process.

To run an EIM process using the command-line interface

1 Start the srvrmgr program in the command-line interface.

   For information on srvrmgr program, see *Siebel System Administration Guide*.

2 Execute a start task command or a run task command on the Enterprise Integration Mgr component. Be sure to specify the configuration file with the config parameter.

NOTE: You cannot use the Uniform Naming Convention (UNC) in the Server Manager command-line interface when specifying the configuration file.

If you do not specify a configuration file, the default.ifb configuration file will be used. If you put the .IFB file you want to use in a directory other than the default directory (<*SiebSrvr\Admin*> folder), you will need to specify the path to the .IFB file when you start the EIM component.

The following example shows how to use the run task command to start an import process:

```
run task for component eim with config=import.ifb
```

For more information on the start task command and the run task command, see *Siebel System Administration Guide*.

CAUTION: EIM is a multistep process. *When the EIM process is running, do not interrupt the task.* Otherwise, some steps may not roll back correctly.

The following example shows how to use the run task command to start an import process that uses a different LOV language than the default setting of the EIM LOV language parameter:

```
run task for component eim with config=import.ifb, LovLang=DEU
```

Viewing the EIM Log File

In the Task Info Log view, you can view information about the results of an EIM process by drilling down in an EIM server task that has completed. This information is also provided in the EIM log file within the `siebel server\log` directory. The log consists of three general sections:

- **Startup messages.** This section pertains to dictionary loading, parameter loading, and .IFB file parsing.
- **Run-time messages.** This section shows the begin and end times for each process.
- **Row-count summary of each process.** This section shows the number of rows updated in each table.

If error flags, SQL trace flags, or trace flags were activated for the EIM process, the EIM log file will also contain the results of each flag. For more information on trace flags and error flags, see "Using Trace Flags, SQL Trace Flags, and Error Flags" on page 129.

Further information on the EIM log file is provided as follows:

- "Using Trace Flags, SQL Trace Flags, and Error Flags" on page 129
- "Setting Event Logging from the Graphical User Interface (GUI)" on page 131
- "Setting Event Logging from the Command-Line Interface" on page 131
- "Trace Flag Settings" on page 132

To view EIM log file information in the Task Info Log

1. Navigate to Administration - Server Management > Tasks.
2. In the Tasks list, select the task for the EIM process.
3. Click the Log tab.

   The log for the selected task is displayed in the Log list.

NOTE: You can also view this information by opening the EIM log file in the `siebel server\log` directory.

Using Trace Flags, SQL Trace Flags, and Error Flags

You can activate trace flags and error flags to log transactions. This topic covers the following types of flags:

- **Error flags.** See "Error Flags" on page 130.
- **SQL Trace flags.** See "SQL Trace Flags" on page 130.
- **Trace flags.** See "Trace Flags" on page 130.

NOTE: Activating flags will have a direct effect on performance. Typically, activating flags should only be done when testing EIM processes. Avoid activating flags in a production environment unless absolutely necessary.

Recommended settings for error flags, SQL trace flags, and trace flags include the following:

- **To display errors and unused foreign keys.** Start with the following setting combination. The setting Trace Flag=1 provides a summary (after each batch) of the elapsed time in EIM steps 10 and 11.

Error Flag	1
SQL Trace Flag	1
Trace Flag	1

- **To determine SQL performance.** The following setting combination produces a log file with SQL statements including the elapsed time for each statement.

Error Flag	1
SQL Trace Flag	8
Trace Flag	3

- **To determine optimal batch size and monitor performance in a particular step.** The following setting combination produces a log file showing the elapsed time for each EIM step.

Error Flag	0
SQL Trace Flag	0
Trace Flag	1

Error Flags

To activate error flags, you must complete Step 6 on page 127 when running an EIM process. Setting the Error Flags parameter to 1 produces a detailed explanation of rows that were not successfully processed.

There are a variety of reasons why rows might not be processed. The following sample shows an excerpt from an EIM Error Flag 1 trace. The log begins with a header that describes an export failure that occurred during Step 2, Pass 101.

```
2001-04-04 03:47:594/4/01 3:47 Warning: No rows in S_ORG_EXT matched by expressions
for export.

2001-04-04 03:47:59Process [Export Old Accounts] had all rows fail

2001-04-04 03:47:59 on EIM_ACCOUNT for ] 2001 in step 2, pass 101:

2001-04-04 03:47:59 No base table rows matched expressions. (severity 5)

2001-04-04 03:47:59Base table:

2001-04-04 03:47:59S_ORG_EXT (Account)

2001-04-04 03:47:59The match expressions specified for exporting rows through this
interface table

2001-04-04 03:47:59did not match any of the rows currently in the target base table.

2001-04-04 03:47:59Since there were no matches for the given match expressions,
processing for

2001-04-04 03:47:59this interface table was discontinued. However, processing of
other interface

2001-04-04 03:47:59tables will continue.

2001-04-04 03:47:59Recorded 1 group of failures.
```

SQL Trace Flags

To activate SQL trace flags, you must complete Step 6 on page 127 when running an EIM process.

Setting the SQL Trace Flags parameter to 8 creates a log of all SQL statements that make up the EIM task. The lower values for SQL Debug Flags (1, 2, and 4) are used for logging at the ODBC level.

Trace Flags

Trace flags contain logs of various EIM operations. To activate trace flags, you must complete Step 6 on page 127 when running an EIM process. If you are using Siebel 7.x, you also need to set event logging for the EIM component, as described in "Setting Event Logging from the Graphical User Interface (GUI)" on page 131.

Trace flags are bit-based. Available trace flags include 1, 2, 4, 8, and 32. To activate multiple trace flags, set the Trace Flags parameter to the sum of individual trace flag numbers. For example, to log trace flags 2 and 4, set the Trace Flags parameter to 6.

Setting Event Logging from the Graphical User Interface (GUI)

You can set event logging for the EIM component using the Administration - Server Configuration views in the Siebel client.

NOTE: You can also set event logging using the SrvrMgr command line. See "Setting Event Logging from the Command-Line Interface" on page 131.

To set event logging for the EIM component from the GUI

1. From the application-level menu, choose Navigate > Site Map > Administration - Server Configuration > Servers > Components > Events.
2. In the Components list, select Enterprise Integration Manager as the component.
3. Click the Events tab to view all the configurable event types for the selected component.

   The log level is set to a default value of 1.
4. Perform a query and enter the specified log level for each of the following event types:

Event Type	Log Level Value
EIM SQL	4
SQL Summary	4
Task Configuration	4
EIM Trace	3

NOTE: The event types EIM Debug, EIM Error, and EIM System Stats exist for compatibility with previous versions of the Siebel application. Do not change the default value for these parameters.

It is not necessary to restart the Siebel server to apply the event type log level changes. The changed settings are active in the next EIM task executed.

For more information on event logging administration, see *System Monitoring and Diagnostics Guide for Siebel eBusiness Applications*.

Setting Event Logging from the Command-Line Interface

You can also set event logging for the EIM component from the Server Manager command line.

To set event logging for the EIM component from the command line

- Use the following commands:

```
change evtloglvl SQLSummary=4 for component eim
```

```
change evtloglvl EIMSQL=4 for component eim

change evtloglvl TaskConfig=4 for component eim

change evtloglvl EIMTrace=3 for component eim
```

Other necessary commands for activating tracing levels are the following:

- **When running the EIM task.** Specify the following parameters:

  ```
  Srvrmgr> run task for component eim with config=<configfile.ifb>, TraceFLags=1,
  ErrorFlags=1, SQLFlags=8
  ```

- **To view existing event log levels for the EIM component.** Use the following command:

  ```
  Srvrmg> list evtloglvl for component eim
  ```

Trace Flag Settings

This topic provides Trace Flag setting information as follows:

Trace Flag 1

Setting the Trace Flags parameter to 1 creates a step-oriented log of the task. This can be used to determine the amount of time EIM spends on each step of the EIM task, or for each EIM table processed. The following sample shows an EIM Trace Flag 1 output:

```
Initializing

   Loading configuration fileimacct.ifb0s

   Opening server databaseora_dev6s

   Loading Siebel dictionary15s

 Initializing 21s

 Import Accounts 14

   ImportingEIM_ACCOUNT

     Step 1: initializing IF Table 0s

     Step 4: resolving foreign keys S_ORG_EXT 0s

     Step 5: locating existing rowS_ORG_EXT 0s
```

```
     Step 6: assigning new row IDsS_ORG_EXT 0s
Step 7: finding new foreign keys4s
     Step 9: inserting new rowsS_ORG_EXT2s
   ImportingEIM_ACCOUNT15s
Updating primaries
     Step 10: updating primary keysS_ORG_EXT3s
   Updating primaries3s
 Import Accounts1418s
```

Trace Flag 2

Setting the Trace Flags parameter to 2 creates a file log that traces all substitutions of user parameters. The following example shows an EIM Trace Flag 2 output:

```
[TRC01] Parameter Set << AFTER RESOLUTION >>
[TRC01]  UserParams = IFTABLE=EIM_ACCOUNT
[TRC01] [0] $IFTABLE = EIM_ACCOUNT
[TRC01] [1] $CURRENT_USER = wgong
[TRC01] [2] $CURRENT_DATETIME = 4/6/01 13:17
[TRC01] [Siebel Integration Manager]
[TRC01] log transactions = false
[TRC01] $COLUMN_VALUE = 'EIM ins_acct Test%'
[TRC01] [ins_acct_shell]
[TRC01] TYPE = SHELL
[TRC01] INCLUDE = del_acct
[TRC01] INCLUDE = ins_acct
[TRC01] [del_acct]
[TRC01] SESSIONSQL = DELETE FROM DEV50.EIM_ACCOUNT WHERE IF_ROW_BATCH_NUM=21
[TRC01] TYPE = DELETE
[TRC01] BATCH = 20
[TRC01] TABLE = EIM_ACCOUNT
[TRC01] $COLUMN_NAME = NAME
[TRC01] DELETE MATCHES = EIM_ACCOUNT,(NAME LIKE 'EIM ins_acct Test%')
```

```
[TRC01] [ins_acct]

[TRC01] SESSIONSQL = INSERT INTO DEV50.EIM_ACCOUNT (IF_ROW_STAT, ROW_ID,
IF_ROW_BATCH_NUM, ACCNT_NAME, ACCNT_LOC) SELECT 'X', ROW_ID, 21, 'EIM ins_acct Test
' || ROW_ID, 'Loc' FROM DEV50.S_SYS_PREF

[TRC01] TYPE = IMPORT

[TRC01] BATCH = 21

[TRC01] TABLE = EIM_ACCOUNT
```

Trace Flag 4

Setting the Trace Flags parameter to 4 creates a file log that traces all user-key overrides. The following example shows an EIM Flag 4 output for a user key override to the EIM_ACCOUNT table:

```
[TRC02] ------------------------------------------------------

[TRC02] ***** IF TABLE <EIM_ACCOUNT> uses USER_KEY_COL *****

[TRC02] Action: No Move & Insert

[TRC02] overriding UK Index (S_TERR_ITEM_U1) at position (0)

[TRC02] ##### Destination TABLE (S_TERR_ITEM) index vector: [S_TERR_ITEM_U1]

[TRC02] --- Column (T_TERITE_OUID) index vector: [S_TERR_ITEM_U1]

[TRC02] --- Column (T_TERITE_TERID) index vector: [S_TERR_ITEM_U1]

[TRC02] ------------------------------------------------------
```

Trace Flag 8

Setting the Trace Flags parameter to 8 creates a file log that traces all Interface Mapping warnings. The following example shows an EIM Flag 8 output for an Interface Mapping warning between the EIM_ACCOUNT and S_TERR_ITEM tables:

```
[TRC03] ------------------------------------------------------

[TRC03] IF table EIM_ACCOUNT destination S_TERR_ITEM

[TRC03]   IF column EIM_ACCOUNT.T_TERITE_TERID:

[TRC03] imports to: S_TERR_ITEM.TERR_ID

[TRC03] exports from: S_TERR_ITEM.TERR_ID

[TRC03]     Column NAME of join isn't in table!

[TRC03]     Missing join to user key NAME

[TRC03] ------------------------------------------------------
```

Trace Flag 32

Setting the Trace Flags parameter to 32 creates a file log that traces all file attachment status. The trace file contains four labels, three of which are used to trace file attachment processes as described in Table 19.

Table 19. Flag 32 Trace File Labels

Label	Description
Attachment Imported	Indicates whether the file attachment was encoded, compressed, and copied to the Siebel file server with the new name.
Attachment (Old) Deleted	This label applies only to updates and indicates whether an existing file was replaced and deleted.
Attachment Not Found	Indicates that the file attachment cannot be found in the input directory.

The following sample shows an EIM Flag 32 output for an opportunity file attachment:

```
[TRC32] Attachment Imported: E:\V50\output\openpost.doc ->
\\BALTO\SIEBFILE\ORADEV50\S_OPTY_ATT_10+413+1_10-41R-0.saf

[TRC32] Attachment (Old) Deleted: \\BALTO\SIEBFILE\ORADEV50\S_OPTY_ATT_10+413+1_10-
40Y-0.saf

[TRC32] Attachment Not Found: E:\V50\output\openpost.doc

[TRC32] Attachment Identical: E:\V50\output\openpost.doc IDENTICAL TO
\\BALTO\SIEBFILE\ORADEV50\S_OPTY_ATT_10+413+1_10-41R-0.saf
```

Optimizing EIM Performance

There are several ways you can improve EIM run-time performance. The best practices suggested in this section optimize EIM performance. For additional information on improving the performance of EIM, see Chapter 10, "EIM Performance Tuning."

Table Optimization for EIM

This section discusses ways that you can optimize tables for EIM processing.

Configuration Parameters

Limit base tables and columns to be processed. Four EIM parameters can help improve performance by limiting the affected tables and columns:

- only base Tables
- ignore base Tables

- only base Columns
- ignore base Columns

The ONLY BASE COLUMNS parameter is critical for the performance of an EIM process updating a few columns in many rows.

NOTE: Do not use the IGNORE BASE COLUMNS parameter for merge processes or export processes. This parameter should only be used for import processes and delete processes.

For other suggestions involving parameter settings, see "Parameter Settings Optimization for EIM" on page 138.

Indexes

Verify that all indexes exist for the tables involved. In most implementations, the tables and corresponding indexes in the following list tend to be the most heavily used and should be separated across devices. In general, the following indexes should be on different physical devices from the tables on which they are created.

- S_ACCNT_POSTN
- S_OPTY
- S_ADDR_ORG
- S_OPTY_POSTN
- S_CONTACT
- S_POSTN_CON
- S_DOCK_TXN_LOG
- S_PARTY_RPT_REL
- S_SRV_REQ
- S_EVT_ACT
- S_OPTY
- S_ORG_EXT

For organizations that plan to use EIM extensively, you should put your key EIM tables (based on your unique business requirements) on different devices from the Siebel base tables, because all tables are accessed simultaneously during EIM operations.

You can speed up deletes and merges involving S_ORG_EXT by adding an index to one or more columns. For more information, see "Adding Indexes to Improve Performance of S_ORG_EXT" on page 155.

Maintenance of EIM Tables

Perform regular table maintenance on EIM tables. Frequent insert or delete operations on EIM tables can cause fragmentation in the table. Ask your database administrator to detect and correct fragmentation in the EIM tables.

Always delete batches from EIM tables upon completion. Leaving old batches in the EIM table wastes space and can adversely affect performance. For other suggestions on working with batches, see "Limiting the Number of Records and Rows for Merge Processes" on page 137.

Batch Processing Optimization for EIM

This section suggests ways in which you can optimize EIM batch processing. Try using different batch sizes. Large batch sizes are often not efficient. For import and delete processes that use the DELETE EXACT parameter, use approximately 20,000 rows in a single batch.

Limiting the Number of Records and Rows for Merge Processes

You can improve performance by limiting the number of records in a batch. For information, see "Recommended Number of Rows for a Single Batch" on page 157.

Using Batch Ranges

Try using batch ranges (BATCH = x–y). This allows you to run with smaller batch sizes and avoid the startup overhead on each batch. The maximum number of batches that you can run in an EIM process is 1,000.

For IBM DB2, load a few batches of data into the EIM table and run EIM for just one of these batches. This primes the statistics in the DB2 catalogs. Afterward, do not update statistics on the EIM tables, and run EIM with the parameter UPDATE STATISTICS = FALSE in the .IFB file. This helps achieve consistent performance results when running EIM. See "Parameter Settings Optimization for EIM" on page 138 for other suggestions about parameters.

Run-Time Optimization for EIM

This section describes the ways you can optimize EIM performance at run time.

Parallel Processing

Run independent EIM jobs in parallel. Two or more EIM processes can be started simultaneously by using the Siebel Server Manager.

A special setup is not required to run EIM processes in parallel. For parallel processing, the following conditions must be met:

- No duplicate unique keys between runs for inserts.
- No duplicate updates or deletes between runs.

- No lock escalations on either EIM tables or target tables can be tolerated. Set LOCKLIST and MAXLOCKS as high as necessary to prevent this.

NOTE: If you run EIM jobs in parallel on the same base tables, you might encounter unique constraint errors if you have the same values for the unique index fields in batches being processed by two different EIM jobs.

CAUTION: Running EIM processes in parallel on a DB2 database may cause a deadlock when multiple EIM processes access the same EIM table simultaneously. To avoid this potential problem, set the UPDATE STATISTICS parameter to FALSE in the EIM configuration file. The UPDATE STATISTICS parameter is applicable only for DB2. For other suggestions, see "Parameter Settings Optimization for EIM" on page 138.

For more information on parallel processing, see "Running EIM Tasks in Parallel" on page 160.

Transaction Logging

Consider disabling the Docking: Transaction Logging system preference during the EIM run. Switching off transaction logging improves performance; however, this benefit must be balanced with the need for mobile users to reextract afterward. To disable transaction logging, complete Step 2 on page 58.

Parameter Settings Optimization for EIM

This section discusses ways that you can optimize EIM performance through parameter settings.

USING SYNONYMS Parameter for Optimizing EIM

Ignore account synonyms. Set the USING SYNONYMS parameter to FALSE in the .IFB file to indicate that account synonyms can be ignored during processing. This logical operator indicates to EIM that account synonyms do not require processing during import, thus reducing the amount of processing. Do not set the USING SYNONYMS parameter to FALSE if you plan to use multiple addresses for accounts. Otherwise, EIM will not attach addresses to the appropriate accounts. You can use EIM_ACCOUNT to import accounts with multiple addresses and then specify the primary address for an account by setting ACC_PR_ADDR to Y.

Trace Flag Settings for Optimizing EIM

Generate a task log to identify slow-running steps and queries by using Trace Flags. To use Trace Flags, set Error Flags=1, Trace Flags=1, and SQL Trace Flags=8. Rerun the batch and use the resulting task log to determine which steps and queries are running especially slowly. For additional information on trace flag settings, see "Trace Flag Settings" on page 132.

Database Server Optimization for EIM

The overall performance of EIM is largely dependent on the overall performance of the database server. To achieve optimal database server performance, it is critical that the tables and indexes in the database be arranged across available disk devices in a manner that evenly distributes the processing load.

The mechanism for distributing database objects varies by RDBMS, depending on the manner in which storage space is allocated. Most databases have the ability to assign a given object to be created on a specific disk.

A redundant array of independent disks (or RAID) can provide large amounts of I/O throughput and capacity, while appearing to the operating system and RDBMS as a single large disk (or multiple disks, as desired, for manageability).

The use of RAID can greatly simplify the database layout process by providing an abstraction layer above the physical disks while achieving high performance. Regardless of the RDBMS you implement and your chosen disk arrangement, be sure that you properly distribute the following types of database objects:

- Database log or archive files.
- Temporary workspace used by the database.

By following these suggestions, you should be able to improve the performance of the database server.

10 EIM Performance Tuning

This chapter covers recommended best practices for improving the performance of EIM and is organized into the following sections:

Architecture Planning Requirements

You must consider the size and complexity of the implementation before executing any single item with the Siebel application. Aspects that have a direct impact on how the production application will perform may not be your highest priority when you initially begin your Siebel implementation. However, the decisions made during the initial phases of an implementation have a far reaching impact, not only on performance and scalability but also on the overall maintenance of the Siebel application. It is strongly recommended to have a Siebel certified principal consultant or architecture specialist from Expert Services involved in designing the most effective logical and physical architecture for your organization. This includes capacity planning and system sizing, physical database layout, and other key architecture items.

Database Sizing Guidelines

One of the most important factors to determine about the database is its overall size. During the planning phase, you need to allocate space for system storage, rollback segments and containers, temporary storage space, log files, and other system files required by the relational database management system (RDBMS), as well as space for the Siebel application data and indexes. If you allocate too little space for the system, performance will be affected and, in extreme cases, the system itself may be halted. If you allocate too much space, it may cause inefficiency.

The space needed by the database depends on the total number and types of supported users. It is recommended that you consult your vendor RDBMS technical documentation for more information on these requirements.

The space required for Siebel data and indexes depends on the functionality being implemented and the amount and nature of data supporting this functionality.

The process for making accurate database size calculations is a complex one involving many variables. Use the following guidelines:

- Determine the total number, and types, of users of Siebel eBusiness applications (for example, 500 sales representatives and 75 sales managers).
- Determine the functionality that you will implement and the entities required to support them. Typically, the largest entities are as follows:
  - Accounts
  - Activities
  - Contacts
  - Forecasts
  - Opportunities
  - Service Requests
- Estimate the average number of entities per user (for example, 100 accounts per sales representative) and calculate an estimated total number of records per entity for the total user base.
- Using standard sizing procedures for the specific database, and *Siebel Data Model Reference*, calculate the average record size per entity and multiply by the total number of records. Typically, these entities span multiple physical tables, all of which must be included in the row size calculation. This determines the estimated data sizes for the largest entities.
- You must add additional space for the storage of other Siebel application data. A rough guideline for this additional amount would be one-half the storage required for these key entities.
- Indexes typically require approximately the same amount of space as data.
- Be sure to allow for a margin of error in the total size calculation.
- Be sure to factor growth rates into the total size calculation.

Database Layout Guidelines (Logical and Physical)

As with most Siebel Smart Web Architecture applications, the overall performance of Siebel eBusiness applications is largely dependent on the input/output (I/O) performance of the database server. To achieve optimal I/O performance, it is critical that the tables and indexes in the database be arranged across available disk devices in a manner that evenly distributes the I/O load.

The mechanism for distributing database objects varies by RDBMS, depending on the manner in which storage space is allocated. Most databases have the ability to assign a given object to be created on a specific disk. These objects, and guidelines for some of them, are provided in the following list.

A redundant array of independent disks, or RAID, can provide large amounts of I/O throughput and capacity, while appearing to the operating system and RDBMS as a single large disk (or multiple disks, as desired, for manageability). The use of RAID can greatly simplify the database layout process by providing an abstraction layer above the physical disks while ensuring high performance. Regardless of the implemented RDBMS and the chosen disk arrangement, be sure that you properly distribute the following types of database objects:

- Database log or archive files.
- Temporary workspace used by the database.
- Tables and Indexes: In most implementations, the tables and corresponding indexes in the following list tend to be some of the more heavily used and should be separated across devices. For a more complete listing, see the *Siebel Installation Guide* for the operating system you are using. In general, the indexes listed below should be on different physical devices from the tables on which they are created.

  - S_ACCNT_POSTN
  - S_OPTY
  - S_ADDR_ORG
  - S_OPTY_POSTN
  - S_CONTACT
  - S_POSTN_CON
  - S_DOCK_TXN_LOG
  - S_PARTY_REL
  - S_PARTY
  - S_SRV_REQ
  - S_EVT_ACT
  - S_OPTY
  - S_ORG_EXT

  NOTE: If you plan on making extensive use of EIM, put the key EIM tables (based on the unique business requirements) and their corresponding indexes on different devices from the Siebel base tables and indexes, because all of them are accessed simultaneously during EIM operations.

EIM Usage Planning

This section provides a number of general guidelines for effective and efficient implementations of EIM, regardless of the size of the overall Siebel implementation. It cannot be emphasized enough that taking a strategic perspective to implementing EIM is crucial not only to being able to use EIM, but to the overall success of the Siebel implementation.

Team Definition

Based on customer experience, it is recommended that a team of individuals is assigned to manage and maintain the EIM processes required for your organization. You should consider using individuals with the following skill sets:

- For small to medium-sized Siebel application implementations:

- A database administrator with a detailed understanding of not only the RDBMS used by your organization, but also the Siebel Data Model. This individual would be responsible for identifying the actual data to be loaded into the EIM tables and making sure that the physical layout of the database is done in a way that provides optimal performance. This person would also be responsible for the crucial task of mapping the data into the Siebel base tables. For more information on performing this task, see "Mapping Data into Siebel Applications" on page 144.
- A system administrator with a strong background in the systems (both the database server and application server) used by your organization. This individual would be responsible for developing scripts unique to your organization to automate the loading of data into the EIM tables, and to execute EIM in order to process the data into the Siebel base tables.

NOTE: Your organization may have one individual with both these skill sets and so you might rather dedicate only a single individual to these tasks. If this is the case, consider having a backup person, so that when this primary individual is unavailable, the backup person is capable of performing what needs to be done to keep the Siebel implementation operational.

- For larger to very large-sized Siebel implementations:
  - A database administrator with a detailed understanding of not only the RDBMS used by your organization, but also the Siebel Data Model. This individual would be responsible for identifying the actual data to be loaded into the EIM tables and to make sure that the physical layout of the database provides optimal performance. This team member would also be responsible for the crucial task of mapping the data into the Siebel base tables. For more information on performing this task, see "Mapping Data into Siebel Applications" on page 144.
  - A system administrator with a strong background in the systems (both the database server and application server) used by your organization. This individual would be responsible for developing scripts unique to your organization to automate the loading of data into the EIM tables, and to execute EIM in order to process the data into the Siebel base tables.
  - A business analyst with a strong understanding of the Siebel Data Model and its intended usage in the Siebel implementation. This individual would act as a liaison between the business and technical members of the EIM team.

Mapping Data into Siebel Applications

EIM uses EIM table mappings to map columns from EIM tables to Siebel base tables. Siebel predefined EIM mappings are fixed and cannot be remapped.

NOTE: EIM uses only EIM table mappings to determine table relationships. EIM does not use configuration logic in the Siebel repository to determine table relationships.

Using Siebel Tools, you can view:

- EIM table mappings to Siebel base tables
- Column mappings to Siebel base table columns
- Siebel base table mappings to EIM tables

Some base tables may not be mapped to a corresponding EIM table. In such cases, use Siebel Visual Basic (VB) to load data into these base tables and inform Siebel Technical Services regarding the missing mapping. For information on using Siebel VB, see *Siebel VB Language Reference*.

If you have licensed Database Extensibility and created extensions, you can use the Column Mapping screen to specify mappings to the new fields. Database extensibility and EIM support mappings between columns in extension tables and EIM tables only if these columns share the same base table. To map EIM table extensions to base table extensions, you must specify which column the extended field will point to in the base table. For more information on Database Extensibility, see *Configuring Siebel eBusiness Applications*.

To map data into a Siebel application

1. Determine which Siebel base table columns need to be populated for the Siebel implementation, along with the external data that will be loaded into these base tables.
2. Determine which EIM table and columns will be used to import from the source to the destination.
3. Analyze this external data to determine which attributes need to be stored and the relationship this data has to other entities.

   To facilitate this, you can request an EIM Data Mapping and Design review from Siebel Expert Services. This review can be used to make sure that the EIM mappings are correct and will accomplish intended goals.

Testing EIM Processes

This issue, fully and completely testing the EIM processes, tends to be overlooked. Testing is more than simply mapping the data and then running an EIM process using the default EIM configuration file. Complete testing requires you to run a large number of identical EIM jobs with similar data. This allows you to not only find any areas that you may have overlooked, but it also provides some insight into the optimal sizing of the EIM batches and exposure to scenarios that may occur in a production environment.

Before using EIM, a database administrator must populate the EIM tables with data to be processed by EIM. Then, you can invoke EIM to process this data, with EIM making multiple passes through the tables to complete the specified process.

EIM reads a special configuration file that specifies the EIM process to perform (import, merge, delete, or export) and the appropriate parameters. The EIM configuration file (the default file is default.ifb) is an ASCII text file of extension type .IFB that resides in the admin subdirectory under the Siebel server directory. Before running an EIM process, you must edit the contents of the EIM configuration file to define the processes that EIM will perform.

The EIM log file can contain information at different levels of detail depending on the values of three flags—the Error flag, the SQL flag, and the Trace flag. For more information on these flags, see "Viewing the EIM Log File" on page 128. Some of the recommended settings are described in the following list:

- As a starting point, it is recommended to set the Error Flag=1, the SQL flag = 1, and the Trace Flag=1. This setting will show errors and unused foreign keys. The setting Trace Flags=1 will provide a summary (after each batch) of the elapsed time in Step 10 on page 55 and Step 11 on page 55.
- Set Error flag = 1, SQL flag = 8, and Trace flag = 3. These settings will produce a log file with SQL statements that include how long each statement took, which is useful for optimizing SQL performance.
- Set Error flag = 0, SQL flag = 0, and Trace flag = 1. These settings will produce a log file showing how long each EIM step took, which is useful when figuring out the optimal batch size as well as monitoring for deterioration of performance in a particular step.

General Guidelines for Optimizing EIM

The following guidelines are recommended for improving EIM performance:

- Verify that all indexes exist for the tables involved. Keep in mind, however, that for large loads you should drop most of the indexes from the target tables to increase the speed of the process, rebuilding those indexes afterward when the process is finished.
- Limit tables and columns to be processed using ONLY BASE TABLES/COLUMNS configuration parameters to minimize EIM processing.
- Consider disabling the Docking: Transaction Logging system preference during the EIM run. Switching off transaction logging improves performance; however, this benefit must be balanced with the need for mobile users to reextract afterward. To disable transaction logging, complete Step 2 on page 58.
- Altering batch sizes to find the optimal batch size for a given business component typically helps resolve performance issues. The batch size is dependent upon the quantity of data and which type of EIM process you are running.

  NOTE: Although the limit of rows you can process is directly related to the capabilities of your database server, executing batches greater than 100,000 rows is strongly discouraged.
- For EIM delete processes that use the DELETE EXACT parameter, use a batch size of 20,000 rows or less.
- Try using batch ranges (BATCH = x–y). This allows you to run with smaller batch sizes and avoid the startup overhead on each batch. The maximum number of batches that you can run in an EIM process is 1,000.
- Perform regular table maintenance on EIM tables. Frequent insert or delete operations on EIM tables can cause fragmentation. Consult your database administrator to detect and correct fragmentation in the EIM tables.
- Delete batches from EIM tables on completion. Leaving old batches in the EIM table wastes space and could adversely affect performance.
- Run independent EIM jobs in parallel. For more information, see "Parallel Processing" on page 137.
- Set the USING SYNONYMS parameter to FALSE in the .IFB file to indicate that account synonyms do not need to be checked.

- If no other strategy appears to be successful, use the SQLPROFILE parameter to identify slow-running steps and queries. For more information, see "Using the SQLPROFILE Parameter" on page 153.

Recommended Sequence for Implementing EIM Processes

The following sequence is recommended for implementing EIM processes:

1. Customize and test the .IFB file to meet the business requirements.
2. Tune the .IFB parameters.
3. Separate the EIM processes.
4. Set the database parameters, making sure the basic requirements are met, including the hardware, the settings, and no or minimal fragmentation.

Before you start optimizing EIM processes, make sure there are no network problems or server performance problems that can affect the results. Siebel Expert Services recommends using at least 100 MB network segments and network-interface cards (NICs) to connect the Siebel server and Siebel database server. In addition, Siebel Expert Services recommends using a network switch or similar technology, rather than a hub, to maximize throughput.

Optimizing the .IFB File

When you have finished coding and testing the .IFB file to meet your business requirements, the next step is to optimize the .IFB file. The selected parameters in each section of the .IFB file determine the focus of each EIM task. The following recommendations are provided for each section of the .IFB file:

- **ONLY BASE TABLES** or **IGNORE BASE TABLES**. These parameters specify and restrict the selected base tables for the EIM process. A single EIM table (sometimes referred to as an interface table) is mapped to multiple user or base tables. For example, the table EIM_ACCOUNT is mapped to S_PARTY, S_ORG_EXT, and S_ADDR_ORG, as well as others. The default configuration is to process all base tables for each EIM table.

  NOTE: Siebel Expert Services strongly recommends that you always include these parameters in every section of the .IFB file, and list only those tables and columns that are relevant for a particular EIM task.

- **ONLY BASE COLUMNS** or **IGNORE BASE COLUMNS**. These parameters specify and restrict the selected base columns for the EIM process. The default is to process all base columns for each base table. It is likely that you are not using every column in a base table, and these parameters will make sure that EIM is only processing the desired columns in the table. You will see an additional performance increase if you exclude those columns that are defined as foreign keys (FKs) and are not used by the Siebel configuration; this is because EIM does not need to perform the interim processing (using SQL statements) to resolve the values for these FKs. Set the EIM Task parameter Error Flags = 1 to see which FKs are failing to be resolved by EIM (you may have missed excluding that FK with this parameter).

  NOTE: Do not use the IGNORE BASE COLUMNS parameter for merge processes or export processes. This parameter should only be used for import processes and delete processes.

Checking .IFB File Optimization

One method to find out if the .IFB file is optimized is to check the status of the records being processed in the EIM tables. This indicates if there are tables or columns that are being processed unnecessarily. The following query can be used to check the status of records in an EIM table:

```
select count(*), IF_ROW_STAT from <EIM Table>

where IF_ROW_BATCH_NUM = ?

group by IF_ROW_STAT;
```

If many rows have a status of PARTIALLY IMPORTED it is likely that further tuning can be done by excluding base tables and columns that are not necessary. For example, two tests were run to IMPORT 5000 accounts from EIM_ACCOUNT table. The first test included all of the base tables while the second test only focused on the four necessary tables by including the following line in the .IFB file:

```
ONLY BASE TABLES = S_ORG_EXT, S_ADDR_ORG, S_ACCNT_POSTN, S_ORG_TYPE
```

The first test took 89 minutes to import (excluding the Updating Primaries step), while the second test only took 2 minutes to import (excluding the Updating Primaries step).

Separating EIM Processes by Operation

Wherever possible, divide the EIM batches into insert-only transactions and update-only transactions. For example, assume that you are loading 50,000 records into an EIM table as part of a weekly process. 10,000 records represent new data and 40,000 records represent updates to existing data. By default, EIM can determine which records are to be added and which records are to be updated in the base tables, however, EIM will need to perform additional processing (through SQL statements) to make these determinations. If you were able to divide the 50,000 records into different batch numbers based on the type of transaction, you could avoid this additional processing. In addition, the columns being processed as part of the update activity might be less than those for the insert activity (resulting in an additional performance increase). To illustrate this, the .IFBs in the preceding example can be coded with the following sections:

- .IFB for mixed transactions:

  ```
  [Weekly Accounts]
  ```

```
TYPE = IMPORT
BATCH = 1-10
TABLE = EIM_ACCOUNT
ONLY BASE TABLES = S_ORG_EXT
IGNORE BASE COLUMNS = S_ORG_EXT.?
```

- .IFB for separate insert or update transactions:

```
[Weekly Accounts - New]
TYPE = IMPORT
BATCH = 1-2
TABLE = EIM_ACCOUNT
ONLY BASE TABLES = S_ORG_EXT
IGNORE BASE COLUMNS = S_ORG_EXT.?
INSERT ROWS = TRUE
UPDATE ROWS = FALSE
[Weekly Accounts - Existing]
TYPE = IMPORT
BATCH = 3-10
TABLE = EIM_ACCOUNT
ONLY BASE TABLES = S_ORG_EXT
ONLY BASE COLUMNS = S_ORG_EXT.NAME, S_ORG_EXT.LOC, S_ORG_EXT.?
INSERT ROWS = FALSE
UPDATE ROWS = TRUE
```

Troubleshooting EIM Performance

Before troubleshooting EIM performance, verify that there are no performance bottlenecks on the Siebel server or network.

Optimizing SQL for EIM

During this process, you need to be able to run several similar batches. If you do not have enough data with which to experiment, you may need to back up and restore the database between runs, so that you can continue processing the same batch.

First, you should run an EIM job with the following flag settings: Error flag = 1, SQL flag = 8, and Trace flag = 3. This will produce a log file that contains SQL statements and shows how long each statement took. Identify SQL statements that are taking too long (on a run of 5000 rows in a batch, look for statements that took longer than one minute). These are the statements that you want to concentrate on, and you should consult an experienced database administrator at this point. The process of optimizing the SQL for EIM involves the following:

- Use the respective database vendor's utility or a third-party utility to analyze the long-running SQL statements.
- Based on the review of the data access paths, review the SQL statements for proper index usage. There may be cases where an index is not used at all or the most efficient index is not being chosen. This may require a thorough analysis.
- Based on this analysis, use a systematic approach to tuning these long-running statements. You should perform one change at a time and then measure the results of the change by comparing them to the initial benchmarks. For example, you may find that dropping a particular index to improve the performance of one long-running statement might negatively impact the performance of other SQL statements. The decision on whether to drop the index should be based on the impact to the overall process as opposed to the individual long-running SQL statement. For this reason, it is important that one change be implemented at a time in order to measure the impact of the change.
- After repetitively going through and optimizing each long-running SQL statement, the focus can be shifted to other tuning measures, such as increasing the number of records processed in the EIM table at a time and the running of parallel EIM tasks.

Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters

Perform testing with the .IFB file parameters USE INDEX HINTS and USE ESSENTIAL INDEX HINTS, trying both settings (TRUE and FALSE). The default value for USE INDEX HINTS is FALSE. The default value for USE ESSENTIAL INDEX HINTS is TRUE.

NOTE: If your configuration file has more than one process section, you must specify USE INDEX HINTS within each one.

If these parameters are set to FALSE, EIM does not generate hints during processing. By setting the value to FALSE, you may realize performance gains if the TRUE setting means that hints are being generated that direct the database optimizer to use less than optimal indexes. EIM processing should be tested with both the TRUE and FALSE settings to determine which one provides better performance for each of the respective EIM jobs.

NOTE: The USE INDEX HINTS parameter is only applicable for Oracle database platforms. The USE ESSENTIAL INDEX HINTS parameter is only applicable for Microsoft SQL Server and Oracle database platforms.

These two parameters work for different queries, so you need to enable both to get all of the index hints on Oracle.

Further information on USE INDEX HINTS and USE ESSENTIAL INDEX HINTS is provided as follows:

Example: Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters

The following example illustrates the results achieved for an SQL statement with index hints and without index hints. This example was performed on the MS SQL Server platform.

SQL User Name	CPU	Reads	Writes	Duration	Connection ID	SPID
SADMIN	549625	**38844200**	141321	**626235**	516980	9

```
UPDATE dbo.S_ASSET5_FN_IF

SET T_APPLDCVRG__RID =

(SELECT MIN(BT.ROW_ID)

FROM dbo.S_APPLD_CVRG BT (INDEX = S_APPLD_CVRG_U2)

WHERE (BT.COVERAGE_CD = IT.CVRG_COVERAGE_CD AND

BT.TYPE = IT.CVRG_TYPE AND

BT.ASSET_ID = IT.T_APPLDCVRG_ASSETI AND

(BT.ASSET_CON_ID = IT.T_APPLDCVRG_ASSETC OR

(BT.ASSET_CON_ID IS NULL AND IT.T_APPLDCVRG_ASSETC IS NULL)) AND

(BT.INSITEM_ID = IT.T_APPLDCVRG_INSITE OR

(BT.INSITEM_ID IS NULL AND IT.T_APPLDCVRG_INSITE IS NULL))))

FROM dbo.S_ASSET5_FN_IF IT

WHERE (CVRG_COVERAGE_CD IS NOT NULL AND

CVRG_TYPE IS NOT NULL AND

T_APPLDCVRG_ASSETI IS NOT NULL AND

IF_ROW_BATCH_NUM = 10710001 AND

IF_ROW_STAT_NUM = 0 AND

T_APPLDCVRG__STA = 0)

SET STATISTICS PROFILE ON

GO

SET STATISTICS IO ON

GO
```

```
select

(SELECT MIN(BT.ROW_ID)

FROM dbo.S_APPLD_CVRG BT (INDEX = S_APPLD_CVRG_U2)

WHERE (BT.COVERAGE_CD = IT.CVRG_COVERAGE_CD AND

BT.TYPE = IT.CVRG_TYPE AND

BT.ASSET_ID = IT.T_APPLDCVRG_ASSETI AND

(BT.ASSET_CON_ID = IT.T_APPLDCVRG_ASSETC OR

(BT.ASSET_CON_ID IS NULL AND IT.T_APPLDCVRG_ASSETC IS NULL)) AND

(BT.INSITEM_ID = IT.T_APPLDCVRG_INSITE OR

(BT.INSITEM_ID IS NULL AND IT.T_APPLDCVRG_INSITE IS NULL))))

FROM dbo.S_ASSET5_FN_IF IT

WHERE (CVRG_COVERAGE_CD IS NOT NULL AND

CVRG_TYPE IS NOT NULL AND

T_APPLDCVRG_ASSETI IS NOT NULL AND

IF_ROW_BATCH_NUM = 10710001 AND

IF_ROW_STAT_NUM = 0 AND

T_APPLDCVRG__STA = 0)
```

With hints:

```
Table 'S_APPLD_CVRG'. Scan count 1, logical reads 394774, physical reads 0, read-ahead
reads 280810.

   Table 'S_ASSET5_FN_IF'. Scan count 1, logical reads 366, physical reads 0, read-ahead
   reads 0.
```

Without hints:

```
Table 'S_APPLD_CVRG'. Scan count 1268, logical reads 10203, physical reads 697, read-
ahead reads 0.

   Table 'S_ASSET5_FN_IF'. Scan count 1, logical reads 366, physical reads 0, read-ahead
   reads 0.
```

USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: EIM Criteria for Passing Indexes to the Database

This topic explains how EIM determines which indexes to include on the hint clause passed to the database when using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS parameters. When determining which indexes to pass on to the database as index hints, EIM takes the following steps:

1. Before generating a query, EIM makes a list of columns for which it has determined that an index is needed.
2. EIM then checks all of the indexes in the repository to find the index with the most matching columns.

EIM uses the following selection criteria in choosing indexes:

- Unique indexes have priority over non-unique indexes.
- Required columns have priority over non-required columns.

If a new index is created and it is declared in the repository, then there is a chance that EIM will choose it and pass it to the database on a hint.

NOTE: Regarding the behavior of the Oracle optimizer, when you specify an INDEX hint, the optimizer knows that it must use the index specified in the hint. In this case, the optimizer does not perform a full table scan, nor does it use any other index.

Using the SQLPROFILE Parameter

The inclusion of this parameter greatly simplifies the task of identifying the most time-intensive SQL statements. By inserting the following statement in the header section of the .IFB file, the most time-intensive SQL statements will be placed in the file:

```
SQLPROFILE = c:\temp\eimsql.sql
```

Below is an example of the file eimsql.sql:

<Start of the file – list of most time-intensive queries>

```
EIM: Integration Manager v6.0.1.2 [2943] ENU SQL profile dump (pid 430).

********************************************************************************

Top 34 SQL statements (of 170) by total time:

Batch Step Pass Total Rows Per Row What

-------- -------- -------- -------- -------- -------- -----------------

106 10 401 1334.48 5000 0.27 update implicit primaries to child

106 9 114 242.56 5000 0.05 copy
```

<…list of queries continues>

<Statistics by step and by pass>

```
********************************************************************************

Statements per step by total time:

Step Stmts Total Min Max Avg %

-------- -------- -------- -------- -------- -------- --------
```

```
    10 15 2627.27 0.00 1334.48 175.15 83.73
    9 11 329.52 0.00 242.56 29.96 10.50
```

<…list of statistics continues>

<SQL statements>

```
    ********************************************************************************
    batch 106, step 10, pass 401: "update implicit primaries to child":
    (total time 22:14m (1334s), 5000 rows affected, time/row 0.27s)
    UPDATE siebel.S_CONTACT BT
    SET PR_BL_PER_ADDR_ID =
    (SELECT VALUE(MIN(ROW_ID), 'No Match Row Id')
    FROM siebel.S_ADDR_PER CT
    WHERE (CT.PER_ID = BT.ROW_ID)),
    LAST_UPD = ?,
    LAST_UPD_BY = ?,
    MODIFICATION_NUM = MODIFICATION_NUM + 1
    WHERE (ROW_ID IN (
    SELECT T_ADDR_PER_PER_ID C1
    FROM siebel.EIM_CONTACT
    WHERE(
    T_ADDR_PER_PER_ID IS NOT NULL AND
    IF_ROW_BATCH_NUM = 106 AND
    T_ADDR_PER__STA = 0 AND
    T_ADDR_PER__EXS = 'N' AND
    T_ADDR_PER__UNQ = 'Y' AND
    T_ADDR_PER__RID IS NOT NULL)
    GROUP BY T_ADDR_PER_PER_ID)
    AND
    (PR_BL_PER_ADDR_ID IS NULL OR PR_BL_PER_ADDR_ID = 'No Match Row Id'))
    ********************************************************************
```

<…list of SQL statements continues>

Additional Indexes on EIM Tables

An examination of the data access path will assist you in determining whether additional indexes are necessary to improve the performance of the long-running SQL. In particular, look for table scans and large index range scans. For example, the following index was implemented to improve Step 10 on page 55. After evaluating the inner loop of the nested select, it was recommended to add an index on all T2 columns:

Inner loop:

```
(SELECT MIN(ROW_ID)
FROM siebel.EIM_ACCOUNT T2
WHERE (T2.T_ADDR_ORG__EXS = 'Y' AND
    T2.T_ADDR_ORG__RID = T1.T_ADDR_ORG__RID AND
    T2.IF_ROW_BATCH_NUM = 105 AND
    T2.IF_ROW_STAT_NUM = 0 AND
    T2.T_ADDR_ORG__STA = 0))
```

The index was created to consist of T2 columns used in the WHERE clause with ROW_ID at the end of the index. This influenced the database optimizer to choose this index for index-only access. Since the query wants the minimum (ROW_ID), the very first qualifying page in the index will also contain the lowest value.

NOTE: Having the ROW_ID column as the leading index column would also be a good strategy. Since the ROW_ID is unique, the index is likely to be more selective.

Adding Indexes to Improve Performance of S_ORG_EXT

Table S_ORG_EXT has indexes on many columns, but not all columns. If you have a large number of records (several million accounts) in S_ORG_EXT, you may get a performance improvement in deleting and merging by adding an index to one or more of the following:

- PR_BL_OU_ID
- PR_PAY_OU_ID
- PR_PRTNR_TYPE_ID
- PR_SHIP_OU_ID

Before implementing any additional indexes, first discuss this with qualified support personnel.

Creating Proper Statistics on EIM Tables

Use of the .IFB file parameter UPDATE STATISTICS is only applicable to the DB2 database platform. This parameter can control whether EIM dynamically updates the statistics of EIM tables. The default setting is TRUE. This parameter can be used to create a set of statistics on the EIM tables that you can save and then reapply to subsequent runs. After you have determined this optimal set of statistics, you can turn off the UPDATE STATISTICS parameter in the .IFB file (*UPDATE STATISTICS = FALSE*) thereby saving time during the EIM runs.

To determine the optimal set of statistics, you need to run several test batches and RUNSTATS commands with different options to see what produces the best results.

Before and after each test, you should execute *db2look utility* in mimic mode to save the statistics from the database system catalogs. For example, if you are testing EIM runs using EIM_CONTACT1 in database SIEBELDB, the following command generates UPDATE STATISTICS commands in the file EIM_CONTACT1_mim.sql:

```
db2look -m -a -d SIEBELDB -t EIM_CONTACT1 -o
EIM_CONTACT1_mim.sql
```

The file EIM_CONTACT1_mim.sql contains SQL UPDATE statements to update database system catalog tables with the saved statistics.

You can experiment with running test EIM batches after inserting the RUNSTATS commands provided in "DB2 Version 6/7 Options" and "DB2 Version 8 Options." After you find the set of statistics that works best, you can apply that particular mim.sql file to the database.

NOTE: Do not forget to save statistics with *db2look* between runs.

DB2 Version 6/7 Options

The following RUNSTATS commands can be used with DB2 versions 6 and 7:

```
db2 runstats on table SIEBELDB.EIM_CONTACT1 with distribution and detailed indexes
all shrlevel change

db2 runstats on table SIEBELDB.EIM_CONTACT1 and indexes all shrlevel change

db2 runstats on table SIEBELDB.EIM_CONTACT1 with distribution and indexes all
shrlevel change

db2 runstats on table SIEBELDB.EIM_CONTACT1 and detailed indexes all shrlevel change
```

DB2 Version 8 Options

The syntax for DB2 V8 commands provides more options, as follows:

- *shrlevel change*
- *allow write access*
- *allow read access*

The clauses *allow read access* and *shrlevel change* provide the greatest concurrency.

Dropping Indexes in Initial Runs

Typically, the EIM initial load is a very database-intensive process. Each row that is inserted into the base table requires modifications on the data page and the index pages of all the affected indexes. However, most of these indexes are never used during an EIM run. Index maintenance is a very time-consuming process for most database managers and should be avoided as much as possible. Therefore, the goal is to determine any indexes that are unnecessary for EIM and that can be dropped for the durations of the EIM run. You can create these indexes later in batch mode by using parallel execution strategies available for the respective database platform. Using this approach can save a significant amount of time.

NOTE: Under normal operations, using parallel execution strategies is *not* recommended.

- **Target Table Indexing Strategy.** For a target base table (such as S_ORG_EXT) you only need to use the Primary Index (Px, for example P1), and the Unique Indexes (Ux, for example U1), and then drop the remaining indexes for the duration of the EIM import. Past experience has determined that the Fx and Mx indexes can be dropped after an extensive SQL analysis of sample EIM runs.
- **Non-target Table indexing Strategy.** For child tables (such as S_ADDR_ORG) you only need to use the Primary Index (Px), the Unique Indexes (Ux), and the Foreign Key Indexes (needed for setting primary foreign keys in the parent table). Past experience has determined that the Fx and Mx indexes can be dropped after an extensive SQL analysis of sample EIM runs.

NOTE: Testing should always be performed when dropping indexes (or adding indexes) to make sure that expected results are achieved.

Controlling the Size of Batches

After tuning the long-running SQL statements, further tests can be run to determine the optimal batch size for each entity to be processed. The correct batch size varies and is influenced by the amount of buffer cache available. Optimal batch ranges have been observed to range anywhere between 500 and 15,000 rows. You should run several tests with different batch sizes to determine the size that provides the best rate of EIM transactions per second. Using the setting Trace Flag = 1 while running EIM helps in this task because you are then able to see how long each step takes and how many rows were processed by the EIM process.

NOTE: You should also monitor this throughput rate when determining degradation in parallel runs of EIM.

Recommended Number of Rows for a Single Batch

For an initial load, you can use 30,000 rows for a large batch. For ongoing loads, you can use 20,000 rows for a large batch. You should not exceed 100,000 rows in a large batch.

Furthermore, for MS SQL and Oracle environments, you should limit the number of records in the EIM tables to those that are being processed. For example, if you have determined that the optimal batch size for your implementation is 19,000 rows per batch and you are going to be running eight parallel EIM processes, then you should have 152,000 rows in the EIM table. Under no circumstances should you have more than 250,000 rows in any single EIM table because this reduces performance.

The restrictions mentioned in the example above do not apply to DB2 environments. As long as an index is being used to access the EIM tables, the numbers of rows in the EIM tables does not matter in DB2 environments.

NOTE: The number of rows you can load in a single batch may vary depending on your physical machine setup and on which table is being loaded. To reduce demands on resources and improve performance, you should generally try to vary batch sizes to determine the optimal size for each entity to be processed. In some cases, a smaller batch size can improve performance. But for simpler tables such as S_ASSET, you may find that loads perform better at higher batch sizes than for more complex tables such as S_CONTACT.

Controlling the Number of Records in EIM Tables

You should determine the number of records that can reside at one time in an EIM table while still maintaining an acceptable throughput rate during EIM processing. One observed effect of increasing the number of records in an EIM table is reduced performance of EIM jobs. This is often caused by object fragmentation or full table scans and large index range scans.

NOTE: In a DB2 environment, EIM table size is not an important factor that impacts performance, because it is easy to correct table scans and non-matching index scans. So a large number of records in an EIM table is not likely to reduce performance in a DB2 environment.

After addressing any object fragmentation and after the long-running SQL statements have been tuned, it is likely that you can increase the number of records that can reside in the EIM tables during EIM processing. When loading millions of records, this can result in a significant time savings because it reduces the number of times that the EIM table needs to be staged with a new data set.

When performing large data loads (millions of records) it is recommended that you perform initial load tests with fewer records in the EIM table. For example, while identifying and tuning the long-running SQL, you should start with approximately 50,000 records. After tuning efforts are complete, you should run additional tests while gradually increasing the number of records. For example you can incrementally increase the number of records to 100,000, then 200,000, and so on until you have determined the optimal number of records to load.

Using the USING SYNONYMS Parameter

The USING SYNONYMS parameter controls the queries of account synonyms during import processing. This parameter is also related to the S_ORG_SYN table. When set to FALSE, this parameter saves processing time because queries that look up synonyms are not used. The default setting is TRUE. You should only set this parameter to FALSE when account synonyms are not needed.

Using the NUM_IFTABLE_LOAD_CUTOFF Extended Parameter

Setting this extended parameter to a positive value will reduce the amount of time taken by EIM to load repository information. This is because when you set this parameter to a positive value, only information for the required EIM tables is loaded. For more information on this parameter, see Chapter 4, "EIM Configuration File."

NOTE: While this parameter is especially important for merge processes, it can also be used for any of the other types of processes.

Here is an example of using this parameter while running on an NT application server from the server command line mode:

```
run task for comp eim server siebserver with config=account2.ifb,
ExtendedParams="NUM_IFTABLE_LOAD_CUTOFF=1", traceflags=1
```

Disabling the Docking: Transaction Logging Parameter

Typically, a disabled Docking: Transaction Logging setting is only used during initial data loads. Disable Docking: Transaction Logging is set from the System Preferences view within the Siebel application. This setting indicates whether or not the Siebel application will log transactions for the purpose of routing data to Siebel Mobile Web Clients.

The default for this parameter is FALSE. If there are no Siebel Mobile Web Clients, then the default setting should remain. If you have Siebel Mobile Web Clients, then this parameter must be set to TRUE in order to route transactions to the mobile clients. However, during initial data loads, you can set this parameter to FALSE to reduce transaction activity to the Siebel docking tables. After the initial loads are complete, set the parameter back to TRUE.

NOTE: For incremental data loads, Docking: Transaction Logging should remain set to TRUE if there are mobile clients. If this setting is changed for incremental data loads then you will need to perform a reextract of all of the mobile clients.

Disabling Triggers

Disabling database triggers, by removing them through the Server Administration screens, can also help improve the throughput rate. This can be done by running the Generate Triggers server task with both the REMOVE and EXEC parameters set to TRUE. Be aware that components such as Workflow Policies and Assignment Manager will not function for the new or updated data. Also, remember to reapply the triggers after completing the EIM load.

Running EIM Tasks in Parallel

Running EIM tasks in parallel is the last strategy you should adopt in order to increase the EIM throughput rate. In other words, do not try this until all long-running SQL statements have been tuned, the optimal batch size has been determined, the optimal number of records to be processed at a time in the EIM table has been determined, and the database has been appropriately tuned. Before running tasks in parallel, check the value of the Maximum Tasks parameter. This parameter can be found under Enterprise Component Definitions, Siebel Server Parameters, Server Component Parameters, and Task Parameters. This parameter specifies the maximum number of running tasks that can be run at a time for a service.

NOTE: UPDATE STATISTICS must be set to FALSE in the .IFB file when running parallel EIM tasks on the IBM DB2 platform.

Database Guidelines for Optimizing EIM

The following section describes EIM tuning tips for the database platforms supported by Siebel applications (DB2, MS SQL Server, and Oracle).

IBM DB2 UDB

- Use the IBM DB2 load replace option when loading EIM tables and, if possible, turn off table logging.
- Use separate tablespaces for EIM tables and the base tables.
- Use large page sizes for EIM and the larger base tables. Previous experience has determined that a page size of 16 KB or 32 KB provides good performance. The larger page sizes allow more data to be fitted on a single page and also reduces the number of levels in the index B-tree structures.
- Similarly, use large extent sizes for both EIM and the large base tables.
- Consider using DMS containers for all Siebel tablespaces. Using raw devices or volumes will further help to improve performance.
- Make sure that the tablespace containers are equitably distributed across the logical and physical disks and across the input/output (I/O) controllers of the database server.
- Use separate bufferpools for EIM tables and the target base tables. Since initial EIM loads are quite large and there are usually no online users, it is recommended to allocate a significant amount of memory to the EIM and the base table bufferpools.
- Reorganize the tables if data on disk is fragmented. Use the reorgchk utility with current statistics to find the fragmented tables or indexes.
- Periodically make sure that table and index statistics are collected. Do not use RUNSTATS with the DETAILED option.
- Use IBM DB2 snapshot monitors to make sure performance is optimal and to detect and resolve any performance bottlenecks.

- Log retain can be turned OFF during the initial load. However, you should turn it back on before moving into a production environment.
- For the EIM tables and the base tables involved, alter the tables to set them to VOLATILE. This makes sure that indexes are preferred over table scans.
- Consider the following settings for DB2 registry values:

Registry Value	Setting
DB2_CORRELATED_PREDICATES =	YES
DB2_HASH_JOIN =	NO
DB2_RR_TO_RS =	YES
DB2_PARALLEL_IO =	"*"
DB2_STRIPPED_CONTAINERS =	When using RAID devices for tablespace containers

- Consider the following settings for the DB2 database manager configuration parameters:

Registry Value	Setting
INTRA_PARALLEL =	NO (may be used during large index creation)
MAX_QUERYDEGREE =	1 (may be increased during large index creation)
SHEAPTHRES =	100,000 (depends upon available memory, SORTHEAP setting, and other factors)

- Consider the following settings for the database parameters:

Registry Value	Setting
CATALOGCACHE_SZ =	6400
DFT_QUERYOPT =	3
LOCKLIST =	5000
LOCKTIMEOUT =	120 (between 30 and 120)
LOGBUFSZ =	512
LOGFILESZ =	8000 or higher
LOGPRIMARY =	20 or higher
LOGRETAIN =	NO (only during initial EIM loads)
MAXLOCKS =	30
MINCOMMIT =	1
NUM_IOCLEANERS =	Number of CPUs in the database server

Registry Value	Setting
NUM_IOSERVERS =	Number of disks containing DB2 containers
SORTHEAP =	10240 (This setting is only for initial EIM loads. During production, set it to between 64 and 256.)
STAT_HEAP_SZ =	8000

MS SQL Server

The following sections describe EIM tuning tips for the MS SQL Server database platform.

Fixing Table Fragmentation

Table and index fragmentation occurs on tables that have many insert, update, and delete activities. Because the table is being modified, pages begin to fill, causing page splits on clustered indexes. As pages split, the new pages may use disk space that is not contiguous, hurting performance because contiguous pages are a form of sequential input/output (I/O), which is faster than nonsequential I/O.

Before running EIM, it is important to defragment the tables by executing the DBCC DBREINDEX command on the table's clustered index. This applies especially to those indexes that will be used during EIM processing, which packs each data page with the fill factor amount of data (configured using the FILLFACTOR option) and reorders the information on contiguous data pages. You can also drop and recreate the index (without using the SORTED_DATA option). However, using the DBCC DBREINDEX command is recommended because it is faster than dropping and recreating the index, as shown in the following example:

```
DBCC SHOWCONTIG scanning '**S_GROUPIF' table...

Table: '**S_GROUPIF' (731969784); index ID: 1, database ID: 7

TABLE level scan performed.

Pages Scanned................................: 739

Extents Scanned..............................: 93

Extent Switches..............................: 92

Avg. Pages per Extent........................: 7.9

Scan Density [Best Count:Actual Count].......: 100.00% [93:93]

Logical Scan Fragmentation ..................: 0.00%

Extent Scan Fragmentation ...................: 1.08%

Avg. Bytes Free per Page.....................: 74.8

Avg. Page Density (full).....................: 99.08%
```

```
DBCC execution completed. If DBCC printed error messages, contact the system
administrator.
```

To determine whether you need to rebuild the index because of excessive index page splits, look at the Scan Density value displayed by DBCC SHOWCONTIG. The Scan Density value should be at or near 100%. If it is significantly below 100%, rebuild the index.

Purging an EIM Table

When purging data from the EIM table, use the TRUNCATE TABLE statement. This is a fast, nonlogged method of deleting all rows in a table. DELETE physically removes one row at a time and records each deleted row in the transaction log. TRUNCATE TABLE only logs the deallocation of whole data pages and immediately frees all the space occupied by that table's data and indexes. The distribution pages for all indexes are also freed.

Parallel Data Load for EIM tables Using bcp

Microsoft SQL Server allows data to be bulk copied into a single EIM table from multiple clients in parallel, using the bcp utility or BULK INSERT statement. You should use the bcp utility or BULK INSERT statement when the following conditions are true:

- The SQL Server is running on a computer with more than one processor.
- The data to be bulk copied into the EIM table can be partitioned into separate data files.

These recommendations can improve the performance of data load operations. Perform the following tasks, in the order in which they are presented, to bulk copy data into SQL Server in parallel:

1. Set the database option truncate log on checkpoint to TRUE using sp_dboption.(*)
2. Set the database option select into/bulkcopy to TRUE using sp_dboption.

   In a logged bulk copy all row insertions are logged, which can generate many log records in a large bulk copy operation. These log records can be used to both roll forward and roll back the logged bulk copy operation.

   In a nonlogged bulk copy, only the allocations of new pages to hold the bulk copied rows are logged. This significantly reduces the amount of logging that is needed and speeds the bulk copy operation. Once you do a nonlogged operation you should immediately back up so transaction logging can be restarted.
3. Make sure that the table does not have any indexes, or if the table has an index, make sure it is empty when the bulk copy starts.
4. Make sure you are not replicating the target table.
5. Make sure the TABLOCK hint is specified using bcp_control with eOption set to BCPHINTS.

   NOTE: Using ordered data and the ORDER hint will not affect performance because the clustered index is not present in the EIM table during the data load.
6. After data has been bulk copied into a single EIM table from multiple clients, any clustered index on the table should be recreated using DBCC DBREINDEX.

TempDB

This is the database that Microsoft SQL Server uses for temporary space needed during execution of various queries. Set the initial size of the TEMPDB to a minimum of 100 MB, and configure it for auto-growth, which allows SQL Server to expand the temporary database as needed to accommodate user activity.

Configuration Parameters

Additional parameters have a direct impact on SQL Server performance and should be set according to the following guidelines:

- **SPIN COUNTER.** This parameter specifies the maximum number of attempts that Microsoft SQL Server will make to obtain a given resource. The default settings should be adequate in most configurations.
- **MAX ASYNC I/O.** This parameter configures the number of asynchronous inputs/outputs (I/Os) that can be issued. The default is 32, which allows a maximum of 32 outstanding reads and 32 outstanding writes per file. Servers with nonspecialized disk subsystems do not benefit from increasing this value. Servers with high-performance disk subsystems, such as intelligent disk controllers with RAM caching and RAID disk sets, may gain some performance benefit by increasing this value because they have the ability to accept multiple asynchronous I/O requests.
- **MAX DEGREE OF PARALLELISM.** This option is used to configure Microsoft SQL Server's use of parallel query plan generation. Set this option to 1 to disable parallel query plan generation. This setting is mandatory to avoid generating an unpredictable query plan.
- **LOCKS.** This option is used to specify the number of locks that Microsoft SQL Server allocates for use throughout the server. Locks are used to manage access to database resources such as tables and rows. This option should be set to 0 to allow Microsoft SQL Server to dynamically manage lock allocation based on system requirements.
- **AUTO CREATE STATISTICS.** This option allows SQL Server to create new statistics for database columns as needed to improve query optimization. This option should be enabled.
- **AUTO UPDATE STATISTICS.** This allows Microsoft SQL Server to automatically manage database statistics and update them as necessary to achieve proper query optimization. This option should be enabled.

Oracle Databases

This section provides EIM tuning tips for the Oracle database platform.

Avoiding Excessive Table Fragmentation

Before running EIM, you should consult with an experienced DBA in order to evaluate the amount of space necessary to store the data to be inserted in the EIM tables and the Siebel base tables. Also, for example with Oracle, you can make sure that the extent sizes of those tables and indexes are defined accordingly.

Avoiding excessive extensions and keeping a small number of extents for tables and indexes is important because extent allocation and disallocation activities (such as truncate or drop commands) can be demanding on CPU resources.

To check if segment extension is occurring in an Oracle database

- Use the SQL statement that follows to identify objects with greater than 10 extents.

  NOTE: Ten extents is not a target number for segment extensions.

```
SELECT segment_name,segment_type,tablespace_name,extents
    FROM dba_segments
    WHERE owner = (Siebel table_owner)
        and extents > 9;
```

To reduce fragmentation, the objects can be rebuilt with appropriate storage parameters. Always be careful when rebuilding objects because of issues such as defaults or triggers on the objects.

Purging an EIM Table

When purging data from an EIM table, use the TRUNCATE command as opposed to the DELETE command. The TRUNCATE command releases the data blocks and resets the high water mark while the DELETE command does not, which causes additional blocks to be read during processing. Also, be sure to drop and recreate the indexes on the EIM table to release the empty blocks.

Creating Indexes

When working with large volumes of data in EIM tables, index build time can be costly when refreshing an EIM table with a new data set. To improve the performance of the index build use the UNRECOVERABLE option (Oracle 7.3) or NOLOGGING (Oracle 8) option. This prevents the Oracle database from writing to the REDO LOG files. You can also improve index build time by creating multiple SQL scripts to create the indexes, and then by running these scripts in parallel through SQLPlus. The following section provides a sample SQL statement that demonstrates the syntax for using the UNRECOVERABLE or NOLOGGING options:

```
CREATE INDEX S_SAMPLE_M1 ON
S_SAMPLE (SAMPLE_ID)
TABLESPACE TS_INDX
STORAGE (INITIAL 10M NEXT 5M PCTINCREASE 0)
UNRECOVERABLE/NOLOGGING;
```

NOTE: The option you choose (UNRECOVERABLE or NOLOGGING) depends on the version of the Oracle database you are using.

Disabling Archive Logging

It is recommended that Archive Logging be disabled during initial data loads. You can enable this feature to provide for point-in-time recovery after completing the data loads.

FREELIST Parameter

Multiple EIM processes can be executed against an EIM table provided they all use different batches or batch ranges. The concern is that you may experience contention for locks on common objects. To run multiple jobs in parallel against the same EIM table, you should make sure that the FREELIST parameter is set appropriately for the tables and indexes used in the EIM processing.

This includes EIM tables and indexes, as well as base tables and indexes. The value of this parameter specifies the number of block IDs that will be stored in memory which are available for record insertion. Generally, you should set this to at least half of the intended number of parallel jobs to be run against the same EIM table (for example, a FREELIST setting of 10 should permit up to 20 parallel jobs against the same EIM table).

This parameter is set at the time of object creation and the default for this parameter is 1. To check the value of this parameter for a particular object, the following query can be used:

```
SELECT SEGMENT_NAME, SEGMENT_TYPE, FREELISTS

FROM DBA_SEGMENTS

WHERE SEGMENT_NAME='<OBJECT NAME TO BE CHECKED>';
```

To change this parameter, the object must be rebuilt. Again, be careful when rebuilding objects because of issues such as defaults or triggers on the objects.

To rebuild an object

1. Export the data from the table with the grants.
2. Drop the table.
3. Recreate the table with the desired FREELIST parameter.
4. Import the data back into the table.
5. Rebuild the indexes with the desired FREELIST parameter.

Caching Tables

Another method to improve performance is to put small tables that are frequently accessed in cache. The value of BUFFER_POOL_KEEP determines the portion of the buffer cache that will not be flushed by the LRU algorithm. This allows you to put certain tables in memory, which improves performance when accessing those tables. This also makes sure that after accessing a table for the first time, it will always be kept in the memory. Otherwise, it is possible that the table will get pushed out of memory and will require disk access the next time used. Be aware that the amount of memory allocated to the keep area is subtracted from the overall buffer cache memory (defined by DB_BLOCK_BUFFERS). A good candidate for this type of operation is the S_LST_OF_VAL table. The syntax for keeping a table in the cache is as follows:

```
ALTER TABLE S_LST_OF_VAL CACHE;
```

Updating Tables

When there are 255 or more NVL functions in an update statement, Oracle updates the wrong data due to hash keys overflow. This is an Oracle-specific issue. To avoid this problem, use less than 255 NVL functions in the update statement.

IBM DB2/390

For DB2 configuration settings, you can find a listing (from the JCL) of the Database Manager Configuration Parameters (DSNZPARM) in *Implementing Siebel eBusiness Applications on DB2 UDB for OS/390 and z/OS*.

Further IBM DB2 information is provided in the following sections:

- "IBM DB2 Loading Process for EIM" on page 167
- "General Recommendations for the IBM DB2 Loading Process" on page 168

IBM DB2 Loading Process for EIM

Figure 6 illustrates the load process for IBM DB2.

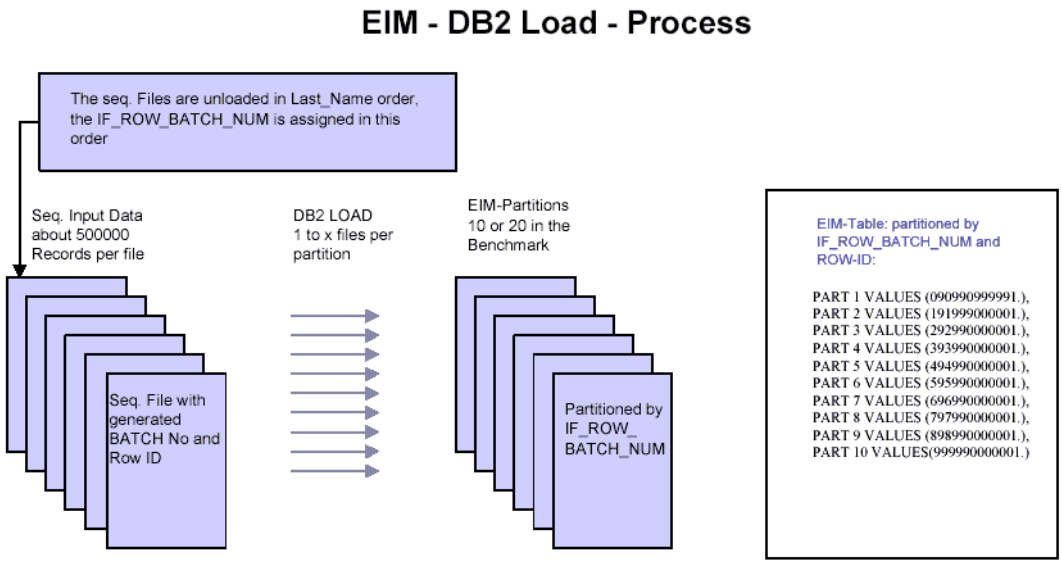


Figure 6. IBM DB2 Loading Process for EIM

For more information, see "General Recommendations for the IBM DB2 Loading Process" on page 168.

General Recommendations for the IBM DB2 Loading Process

The following general recommendations apply when performing the IBM DB2 loading process for EIM:

- Use the ONLY/IGNORE BASE TABLES parameters or ONLY/IGNORE BASE COLUMNS parameters in the .IFB files to reduce the amount of processing performed by EIM. By using the IGNORE BASE COLUMNS option, you allow foreign keys to be excluded, which reduces both processing requirements and error log entries for keys which cannot be resolved. Remember that the key words ONLY and IGNORE are mutually exclusive. For example, the following settings exclude the options IGNORE BASE TABLES and ONLY BASE COLUMNS:

  ```
  ONLY BASE TABLES = S_CONTACT

  IGNORE BASE COLUMNS = S_CONTACT.PR_MKT_SEG_ID
  ```

  The preceding example also causes the foreign key PR_MKT_SEG_ID to be forced to a nonmetal.

- Import parents and children separately. Wherever possible, load data such as accounts, addresses, and teams at the same time, using the same EIM table.
- Use batch sizes that allow all of the EIM table data in the batch to be stored in the database cache (approximately 2,000 records, 5000 for DB2/390). EIM can be configured through the use of an extended parameter to use a range of batches, you should remember to put the variable name into the .IFB file.
- Multiple EIM processes can be executed against an EIM table, provided they all use different batches or batch ranges. However, the main limit to EIM performance is not the application server but the database. Contention for locks on common objects may occur if multiple EIM streams are executed simultaneously for the same base table. Multiple EIM job streams can run concurrently for different base tables, for example, S_ORG_EXT and S_ASSET.
- Run EIM during periods of minimum user activity, outside of business hours, if possible. This reduces the load for connected users and makes sure that the maximum processing capacity is available for the EIM processes.
- Set the system preference (in Administration - Application > System Preferences) for Docking: Transaction Logging to FALSE during the initial database load. This reduces transaction activity to the Siebel docking tables, which are used for synchronizing mobile clients.
- Disable the database triggers by removing them through the Server Administration screens. Doing this can also help to improve the throughput rate. Remember to reapply the triggers after the EIM load has completed, because the lack of triggers will mean that components, such as Workflow Policies and Assignment Manager, will not function for the new or updated data.
- Remember to make sure that the required columns ROW_ID, IF_ROW_STAT, and IF_ROW_BATCH_NUM are correctly populated in the EIM table to be processed. The most efficient time to do this is when populating the EIM table from the data source or staging area, after cleansing the data.

- Unless there are specific processing requirements, make sure the EIM table is empty before loading data into it for EIM processing. Always make sure that suitable batch numbers are being used to avoid conflicts within the EIM table. If you are using an automated routine, truncating the EIM table between loads from the data source helps to preserve performance.
- When running Siebel applications on an IBM DB2 database, EIM can sometimes stop responding when updating the S_LST_OF_VAL base table. This is due to a data issue. The BU_ID column in the S_LST_OF_VAL base table may have only one or very few distinct values. That makes the DB2 optimizer perform a table scan through all rows in the S_LST_OF_VAL table when most or all rows have the same BU_ID column value.

  To avoid this problem and speed up the query, you should modify the statistics data by running the following SQL statements:

  ```
  update sysibm.sysindexes set firstkeycard=1000 where name='S_LST_OF_VAL_M2';

  update sysibm.syscolumns set colcard = 1000 where tbname='S_LST_OF_VAL' and
  name='BU_ID';
  ```

  NOTE: Depending on the data with which you are working, you may need to run other SQL statements beforehand.

Data Management Guidelines for Optimizing EIM

The following recommendations apply when performing the EIM loading process:

- The EIM mapping chart shows that many of the EIM table columns derive their values not from legacy database fields but from unvarying literal strings. Avoid filling up the EIM tables with this type of information, because it slows down the movement of real legacy data from the EIM tables to the base tables.
- EIM offers an alternative method for populating base table columns with unvarying literal strings, namely by using the DEFAULT COLUMN statement. This approach allows you to specify default literals that must be imported into the base tables without having to retrieve them from the EIM tables. For example, the EIM mapping chart shows Default Organization as the constant value for CON_BU in EIM_CONTACT, which in turn will move into BU_ID in S_CONTACT. The same result can be achieved with the setting `DEFAULT COLUMN = CON_BU, Default Value` in the .IFB file. There are many other opportunities for moving literal strings from the EIM tables to the .IFB file.

Run Parameter Guidelines for Optimizing EIM

The following recommendations are for setting run parameters when performing the EIM loading process:

- Do not set TRIM SPACES to FALSE. Using the TRIM SPACES parameter causes trailing spaces to be stored in the Siebel base table. This can lead to inefficient use of disk space since Siebel applications use VarChar on virtually all text columns longer than a single character. Setting TRIM SPACES to FALSE can also waste valuable bufferpool space for the tablespace data.
- Use either the IGNORE BASE TABLES parameter or the ONLY BASE TABLES parameter to limit the number of tables being inserted into or updated. The ONLY BASE TABLES parameter is preferable because the list is usually shorter and it is self-documenting. Using these parameters improves performance because it limits the number of tables EIM attempts to load and they also save space for tables that will not be used by the user interface.
- Use either the IGNORE BASE COLUMNS parameter or the ONLY BASE COLUMNS parameter to limit the number of tables being inserted into or updated. The ONLY BASE COLUMNS parameter is preferable because the list is usually shorter and it is self-documenting. Using these parameters improves performance because they limit the number of foreign keys EIM attempts to resolve.
- Set the USING SYNONYMS parameter to FALSE in the .IFB file. This logical operator indicates to EIM that account synonyms do not require processing during import, which reduces the amount of processing. Do not set the USING SYNONYMS parameter to FALSE if you plan to use multiple addresses for accounts. Otherwise, EIM will not attach addresses to the appropriate accounts.
- Suppress inserts when the base table is already fully loaded and the table is the primary table for an EIM table used to load and update other tables. The command format is `INSERT ROWS = <table name>, FALSE`.
- Suppress updates when the base table is already fully loaded and does not require updates such as foreign key additions, but the table is the primary table for an EIM table used to load and update other tables. The command format is `UPDATE ROWS = <table name>, FALSE`.

Monitoring the Siebel Server

When monitoring the Siebel server, the assumption is that you have allocated sufficient processor and memory resources for running the EIM task on the Siebel application servers and Siebel database servers.

If you are using Windows 2000 as the operating system for the Siebel server, the Windows Performance Monitor can be used to verify the amount of processor and memory being used by the hardware.

If you are using Sun Solaris or IBM AIX as operating systems for the Siebel Server, you can use *vmstat* and *iostat* to verify the amount of processor and memory being used by the hardware.

A EIM: Examples of Common Usage

This appendix provides examples that illustrate the usage of Siebel EIM. The information is organized as follows:

EIM Import Process Examples

This section provides usage examples that can be applied to your running of import processes.

Example of Importing from Multiple EIM Tables in a Single .IFB File

You use shell processes to import multiple EIM tables in a single .IFB file. In the sample .IFB file that follows, first EIM_CONTACT is imported, then EIM_ACCOUNT is imported.

```
[Siebel Interface Manager]

   PROCESS = Import Contacts and Accounts

[Import Contacts and Accounts]

   TYPE = SHELL

   INCLUDE = "Import Contacts"

   INCLUDE = "Import Accounts"

[Import Contacts]

   TYPE = IMPORT

   TABLE = EIM_CONTACT

   BATCH = 100

[Import Accounts]

   TYPE = IMPORT

   TABLE = EIM_ACCOUNT
```

```
BATCH = 200
```

Example of Updating a Table in a One-to-One Relationship with Its Parent

To update a table that has a one-to-one relationship with its parent table, make sure that the EIM table has only one record matching the user key of the target table.

For example, to update column values in S_ORG_EXT_X using EIM_ACCNT_DTL, there can be only one record in EIM_ACCNT_DTL that matches the user key of the S_ORG_EXT_X table. If more than one record with the same user key is inserted into this EIM table, then EIM might select the wrong record for update, and update IF_ROW_STAT with DUP_RECORD_EXISTS for the rest of the records.

Example of Updating Columns When There Are Two Records with the Same User Key in a Single Batch

EIM does not update columns in the following scenario: you have two records with same user key in the same batch, but with different nonuser keys to be updated.

This cannot be done because there is no way for EIM—which runs set-based operations—to know which record updates which of the non-user keys in one batch. EIM chooses the row with MIN(ROW_ID) and marks the other rows as duplicates.

To perform this kind of update, for which you are updating a record more than twice, you must run two different batches.

Example of Updating Columns When There Are Two Non-Target Base Tables Mapped to One EIM Table

If there are two non-target base tables mapped to one EIM table and one of the non-target base tables has a foreign key pointing to the other one, the data in the same row of the EIM table cannot be inserted into these two tables in one batch or one session. In cases like this, run EIM twice.

The reason is that in Step 4 on page 54, the non-target parent base table is queried to resolve the foreign key of the non-target child base table, but the new row in the parent table has not been inserted.

Take for example, the EIM_CONTACT mapping in 7.5 Siebel Industry Applications. You cannot insert the data in one EIM_CONTACT row into S_PARTY, S_CONTACT, S_ADDR_PER, and S_CON_ADDR simultaneously, because the foreign key of the non-target child base table S_CON_ADDR is pointing to the non-target parent table S_ADDR_PER.

In the first run, a new row will be created in S_PARTY, S_CONTACT, S_ADDR_PER. In the second run, a new row will be inserted in S_CON_ADDR.

Example of Importing Primary Keys

In order to import a primary column, you must populate the following interface columns:

- These interface columns:
  - ROW_ID
  - IF_ROW_BATCH_NUM
  - IF_ROW_STAT
- The interface columns that map to the user key columns of the EIM table's target base table
- The interface columns that map to the user key columns of the primary column's base table
- The primary flag interface column that maps to the primary base column
- The interface columns that map to the primary's intersection table

The intersection row must exist before setting the primary. If you want to import the intersection row and set it as the primary at the same time, you must also populate the interface columns that map to the intersection table's required columns.

For example:

If you want to update the S_ORG_EXT.PR_POSTN_ID primary column with the EIM_ACCOUNT interface table, you must populate:

- The interface columns:
  - ROW_ID
  - IF_ROW_BATCH_NUM
  - IF_ROW_STAT
- The interface columns that map to the user keys of the S_PARTY table (EIM_ACCOUNT's target base table):
  - PARTY_UID
  - PARTY_TYPE_CD
- The interface columns that map to the user keys of the S_ORG_EXT table:
  - NAME
  - LOC
  - ACCNT_BU
- The primary flag interface column that maps to S_ORG_EXT.PR_POSTN_ID:
  - ACC_PR_POSTN
- The interface columns that map to the S_ACCNT_POSTN table (S_ORG_EXT.PR_POSTN_ID primary's intersection table):
  - NAME
  - LOC

- ACCNT_BU
- POSTN_NAME
- POSTN_DIVN
- POSTN_LOC
- POSTN_BU

NOTE: You can find the S_ORG_EXT.PR_POSTN_ID primary's intersection table using Siebel Tools. In Table, query and select S_ORG_EXT > Column, then query and select PR_POSTN_ID > Primary Inter Table Name property value.

The following are .IFB settings that you can use when running an EIM task that populates an EIM table to update a S_ORG_EXT row's PR_POSTN_ID primary position to reference the S_POSTN row:

```
[Siebel Interface Manager]

USER NAME = "SADMIN"

PASSWORD = "<SADMIN's password>"

RUN PROCESS = Update S_ORG_EXT.PR_POSTN_ID

[Update S_ORG_EXT.PR_POSTN_ID]

TYPE = IMPORT

BATCH = 1

TABLE = EIM_ACCOUNT

ONLY BASE TABLES = S_PARTY, S_ORG_EXT, S_ACCNT_POSTN

INSERT ROWS = S_PARTY, FALSE

UPDATE ROWS = S_PARTY, FALSE

INSERT ROWS = S_ORG_EXT, FALSE

ONLY BASE COLUMNS = S_PARTY.PARTY_UID, \

S_PARTY.PARTY_TYPE_CD, \

S_ORG_EXT.NAME, \

S_ORG_EXT.LOC, \

S_ORG_EXT.BU_ID, \

S_ORG_EXT.PR_POSTN_ID, \

S_ACCNT_POSTN.OU_EXT_ID, \

S_ACCNT_POSTN.POSITION_ID
```

There are some cases that require you to include the MISC SQL parameter to set the primaries. For more information, see "MISC SQL Parameter" on page 71.

Example of Setting a Primary

As one example of setting a primary, you can populate the PR_PROD_LN_ID column in the S_PROD_INT base table by completing the following procedure:

To populate the PR_PROD_LN_ID column in the S_PROD_INT base table

1. Populate the S_PROD_INT base table using the EIM_PROD_INT interface table.
2. Populate the S_PROD_LN base table using the EIM_PROD_LN interface table.
3. Populate S_PROD_LN_PROD using EIM_PROD_INT1 and specifying the primary product lines by setting PROD_PR_PROD_LN to Y.

Visibility of Fields: Example of Importing Party Objects

Loading of party objects affects visibility of fields. You should be aware that, in most cases, an organization table should be populated along with the party object table.

For example, when a user clicks the Account field to open the MVG applet in the Contact form applet, the Account field disappears and returns to a null value after the EIM process is run.

This is because there is an association between Contacts and Accounts that is stored in the intersection table S_PARTY_PER. So to establish this relationship, you should fill in the columns for only the S_PARTY, S_CONTACT, and S_PARTY_PER table.

Visibility of Fields: Example of Importing Accounts

This example is specific to Siebel Industry Applications.

To view all accounts, the data must be inserted into the S_PARTY, S_ACCNT_POSTN, S_ORG_EXT, and S_ORG_BU tables, as well as other relevant tables.

NOTE: S_ORG_BU is a table that is new in Siebel 7. This table must be populated for visibility in the All Accounts view.

To insert the data into the required tables, you can use the EIM_ACCNT_CUT and EIM_ACCOUNT interface tables. Make sure the values in the OU_NUM and MASTER_OU_ID columns of the S_ORG_EXT base table are populated.

In Siebel Industry Solutions (SIS) version 7.0.x and Siebel Industry Applications (SIA) version 7.5.x, there is no mapping in the EIM_ACCNT_CUT interface table to the S_ORG_BU table. However, the EIM_ACCOUNT and EIM_ORG_BU interface tables are mapped to S_ORG_BU. You can use EIM_ACCOUNT and EIM_ORG_BU to populate S_ORG_BU.

In SIS and SIA, MASTER_OU_ID in S_ORG_EXT must be populated for visibility in any of the Accounts views. If S_ORG_EXT.MASTER_OU_ID is not populated, the imported accounts will be visible only in the Data Administration >Accounts/Orgs view. The imported accounts will not be visible in the Data Administration >Accounts view or any other view including My Accounts, All Accounts, and All Accounts Across Organizations.

NOTE: When loading account addresses, make sure to set an explicit primary. The default setting is implicit, which means that primaries are not set until a record is retrieved in the application. This can cause queries, such as on the State field, to return incomplete or inconsistent data. For more information, see "About Explicit Primary Mappings" on page 24.

The sample .IFB file that follows can be used for importing accounts. The account visibility depends on S_ORG_BU to resolve the organization and S_ACCT_POSTN for the position.

```
[Siebel Interface Manager]

      USER NAME = "SADMIN"

      PASSWORD = "SADMIN"

      PROCESS = Import Account

[Import Account]

      TYPE = IMPORT

      BATCH = 555

      TABLE = EIM_ACCOUNT

ONLY BASE TABLES = S_PARTY, S_ACCNT_POSTN,  S_ORG_EXT, S_ORG_BU

DEFAULT COLUMN = ACCNT_FLG, "Y"

DEFAULT COLUMN = ACTIVE_FLG, "Y"

DEFAULT COLUMN = BUYING_GROUP_FLG, "N"

DEFAULT COLUMN = CG_DEDN_AUTH_FLG, "Y"

DEFAULT COLUMN = CG_SVP_A_LOCK_FLG, "N"

DEFAULT COLUMN = CG_SVP_LOCK_FLG, "N"

DEFAULT COLUMN = CG_SVP_SKIP_FLG, "N"

DEFAULT COLUMN = CL_SITE_FLG, "N"

DEFAULT COLUMN = DISA_CLEANSE_FLG, "N"

DEFAULT COLUMN = EVT_LOC_FLG, "N"

DEFAULT COLUMN = FCST_ORG_FLG, "N"

DEFAULT COLUMN = FUND_ELIG_FLG, "N"

DEFAULT COLUMN = INCL_FLG, "N"
```

```
DEFAULT COLUMN = INT_ORG_FLG, "N"

DEFAULT COLUMN = PLAN_GROUP_FLG, "N"

DEFAULT COLUMN = PROSPECT_FLG, "N"

DEFAULT COLUMN = PRTNR_FLG, "N"

DEFAULT COLUMN = PRTNR_PUBLISH_FLG, "N"

DEFAULT COLUMN = RPLCD_WTH_CMPT_FLG, "N"

DEFAULT COLUMN = SKIP_PO_CRDCHK_FLG,"N
```

Visibility of Fields: Example of Importing Contacts

This example provides a sample .IFB file for importing contacts. The contact visibility depends on S_CONTACT_BU to resolve the organization and S_POSTN_CON for the position.

```
[Siebel Interface Manager]

      USER NAME = "SADMIN"

      PASSWORD = "SADMIN"

      PROCESS = Import Contact

[Import Contact]

      TYPE = SHELL

      INCLUDE = "Import Contact Informationen"

      INCLUDE = "Import POSTN_CON Informationen"

[Import Contact Informationen]

      TYPE = IMPORT

      TABLE= EIM_CONTACT

      BATCH = 555

      ONLY BASE TABLES = S_PARTY, S_CONTACT, S_CONTACT_BU

    DEFAULT COLUMN = CON_ACTIVE_FLG, "Y"

    DEFAULT COLUMN = CON_DISACLEANSEFLG, "N"

    DEFAULT COLUMN = CON_DISPIMGAUTHFLG, "N"

    DEFAULT COLUMN = CON_EMAILSRUPD_FLG, "N"

    DEFAULT COLUMN = CON_EMP_FLG, "N"

    DEFAULT COLUMN = CON_PRIV_FLG, "N"
```

```
    DEFAULT COLUMN = CON_INVSTGTR_FLG, "N"
    DEFAULT COLUMN = CON_PO_PAY_FLG, "N"
    DEFAULT COLUMN = CON_PROSPECT_FLG, "N"
    DEFAULT COLUMN = CON_PTSHPCONTACTFL, "N"
    DEFAULT COLUMN = CON_PTSHPKEYCONFLG, "N"
    DEFAULT COLUMN = CON_SENDSURVEY_FLG, "N"
    DEFAULT COLUMN = CON_SPEAKER_FLG, "N"
    DEFAULT COLUMN = CON_SUPPRESSEMAILF, "N"
    DEFAULT COLUMN = CON_SUPPRESSFAXFLG, "N"
[Import POSTN_CON Informationen]
      TYPE = IMPORT
      TABLE= EIM_CONTACT1
      BATCH = 555
      ONLY BASE TABLES = S_PARTY, S_CONTACT, S_POSTN_CON
```

Visibility of Fields: Example of Importing Employees

This example is specific to Siebel Industry Applications.

This example provides a sample .IFB file for importing employees. The employee visibility depends on S_CONTACT_BU to resolve the organization, S_POSTN_CON for the position, S_PER_RESP for responsibility, and S_PARTY_PER for the relationship between the S_PARTY and S_CONTACT.

```
[Siebel Interface Manager]
      USER NAME = "SADMIN"
      PASSWORD = "SADMIN"
      PROCESS = Import New Employee
[IMPORT New Employee]
TYPE = SHELL
INCLUDE = "Import Employee"
INCLUDE = "Import Contact"
INCLUDE = "Import Contact1"
[Import Employee]
```

```
        TYPE = IMPORT
        BATCH = 666
        TABLE = EIM_EMPLOYEE
ONLY BASE TABLES = S_PARTY, S_CONTACT, S_EMP_PER, S_PARTY_PER, S_PER_RESP, S_USER
; For S-contact
DEFAULT COLUMN = CON_ACTIVE_FLG, "Y"
        DEFAULT COLUMN = CON_DISACLEANSEFLG, "N"
        DEFAULT COLUMN = CON_EMAILSRUPD_FLG, "N"
        DEFAULT COLUMN = CON_DISPIMGAUTHFLG, "N"
        DEFAULT COLUMN = CON_EMP_FLG, "Y"
        DEFAULT COLUMN = CON_PO_PAY_FLG, "N"
        DEFAULT COLUMN = CON_PRIV_FLG, "N"
        DEFAULT COLUMN = CON_PROSPECT_FLG, "N"
        DEFAULT COLUMN = CON_PTSHPCONTACTFL, "N"
        DEFAULT COLUMN = CON_PTSHPKEYCONFLG, "N"
        DEFAULT COLUMN = CON_SENDSURVEY_FLG, "N"
        DEFAULT COLUMN = CON_SUPPRESSEMAILF, "N"
        DEFAULT COLUMN = CON_SUPPRESSFAXFLG, "N"
; For vertical version
        DEFAULT COLUMN = CON_COURT_PAY_FLG, "N"
        DEFAULT COLUMN = CON_INVSTGTR_FLG, "N"
        DEFAULT COLUMN = CON_SPEAKER_FLG, "N"
        DEFAULT COLUMN = CON_SUSPECT_FLG, "N"
; For S-EMP_PER
DEFAULT COLUMN = ACCEPT_SR_ASGN_FLG, "N"
        DEFAULT COLUMN = CNTRCTR_FLG, "N"
        DEFAULT COLUMN = INT_NEWS_APPR_FLG, "N"
        DEFAULT COLUMN =EMP_CPFINALAPPRFLG, "N"
        DEFAULT COLUMN = STORE_BUDGET_FLG, "N"
        DEFAULT COLUMN =STORE_FORECAST_FLG, "N"
```

```
[Import Contact]
TYPE = IMPORT
        BATCH = 666
        USE INDEX HINTS = TRUE
TABLE = EIM_CONTACT
ONLY BASE TABLES = S_PARTY, S_CONTACT_BU
[Import Contact1]
TYPE = IMPORT
        BATCH = 666
TABLE = EIM_CONTACT1
ONLY BASE TABLES = S_PARTY, S_CONTACT, S_POSTN_CON
```

Visibility of Fields: Example of Importing Opportunities

To make opportunity records visible in the GUI, populate the following tables and columns.

```
S_REVN
      REVN_ITEM_NUM,
      SUMMARY_FLG,
      OPTY_ID,
      ASGN_USR_EXCLD_FLG,
      COMMIT_FLG,
      BU_ID,
      CRDT_POSTN_ID,
      SPLIT_FLG,
      AUTOQUOTE_APPL_FLG,
      REVN_AMT_CURCY_CD,
      DYNMC_GRP_NUM,
      EFFECTIVE_DT,
      PROD_DESC_TEXT
S_OPTY_POSTN
```

```
        ROW_STATUS,
        PRIORITY_FLG,
        COMMITTED_FLG,
        ASGN_SYS_FLG,
        OPTY_ID,
        POSITION_ID,
        CREDIT_ALLC_PCT,
        FCST_CLS_DT,
        FCST_REVN_CURCY_CD,
        ASGN_MANL_FLG,
        ASGN_DNRM_FLG,
    SECURE_FLG,
        OPTY_BU_ID,
        SUM_COMMIT_FLG,
        SUM_EFFECTIVE_DT,
        CONSUMER_OPTY_FLG,
        SUM_REVN_AMT,
        OPTY_NAME,
        OPTY_CLOSED_FLG
    S_OPTY_BU
        OPTY_ID,
        BU_ID,
    SUM_COMMIT_FLG,
        SUM_EFFECTIVE_DT,
        SUM_REVN_AMT,
        OPTY_NAME
    S_OPTY
        PR_POSTN_ID,
        NUM_RC_PERIODS,
        SUM_COMMIT_FLG,
```

```
        CONSUMER_OPTY_FLG,
        PR_REP_DNRM_FLG,
        PR_TERR_ID,
        SECURE_FLG,
        PR_REP_SYS_FLG,
        NAME,
        PR_REP_MANL_FLG,
        STATUS_CD,
        BU_ID,
        CLOSED_FLG,
        SUM_REVN_ITEM_ID,
        SALES_METHOD_ID,
        REVN_SPLIT_FLG,
        APPL_OWNER_TYPE_CD,
        STG_START_DT,
        SUM_EFFECTIVE_DT,
        CURCY_CD,
        EXEC_PRIORITY_FLG,
        ASGN_USR_EXCLD_FLG
```

Visibility of Fields: Example of Importing Assets

To make asset records visible in the GUI, populate the following tables and columns.

```
S_ASSET
        PR_POSTN_ID,
        ALT_FUEL_FLG,
        CAUTION_FLG,
        INTEGRATION_ID,
        ASSET_VAL_EXCH_DT,
        REGISTERED_DT,
```

```
        CUTOFF_FLG,
        ASSET_VAL_CURCY_CD,
        BU_ID,
        ASSET_NUM,
        ROOT_ASSET_ID,
        QTY,
        INSTALL_DT,
        BASE_CURRENCY_CD,
        PROD_ID,
        CUSTOMIZABLE_FLG,
        PR_EMP_ID
    S_ASSET_POSTN
        ASGN_MANL_FLG,
        ASSET_ID,
        POSITION_ID,
        ASGN_SYS_FLG,
        ASGN_DNRM_FLG
    S_ASSET_EMP
        ASSET_ID,
        EMP_ID
    S_ASSET_BU
        ASSET_ID,
        BU_ID
```

Example of Troubleshooting the Import of Extension Columns

Use the guidelines that follow to troubleshoot an import failure that occurs when extension columns are added to some Siebel tables and the EIM import task failed to populate data to these columns.

To troubleshoot the import of extension columns

1. Delete the diccache.dat file from the <siebel server>\bin directory and test the EIM task again.

   EIM will rebuild this file from the information in the repository if the file does not exist.

2. Run the DBCHCK utility to make sure the EIM table in the repository is in synch with the EIM table in the database. For example, use the following command:

   ```
   dbchck /u SADMIN /p <SADMIN's password> /t <table owner> /r "Siebel Repository"
   /l dbchck.log /d /s <ODBC data source> <interface table name>
   ```

   NOTE: For information on running the DBCHCK utility, see "To check the repository using DBCHCK and DICTUTL" on page 185.

   a. If the repository is not in synch with the database, log in to Siebel Tools, and in the Table object, select the EIM table records.

   b. Click the Apply and Activate buttons to apply and activate all changes on the EIM table to the database.

   c. Run the DBCHCK utility again.

3. Follow the instructions to set event logging in "Viewing the EIM Log File" on page 128 and run the EIM task.

   This generates a detailed EIM log file.

   a. Review the log file to see whether there are any errors causing the import failure.

4. If the extension column:

   - is a foreign key or primary column, its mapping should only be created through the EIM Table Mapping Wizard, or by Siebel Expert Services.
     - If the extension column is a foreign key column, its foreign key mapping can be created by running the EIM Table Mapping Wizard on the base table of the extension column.
   - is a primary column for a M:M relationship (that is, an EIM table is defined in the extension primary column's Primary Inter Table property), its EIM Explicit Primary Mapping can be created by running the EIM Table Mapping Wizard on the intersection table.
   - is a primary column for a 1:M relationship (that is, no EIM table is defined in the extension primary column's Primary Inter Table property), its EIM Explicit Primary Mapping can be created by running the EIM Table Mapping Wizard on the primary child table (as defined in the extension primary column's Primary Child Table property).

5. Check the mappings for the extension columns.

   a. Log in to Siebel Tools.

   b. Navigate to the EIM Interface Table object and query to select the interface table EIM Table Mapping.

   If the extension column:

   - is not a foreign key or primary column, it should only have an Attribute Mapping under its base table's EIM Table Mapping.

- is a foreign key column, it should not have any Attribute Mapping defined. It should have a Foreign Key Mapping.
- is a primary column, it should not have any Attribute Mapping or Foreign Key Mapping defined.

If the extension primary column:

- is for a M:M relationship, it should have an EIM Explicit Primary Mapping under its intersection table's EIM Table Mapping.
- is for a 1:M relationship, it should have an EIM Explicit Primary Mapping under its primary child table's EIM Table Mapping.

Checking the Repository

Step 2 in "To troubleshoot the import of extension columns" on page 184 asks you to run the DBCHCK utility to make sure the EIM table in the repository is in synch with the EIM table in the database. Both the DBCHCK and DICTUTL utilities are run from the DOS prompt in the `siebsrvr\bin\` directory. DBCHCK verifies that the physical schema is in synch with the repository. DICTUTL verifies that all dock objects and rule definitions are correct.

To check the repository using DBCHCK and DICTUTL

1. Run the `siebenv.bat` file to make sure that the Siebel application environment variables are set correctly.

   NOTE: There should be no quotes around the parameters in `siebenv.bat`.

2. Make sure there are no quotes around the value to which SIEBEL_REPOSITORY is set.

   If this value is set incorrectly, you will encounter error messages.

3. Run the DBCHCK utility to verify that the physical schema is in synch with the repository. A typical command to run DBCHCK and generate a log file is the following:

   ```
   Prompt>dbchck /S <ODBC_DATASOURCE> /U <USERNAME> /P <PASSWORD> /T <TABLE OWNER>
   /R <REPOSITORY> /L <LOGFILE> /D <CHECK_AGAINST_DICTIONARY> /A <ALL_TABLES>
   ```

   a. Use the `/A` option to specify whether you are running the DBCHCK against all tables; use a Boolean 'Y' (no quotation marks) to specify that you are.

   b. To view all of the options for DBCHCK, run DBCHCK at the DOS prompt without any options.

   This provides all the options that can be used in conjunction with DBCHCK. The following are some of the common options used in conjunction with DBCHCK:

/S	Specifies the ODBC source to use for the database.
/U	Specifies the username to log in to the database.
/P	Specifies the user password to log in to the database.
/T	Specifies the username of the table owner.
/R	Specifies the repository name for the dictionary.
/L	Specifies the log file name for errors.

/D	Checks tables against the dictionary only.
/A	Checks all Siebel tables in the database.

c Check the <LOGFILE> for unacceptable errors.

Unacceptable errors may occur if data types are mismatched. Acceptable errors may occur if a schema object (such as a table or an index) is intentionally external to the repository.

4 Run DICTUTL to verify that all dock objects and rule definitions are correct. A typical command to run DICTUTL and generate a log file is the following:

```
Prompt>dictutl /C <ODBC_DATASOURCE> /U <USERNAME> /P <PASSWORD> /D <TABLEOWNER>
/N <REPOSITORY_NAME> /A <IGNORE_DICTONARY_CACHE> y > LOGFILE.log
```

Further command options are explained as follows:

/A	Y means ignore the dictionary cache.
> LOGFILE.log	LOGFILE is the log file that you designate for DICTUTL.

5 Review the LOGFILE.log file to check for errors.

Example of Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts

This example provides further detail to complement "Troubleshooting the Unique Constraint Error when Importing Accounts or Contacts" on page 77.

The unique constraint error when inserting records using EIM is usually due to inconsistent data in the base tables or incorrect data populated in the interface tables. The inconsistent data may result when two different server tasks, such as Siebel eAI processes and EIM processes, are run at the same time to import the same data.

For example, you populate the EIM_ACCOUNT table with a new record to be added to the S_PARTY table. Uniqueness for the S_PARTY table is based on the S_PARTY_U1 index. However, the associated record in the S_ORG_EXT table that EIM will create may be found to be a duplicate of an existing record in the S_ORG_EXT table, because uniqueness for the S_ORG_EXT table is based on the S_ORG_EXT_U1 index. This duplicate record may have been created by another process, such as an eAI process or a process initiated through the user interface.

Because the S_ORG_EXT table is considered a 1:1 extension table of the S_PARTY table, EIM only checks if there is an existing S_ORG_EXT record that references the S_PARTY row in the PAR_ROW_ID column. In this case, no record is returned since the S_PARTY record is a new one. As a result, the S_ORG_EXT_U1 index is violated when EIM tries to insert the record into the S_ORG_EXT table. This incomplete EIM job then creates new S_PARTY rows without the associated S_ORG_EXT rows.

The PARTY_UID column is part of the user key for the S_PARTY table and is used by the EIM process to identify if an account or contact record is a new record or an existing record.

The S_PARTY.PARTY_UID is a user-definable key and can have any of the following values:

- If contact or account data is being migrated from a system external to Siebel, then the PARTY_UID column can be any user-defined key value or contact or account key in the external system.
- If the contact record is created using the Siebel Web Client, then S_PARTY.PARTY_UID is set to the value in S_PARTY.ROW_ID by default.
- If the account record is created using the Siebel Web Client, then S_PARTY.PARTY_UID is set to the same value in S_PARTY.ROW_ID by default.

The remainder of this topic is divided into two parts which detail diagnostics steps for each of the following two scenarios:

- **Contact data import.** See "Example of Troubleshooting the Import of EIM Contact Data into the S_CONTACT Table" on page 187.
- **Account data import.** See "Example of Troubleshooting the Import of EIM Account Data into the S_ORG_EXT Table" on page 189.

Example of Troubleshooting the Import of EIM Contact Data into the S_CONTACT Table

Use the guidelines that follow to check data consistency in the Siebel tables for contact record import.

To diagnose the unique constraint error for an import of contact data

1. For all contact records, verify that the value in S_PARTY.ROW_ID is set to the same value in S_CONTACT.ROW_ID and S_CONTACT.PAR_ROW_ID. If a record exists in S_PARTY, then a matching record should also exist in S_CONTACT. Run the following two SQL queries to validate the data in these tables:

   a. Query against S_CONTACT:

   ```
   SELECT ROW_ID FROM S_CONTACT WHERE PAR_ROW_ID <> ROW_ID
   ```

   The above statement should return zero rows.

   b. Query against S_PARTY:

   ```
   SELECT PARTY_UID, NAME FROM

   S_PARTY P1

   WHERE PARTY_TYPE_CD = 'Person' AND

   NOT EXISTS

   (SELECT * FROM S_CONTACT O1

   WHERE O1.PAR_ROW_ID = P1.ROW_ID)
   ```

   The above statement should return zero rows.

2. For all contact records, make sure that the corresponding S_PARTY.PARTY_TYPE_CD is set to 'Person'. Use the following SQL statement to validate that this is set correctly:

```
SELECT ROW_ID FROM S_PARTY T1

WHERE EXISTS (SELECT * FROM S_CONTACT T2 WHERE T1.ROW_ID = T2.PAR_ROW_ID)

AND T1.PARTY_TYPE_CD <> 'Person'
```

The above statement should return 0 rows.

3 Populate the exact PARTY_UID value in the EIM_CONTACT table as is in the base table. For example, if a record is created through the Siebel Client UI, then make sure that the value in S_PARTY.PARTY_UID is the same as the value in S_PARTY.ROW_ID. For these records, populate S_PARTY.ROW_ID into EIM_CONTACT.PARTY_UID.

The following SQL query checks for any mismatch in the PARTY_UID values between the EIM_CONTACT and S_PARTY tables:

```
SELECT PARTY_UID FROM EIM_CONTACT T1 WHERE T1.PARTY_TYPE_CD = 'Person' AND

NOT EXISTS (SELECT * FROM S_PARTY T2 WHERE T1.PARTY_UID = T2.PARTY_UID)
```

The above statement should return zero rows.

4 Values in the EIM_CONTACT.PARTY_UID column should be unique in an EIM batch. Use the following SQL statement to verify that there are no duplicate values in this column:

```
SELECT PARTY_UID, COUNT(*) FROM EIM_CONTACT

WHERE IF_ROW_BATCH_NUM = <eim batch#>

GROUP BY PARTY_UID

HAVING COUNT(*) > 1
```

The above statement should return zero rows. If any rows are returned, duplicate values in the PARTY_UID column exist within the same batch. The duplicate rows should be removed from this batch.

Solution

If an EIM batch has records created in an external system, but records created through the Siebel Client UI also exist, make sure that the correct PARTY_UID values are populated in the EIM_CONTACT table. For example, for externally generated contacts, EIM_ CONTACT.PARTY_UID will have a user-defined value from the external system. For contact records that were created using the Siebel Client UI, make sure that the value in EIM_CONTACT.PARTY_UID is set to the value in S_PARTY.ROW_ID.

If externally generated PARTY_UID values are incorrectly populated into contact records created through the Siebel Client UI, where the value in S_PARTY.PARTY_UID equals the value in S_PARTY.ROW_ID, EIM treats these records as new and tries to insert these records into the S_CONTACT table. Because the PARTY_UID value already exists in S_CONTACT, the EIM process fails with the unique constraint violation error for the S_CONTACT table.

If an EIM batch contains both contact records with user-defined PARTY_UID values and records created through the Siebel Client UI, the following solutions can be used to make sure this error does not occur:

- **Option 1.** Configure your Siebel application so that for records generated through the Siebel Client UI, S_PARTY.PARTY_UID matches the format used when loading data into the EIM_CONTACT table.
- **Option 2.** If you have user-defined PARTY_UID values in the S_PARTY table, then before running the EIM process, run the SQL statements that follow to identify any records that exist where an EIM_CONTACT.PARTY_UID value does not match an existing S_PARTY.PARTY_UID value. If records like this exist, then update the EIM_CONTACT table with the correct PARTY_UID value that matches a value in S_PARTY.PARTY_UID.

  The following SQL statement can be used to identify such records in EIM table:

  ```
  SELECT PARTY_UID FROM EIM_CONTACT T1 WHERE T1.PARTY_TYPE_CD = 'Person' AND

  NOT EXISTS (SELECT * FROM S_PARTY T2 WHERE T1.PARTY_UID = T2.PARTY_UID)
  ```

  If the above query does not return any records, run the following query to find the duplicate records:

  ```
  SELECT CON_PERSON_UID FROM EIM_CONTACT T1 WHERE T1.PARTY_TYPE_CD = 'Person' AND

  NOT EXISTS (SELECT * FROM S_CONTACT T2 WHERE T1.CON_PERSON_UID = T2.PERSON_UID)
  ```

  Populate the correct values for PARTY_UID in the EIM_CONTACT table matching the base table S_PARTY.PARTY_UID for such records.

Example of Troubleshooting the Import of EIM Account Data into the S_ORG_EXT Table

The examples below use the EIM_ACCOUNT interface table. You can replace EIM_ACCOUNT with the appropriate EIM table for importing contact data as needed.

Use the guidelines that follow to check data consistency in the Siebel tables for account record import.

To diagnose the unique constraint error for an import of account data

1. For all account records. verify that the value S_PARTY.ROW_ID is set to the same value in S_ORG_EXT.ROW_ID and S_ORG_EXT.PAR_ROW_ID. If a record exists in S_PARTY, then a corresponding record should also exist in S_ORG_EXT. Run the two SQL queries that follow to validate the data in these tables.

   a. Query against S_ORG_EXT:

   ```
   SELECT ROW_ID FROM S_ORG_EXT WHERE PAR_ROW_ID <> ROW_ID
   ```

   The above statement should return zero rows.

   b. Query against S_PARTY:

   ```
   SELECT PARTY_UID, NAME FROM

   S_PARTY P1

   WHERE PARTY_TYPE_CD = 'Organization' AND
   ```

```
NOT EXISTS

(SELECT * FROM S_ORG_EXT O1

WHERE O1.PAR_ROW_ID = P1.ROW_ID)
```

The above statement should return zero rows.

2 For all account records, make sure that the corresponding S_PARTY.PARTY_TYPE_CD is set to 'Organization'. Use the following SQL statement to validate that this is set correctly:

```
SELECT ROW_ID FROM S_PARTY T1

WHERE EXISTS (SELECT * FROM S_ORG_EXT T2 WHERE T1.ROW_ID = T2.PAR_ROW_ID)

AND T1.PARTY_TYPE_CD <> 'Organization'
```

The above statement should return zero rows.

3 Populate the exact PARTY_UID value in the EIM_ACCOUNT table as is in the base table. For example, if a record is created in the Siebel Client UI, make sure that the value in S_PARTY.PARTY_UID is the same as the value in S_PARTY.ROW_ID. For these records, populate S_PARTY.ROW_ID into EIM_ACCOUNT.PARTY_UID.

The following SQL statement checks for any mismatch in the PARTY_UID values between the EIM_ACCOUNT and S_ORG_EXT tables:

```
SELECT PARTY_UID FROM EIM_ACCOUNT T1 WHERE T1.PARTY_TYPE_CD = 'Organization' AND

NOT EXISTS (SELECT * FROM S_PARTY T2 WHERE T1.PARTY_UID = T2.PARTY_UID)
```

The above statement should return zero rows.

4 Values in the EIM_ACCOUNT.PARTY_UID column should be unique in an EIM batch. Use the following SQL statement to verify that there are no duplicate values in this column:

```
SELECT PARTY_UID, COUNT(*) FROM EIM_ACCOUNT

WHERE IF_ROW_BATCH_NUM = <EIM Batch Number>

GROUP BY PARTY_UID

HAVING COUNT(*) > 1
```

The above statement should return zero rows. If any rows are returned, duplicate values in the PARTY_UID column exist within the same batch. The duplicate rows should be removed from this batch.

Solution

If an EIM batch has records created in both the external system and through the Siebel Client UI, make sure that the correct PARTY_UID values are populated in the EIM_ACCOUNT table. For example, for externally generated accounts, EIM_ACCOUNT.PARTY_UID column will have a user-defined value, but for account records created using the Siebel Client UI, the value in EIM_ACCOUNT.PARTY_UID is set to the value in S_PARTY.ROW_ID for the Account.

If externally generated PARTY_UID values are incorrectly populated for account records created using the Siebel Client UI, where the value in S_PARTY.PARTY_UID equals the value in S_PARTY.ROW_ID, EIM treats these records as new records and tries to insert these records into the S_ORG_EXT table. Because the PARTY_UID value already exists in S_ORG_EXT, the EIM process fails with the unique constraint violation error for the S_ORG_EXT table.

If an EIM batch contains both account records with user-defined PARTY_UID and records created through the Siebel Client UI, the two solutions that follow can be used to make sure this error does not occur.

- **Option 1.** Configure your Siebel application so that for records generated through the Siebel Client UI, S_PART.PARTY_UID matches the format you use when loading data into the EIM_ACCOUNT table.
- **Option 2.** If you have user-defined PARTY_UID values in the S_PARTY table, then before running the EIM process, run the following SQL statements to identify any records that exist where an EIM_ACCOUNT.PARTY_UID value does not match with an existing S_PARTY.PARTY_UID value. If records like this exist, then update the EIM_ACCOUNT table with the correct PARTY_UID value that matches a value in S_PARTY.PARTY_UID.

  The following SQL statement can be used to identify such records in the EIM table:

  ```
  SELECT PARTY_UID FROM EIM_ACCOUNT T1 WHERE T1.PARTY_TYPE_CD = 'Organization' AND

  NOT EXISTS (SELECT * FROM S_PARTY T2 WHERE T1.PARTY_UID = T2.PARTY_UID)
  ```

  To correct the data, populate the appropriate values for PARTY_UID in the EIM_ACCOUNT table matching the base table S_PARTY.PARTY_UID for such records.

Example of Importing and Exporting Hierarchical LOVs

You can migrate a hierarchical list of values from one Siebel 7.8 environment to another Siebel 7.8 environment, as shown in this example.

NOTE: The .IFB settings listed in Step 1 below show ROW_ID values for Siebel eBusiness Applications. If you are using Siebel Industry Applications (SIA), first run the SQL suggested in Alert 925 on SupportWeb to get the ROW_ID values, and then replace the corresponding ROW_ID values in the the .IFB settings for Step 1.

To migrate a hierarchical list of values

1 Run an EIM export task using the following .IFB settings:

```
[Siebel Interface Manager]

   USER NAME = "SADMIN"

   PASSWORD = "SADMIN"

   PROCESS = Export LOV
```

```
[Export LOV]
   TYPE = SHELL
   INCLUDE = "Export LOV_TYPE"
   INCLUDE = "Export LOV_REPLICATION_LEVEL"
   INCLUDE = "Export LOVs_Parent"
   INCLUDE = "Export LOVs_Child"
   USE INDEX HINTS = TRUE

[Export LOV_TYPE]
   TYPE = EXPORT
   BATCH = 1
   TABLE = EIM_LST_OF_VAL
   EXPORT MATCHES = (TYPE = 'LOV_TYPE' and \
      VAL <> 'LOV_TYPE' and \
      VAL <> 'REPLICATION_LEVEL')
   USE INDEX HINTS = TRUE

[Export LOV_REPLICATION_LEVEL]
   TYPE = EXPORT
   BATCH = 2
   TABLE = EIM_LST_OF_VAL
   EXPORT MATCHES = (TYPE = 'REPLICATION_LEVEL' and \
      VAL <> 'All')
   USE INDEX HINTS = TRUE

[Export LOVs_Parent]
   TYPE = EXPORT
   BATCH = 3
   TABLE = EIM_LST_OF_VAL
   EXPORT MATCHES = (TYPE <> 'LOV_TYPE' and \
```

```
      PAR_ROW_ID IS NULL and \
      ROW_ID NOT IN ('0-1EOTJ', '0-1EOTR', \
      '0-1EOTT', '0-1EOTX', '0-1EOTZ', \
      '0-1EOUB', '0-1EOUF', '0-1EOUH', \
      '0-1EOUJ', '0-1EOUL', '0-1EOUN', \
      '0-1EOUR', '0-2SRAZ', '0-3EM3U', \
      '0-3EM3Y', '0-3EM42', '0-3G4D0', \
      '0-3G4D2', '0-3GBNN', '0-3GFJQ', \
      '0-3GFJV', '0-3K8OB', '0-3LEF9', \
      '0-3LG6Z', '0-3RL6J', '0-3YWL5', \
      '0-3YWLD', '0-40X27', '0-6ECJG', \
      '04-AZLJB', '04-AZLJD', '04-AZLJF', \
      '04-AZLJH', '04-BF0LX', '04-BF0LZ', \
      '04-BF0M1', '04-BF0M3', '04-BF0M7', \
      '04-BF0M9', '04-BF0MO', '04-BKLND', \
      '04-BKLNN', '04-CYI2Z', '04-CYI32', \
      '04-CYI34'))
   USE INDEX HINTS = TRUE

[Export LOVs_Child]
   TYPE = EXPORT
   BATCH = 4
   TABLE = EIM_LST_OF_VAL
   EXPORT MATCHES = (TYPE <> 'LOV_TYPE' and \
      PAR_ROW_ID IS NOT NULL and \
      ROW_ID NOT IN ('0-6DCE7', '04-AQ79M', \
      '04-AQ79O', '04-AQ79Q'))
   USE INDEX HINTS = TRUE
```

NOTE: The ROW_ID values for LOVs with NAME greater than 30 characters must be included in the "ROW_ID NOT IN" clause of the [Export LOVs_Parent] and [Export LOVs_Child] sections. For more information, see Alert 925 on SupportWeb.

2 Run the SQL statement that follows to populate the EIM_LST_OF_VAL.PAR_BI and other EIM_LST_OF_VAL.*_BU interface columns.

NOTE: This SQL statement can be found in *<siebel server root>*\Admin\eim_export_lookup_bu_name.sql. Locate the SQL for EIM_LST_OF_VAL.

```
update EIM_LST_OF_VAL IT

   set IT.BITMAP_LIT_BU = (select min(OI.NAME) from S_BU OI where OI.ROW_ID =
IT.BITMAP_LIT_BI)

   ,   IT.LOV_BU = (select min(OI.NAME) from S_BU OI where OI.ROW_ID = IT.LOV_BI)

   ,   IT.LOV_VIS_BU = (select min(OI.NAME) from S_BU OI where OI.ROW_ID =
IT.LOV_VIS_BI)

   ,   IT.PAR_BU = (select min(OI.NAME) from S_BU OI where OI.ROW_ID = IT.PAR_BI);
```

3 Make sure the target environment's EIM_LST_OF_VAL interface table is empty, then move the exported data from the source environment's EIM_LST_OF_VAL interface table to the target environment's EIM_LST_OF_VAL interface table.

4 At the target environment, verify the existence of the three list of values records that follow before proceeding to Step 5.

Type	Display Value	Replication Level
LOV_TYPE	LOV_TYPE	All
LOV_TYPE	REPLICATION_LEVEL	All
REPLICATION_LEVEL	All	All

a If the above three records do not exist, create them in the Siebel client at Administration - Application > LOV Explorer view.

5 Run the following SQL at the target environment's database:

```
UPDATE EIM_LST_OF_VAL A

   SET A.IF_ROW_BATCH_NUM = 5

   WHERE NOT EXISTS (SELECT 'x'

   FROM EIM_LST_OF_VAL B

   WHERE B.LOV_TYPE = A.PAR_TYPE

   AND B.LOV_VAL = A.PAR_VAL

   AND B.LOV_LANG_ID = A.PAR_LANG_ID
```

```
AND (B.LOV_SUB_TYPE = A.PAR_SUB_TYPE

OR (B.LOV_SUB_TYPE IS NULL

AND A.PAR_SUB_TYPE IS NULL))

AND B.LOV_BU = A.PAR_BU

AND B.IF_ROW_BATCH_NUM <= 3)

AND A.IF_ROW_BATCH_NUM = 4;
```

6 If the SQL listed in Step 5 has updated zero records, proceed to Step 7 on page 196. Otherwise, run the following SQL at the target environment's database and repeat Step 6 until the SQL has updated zero records:

```
UPDATE EIM_LST_OF_VAL A

SET A.IF_ROW_BATCH_NUM = <see Note A row in table below>

WHERE NOT EXISTS (SELECT 'x'

FROM EIM_LST_OF_VAL B

WHERE B.LOV_TYPE = A.PAR_TYPE

AND B.LOV_VAL = A.PAR_VAL

AND B.LOV_LANG_ID = A.PAR_LANG_ID

AND (B.LOV_SUB_TYPE = A.PAR_SUB_TYPE

OR (B.LOV_SUB_TYPE IS NULL

AND A.PAR_SUB_TYPE IS NULL))

AND B.LOV_BU = A.PAR_BU

AND B.IF_ROW_BATCH_NUM = <see Note B row in table below>)

AND A.IF_ROW_BATCH_NUM = <see Note C row in table below>;
```

Note	Value
A	Next new batch number; that is, 6 for the first time you run, 7 for the second time you run, and so on.
B	Last batch number; that is, 4 for the first time you run, 5 for the second time you run, and so on.
C	Last batch number; that is, 5 for the first time you run, 6 for the second time you run, and so on.

7 Run the following SQL at the target environment's database:

```
UPDATE EIM_LST_OF_VAL
```

```
SET IF_ROW_BATCH_NUM = <next new batch number>

WHERE LOV_VIS_BU IS NOT NULL;
```

8 Run an EIM import task at the target environment using the following parameters:

```
[Siebel Interface Manager]

    USER NAME = "SADMIN"

    PASSWORD = "SADMIN"

    PROCESS = Import LOV


[Import LOV]

    TYPE = IMPORT

    BATCH = 1-<last batch number as specified in step 7>

    TABLE = EIM_LST_OF_VAL

    USE INDEX HINTS = TRUE
```

9 Migrate the S_LOV_REL rows using the EIM_LOV_REL interface table.

EIM Merge Process Example

This section provides an example you might find useful when merging custom columns.

Example of Running a Merge with Custom Columns

In this example, you run a merge that includes two account records with the same location (LOC), and a string of information in the old record that must be copied into the new record. The two records have different values for Name because the account had a name change. The information contained in the records that result from the merge is as follows:

Record	LOC	Name	X_CUSTOM_COLUMN
Old record	1	A	top-tier account
Survivor	1	B	

When these two accounts are merged, the information in the old record's custom column is lost and the custom column in the survivor record appears blank.

NOTE: EIM behavior, whether executed from the GUI or through an EIM run, does not merge data in the base record. It simply repoints the foreign keys in the dependent child records. This applies to all columns in the base table. This could lead to unintended data loss in an extension column.

EIM Delete Process Examples

This section provides usage examples that can be applied to your running of delete processes.

Example: Using DELETE MATCHES to Delete Data from S_PARTY Extension Tables

If the EIM table's target table is S_PARTY:

The syntax is as follows:

```
DELETE MATCHES = S_PARTY, [...criteria...]

DELETE MATCHES = [non-target base tables name of Siebel Extension type],
[...criteria...]
```

In this example, you want to delete an existing account. This account's data is as follows:

```
S_PARTY:  PARTY_TYPE_CD='Organization', PARTY_UID='1-28XIF'

S_ORG_EXT: LOC='San Mateo', NAME='TEST', BU_ID=' 0-R9NH"
```

If you would like to apply criteria against the S_PARTY table, you can use the following session in the .IFB file:

```
[Delete Account]

TYPE = DELETE

BATCH = 100

TABLE = EIM_ACCOUNT

DELETE MATCHES = S_PARTY, (PARTY_UID = '1-28XIF')
```

Or if you would like to apply criteria against the S_ORG_EXT table, you can use the following session in the .IFB file:

```
[Delete Account]

TYPE = DELETE

BATCH = 100

TABLE = EIM_ACCOUNT
```

```
DELETE MATCHES = S_ORG_EXT, (NAME = 'TEST')
```

Both methods achieve the same result. But in this example, it is easier to use criteria against S_ORG_EXT, since you know which account you want to delete.

NOTE: When S_PARTY is the target base table, you cannot use the EIM table name or neglect the target base table name in DELETE MATCHES expressions.

Example: Using DELETE MATCHES to Delete Data from non-S_PARTY Extension Tables

If the EIM table's target table is not S_PARTY:

```
DELETE MATCHES = [EIM table name], [...criteria...]

DELETE MATCHES = [target base table name], [...criteria...]

DELETE MATCHES = [...criteria...
```

For example, if you want to delete all activities created by employee SADMIN, you go to the S_EVT_ACT table and find all the records with the following:

```
OWNER_LOGIN='SADMIN'
```

You can use the following session in your .IFB file:

```
[Delete Activity]

TYPE = DELETE

BATCH = 100

TABLE = EIM_ACTIVITY

DELETE MATCHES = <Table>, (OWNER_LOGIN = 'SADMIN')
```

<Table> can be replaced by EIM_ACTIVITY or S_EVT_ACT, or it can be left empty.

Example of Using DELETE EXACT

The DELETE EXACT parameter is used to delete rows in a Siebel base table with user key values specified in the corresponding EIM table. In this case, the corresponding EIM table has to be populated.

In this example, you want to delete an existing account. This account's user key data is as follows:

```
S_PARTY:  PARTY_TYPE_CD='Organization', PARTY_UID='1-28XIF'

S_ORG_EXT: LOC='San Mateo', NAME='TEST', BU_ID=' 0-R9NH"
```

To delete an existing account

1. Choose the EIM_ACCOUNT table and populate this table as follows:

   EIM_ACCOUNT.LOC ='San Mateo'

   EIM_ACCOUNT.NAME ='TEST'

   EIM_ACCOUNT.ACCNT_BU ='Default Organization' (corresponding to BU_ID=' 0-R9NH")

2. Populate the other required columns of the EIM_ACCOUNT table, such as IF_ROW_BATCH_NUM.
3. Run the EIM delete process.

The following is an excerpt from a sample .IFB file:

```
[Delete Account]

   TYPE = DELETE

   BATCH = 300

   TABLE = EIM_ACCOUNT

   ONLY BASE TABLES = S_ORG_EXT

   DELETE EXACT=TRUE
```

To delete an existing account using S_PARTY's user key to populate the EIM_ACCOUNT table

1. Choose the EIM_ACCOUNT table and populate this table as follows:

   EIM_ACCOUNT : PARTY_TYPE_CD='Organization' and PARTY_UID='1-28XIF'

2. Populate the other required columns of the EIM_ACCOUNT table, such as IF_ROW_BATCH_NUM.
3. Run the EIM delete process.

The following is an excerpt from a sample .IFB file:

```
[Delete Account]

   TYPE = DELETE

   BATCH = 300

   TABLE = EIM_ACCOUNT

   ONLY BASE TABLES = S_PARTY

   DELETE EXACT=TRUE
```

Both examples above achieve the same result.

Note the following when you use DELETE EXACT:

- In the .IFB file, you must specify ONLY BASE TABLES, so that only this data will be deleted.

- Only one base table can be specified in the ONLY BASE TABLES parameter. Otherwise, unexpected SQL statements will be generated
- If you want to delete data in two or more tables, you must specify two or more sessions in your .IFB file, since you can only specify one table in each session.

The following are the differences between DELETE EXACT and DELETE MATCHES:

- DELETE MATCHES does not require data population of an EIM table, while DELETE EXACT does. So DELETE MATCHES is easier to use when the deleting criterion is simple.
- DELETE MATCHES does not work well with complicated deleting criterion, because you do not get the chance to check whether you are mistakenly deleting the right data. With DELETE EXACT, you can always check the data in the EIM table before you start the EIM delete process.
- DELETE MATCHES can only be used when the deleting criterion is against a target base table (or against its extension table if the target base table is S_PARTY), and when only one base table is involved. However, with DELETE EXACT, you can always use EIM or SQL statements to export the user key data from the base table to the EIM table, and then cleanse the data. As long as the corresponding user key columns in the EIM table can be populated, DELETE EXACT can be used to delete the data in the base table.

To find the target base table of an EIM table

1. In Siebel Tools, navigate to EIM Interface Table control, and query the EIM table name.
2. Check the Target Table property to find the target base table name.

Example of Deleting Specific Positions from Accounts

To delete specific positions from an account, you must populate the interface table EIM_ACCOUNT with an SQL script in addition to making modifications to the .IFB file. This is because DELETE MATCHES does not work for nonbase tables.

You can use the following sample .IFB file:

```
[Siebel Interface Manager]

    USER NAME = "SADMIN"

    PASSWORD = "SADMIN"

    PROCESS = DELETE

[DELETE]

    TYPE = SHELL

    INCLUDE = "Delete Accounts Main"

[Delete Accounts Main]

      TYPE = DELETE
```

```
BATCH = 1

TABLE = EIM_ACCOUNT

ONLY BASE TABLES = S_ACCNT_POSTN

DELETE EXACT = TRUE
```

Examples of Resolving Foreign Keys

The examples in this section illustrate ways of dealing with foreign keys that do not resolve to existing values.

Example 1: Error Message "This is a foreign key value in the base table and the values in the interface table did not resolve to existing values."

EIM reports the low-severity error that follows when the foreign key value in the base table does not match the value in the EIM table.

NOTE: This error example is based on the Siebel version 7.7 data model.

```
EIM_SRV_REQ

------------

CAT_CTLG_BI

CAT_CTLG_TYPE_CD

CAT_CTLG_NAME

CAT_CTLG_VER_NUM

CAT_CTLG_CAT_ENDDT

CAT_CTLG_CAT_NAME


Base table:

S_CTLG_CAT_SR

-----------

CTLG_CAT_ID
```

This is a foreign key value in the base table and the values in the interface table did not resolve to existing values. Verify that the IF columns correspond to existing base table rows. This failure caused the rows to be eliminated from further processing for this secondary base table. However, processing of the rows WILL continue for other destination base tables.

Resolution

To resolve the foreign key value, you must find the user key columns in the foreign key table. Based on multiple columns, user keys are used to uniquely identify rows within tables for EIM processing.

Table 1 lists the user key attributes and base table columns for the EIM_SRV_REQ interface column discussed in this example.

Table 1. User Key Attributes and Base Table Columns for the EIM_SRV_REQ Interface Column

EIM_SRV_REQ Column	User Key Attribute	Base Table Column
CAT_CTLG_BU	CTLG_ID/BU_ID/NAME	S_BU.NAME
CAT_CTLG_TYPE_CD	CTLG_ID/CTLG_TYPE_CD	S_CTLG.CTLG_TYPE_CD
CAT_CTLG_NAME	CTLG_ID/NAME	S_CTLG.NAME
CAT_CTLG_VER_NUM	CTLG_ID/VERSION_NUM	S_CTLG.VERSION_NUM
CAT_CTLG_CAT_ENDDT	EFF_END_DT	S_CTLG_CAT.EFF_END_DT
CAT_CTLG_CAT_NAME	NAME	S_CTLG_CAT.NAME

The following example task shows how the user key plays a role to identify the base column for corresponding EIM columns for the above scenario.

To resolve the foreign key value

1. Identify the foreign key table to which S_CTLG_CAT_SR.CTLG_CAT_ID points.
   a. In Siebel Tools, in the Object Explorer list, go to the Table object and query for the S_CTLG_CAT_SR table.
   b. Navigate to the Column object and query for the CTLG_CAT_ID column.
   c. Verify that the foreign key table value is S_CTLG_CAT.
2. Find the user key columns defined in the S_CTLG_CAT table.
   a. In Siebel Tools, in the Object Explorer list, go to the Table object and query for the S_CTLG_CAT table.
   b. Navigate to the User Key object and select the U1 index (S_CTLG_CAT_U1).
   c. Navigate to the User Key Column object and verify that the User Key columns for S_CTLG_CAT are CTLG_ID (FK), EFF_END_DT, and NAME.
3. In Siebel Tools, identify the foreign key table to which S_CTLG_CAT.CTLG_ID points: S_CTLG

4 Find the user key columns defined in the S_CTLG table: BU_ID (FK), CTLG_TYPE_CD, NAME, and VERSION_NUM

5 Identify the foreign key table to which S_CTLG.BU_ID points: S_BU

6 Find the user key columns defined in the S_BU table using Siebel Tools: NAME

7 Based on the above results, populate the following interface columns as listed in the table below to resolve the S_CTLG_CAT_SR.CTLG_CAT.ID foreign key.

Interface Column Name	Instructions
CAT_CTLG_BU	Populate with S_BU.NAME value from Step 6 on page 203.
CAT_CTLG_TYPE_CD	Populate with S_CTLG.CTLG_TYPE_CD value from Step 4 on page 203.
CAT_CTLG_NAME	Populate with S_CTLG.NAME value from Step 4 on page 203.
CAT_CTLG_VER_NUM	Populate with S_CTLG.VERSION_NUM value from Step 4 on page 203.
CAT_CTLG_CAT_ENDDT	Populate with S_CTLG_CAT.EFF_END_DT value from Step c on page 202.
CAT_CTLG_CAT_NAME	Populate with S_CTLG_CAT.NAME value from Step 2 on page 202.

Example 2: Resolving the Foreign Key for Position Division

The table S_ORG_EXT is used to store the Account records and the internal Division records. The user key of S_ORG_EXT consists of the columns NAME, LOC, and BU_ID. For Division records, BU_ID always references Default Organization.

During an EIM run, in order to identify the foreign key S_POSTN.OU_ID, EIM needs information about the user key columns of S_ORG_EXT. The foreign key S_POSTN.OU_ID points to Division records in S_ORG_EXT. So the division's NAME, LOC, and Default Organization should be used to resolve the OU_ID.

NOTE: S_POSTN also has a foreign key BU_ID which may or may not reference Default Organization. This BU_ID is not to be confused with the BU_ID in the user key of S_ORG_EXT. Do not use it together with the division's NAME and LOC to resolve S_POSTN.OU_ID; doing this can result in failure if the BU_ID references organizations other than Default Organization.

Example 3: Resolving the Foreign Key Using a Special User Key

This example is specific to Siebel Industry Applications (SIA).

A typical example of using a special user key (rather than the normal U1 user key) to resolve foreign keys is the use of the special user key "S_ADDR_PER:Communications" in the resolution of foreign keys to S_ADDR_PER in SIA. This special user key contains only the column ADDR_NAME, in contrast to (ADDR_NAME, PER_ID) in the U1 user key.

The mapping of S_ORDER.BL_ADDR_ID in EIM_ORDER, for example, uses the special user key "S_ADDR_PER:Communications" instead of the U1 user key. In fact, mappings of all foreign keys to S_ADDR_PER in SIA use this special user key instead of the U1 user key.

Other Examples

The examples below illustrate various ways of working with EIM: setting explicit primary mappings, improving EIM performance, defining foreign key column values, implementing a multi-org hierarchy, adding a position to a party table, and using the EIM_ASSET interface table.

Example of Setting Explicit Primary Mappings

After importing a Contact using EIM_CONTACT, you can use EIM_CONTACT3 to import the Contact's personal email addresses into S_PER_COMM_ADDR. You can explicitly set the primary email address by populating the primary flag column CON_PR_EMAIL_ADDR with Y.

Table 2 shows an example of setting the primary email address for Contact "CON100" to "John.Smith@hotmail.com."

Table 2. Explicit Primary Mapping for a Contact

PARTY_UID	PARTY_TYPE_CD	CON_PERSON_UID	CON_PRIV_FLG	CON_BU	ADDR_COMMMEDIUM_CD	ADDR_ADDR	CON_PR_EMAIL_ADDR
CON100	Person	CON100	N	Default Organization	Email	JSmith@yahoo.com	
CON100	Person	CON100	N	Default Organization	Email	JSmith@hotmail.com	Y
CON100	Person	CON100	N	Default Organization	Email	JSmith@gmail.com	

If an explicit primary mapping is not used or not used properly—such as no address or more than one address flagged as the primary email address—then EIM ignores this explicit primary mapping and sets the primary implicitly.

Example of Setting Explicit Primary Mappings for Many-to-Many Relationships

"Example of Setting Explicit Primary Mappings" on page 204 explains how to set an explicit primary for a one-to-many relationship. When setting a primary key for a many-to-many relationship, such as the relationship between Opportunities and Contacts, there is an intersection table to consider.

As an example, you can work with the primary S_OPTY.PR_CON_ID. First you import into S_CONTACT using EIM_CONTACT. Then you use EIM_OPTY to import into S_OPTY and the intersection table S_OPTY_CON, and explicitly set the primary S_OPTY.PR_CON_ID during this process.

The column definitions for one-to-many primaries are different from those of many-to-many primaries. In the case of a one-to-many primary, such as S_CONTACT.PR_EMAIL_ADDR_ID, the foreign key table and the primary child table are both defined as S_PER_COMM_ADDR, and the primary intersection table is empty. In the case of a many-to-many primary, such as S_OPTY.PR_CON_ID, the foreign key table is S_CONTACT, and both the primary child table and the primary intersection table are defined as S_OPTY_CON. The explicit primary mapping for S_OPTY.PR_CON_ID is under the table mapping of its primary child table, that is, S_OPTY_CON. It could be easy to mistake S_CONTACT as the primary child table for S_OPTY.PR_CON_ID and this could lead you to look for an explicit primary mapping. This explicit primary mapping would not be found, however, because S_CONTACT is not mapped in EIM_OPTY.

Example of Creating Mappings for Extension Columns

For an example of how to map extension columns, see the section on the EIM Table Mapping Wizard in *Configuring Siebel eBusiness Applications*.

Example of Improving Performance by Dropping Indexes

Often, especially for initial EIM loads, you can improve EIM performance by determining that there are indexes present which are not being used for a particular EIM process. By pinpointing the unnecessary indexes, and by dropping them for the duration of an EIM run, you can achieve performance improvements. For an example of this, see "Dropping Indexes in Initial Runs" on page 157.

Foreign Key Column Values: NO MATCH ROW ID versus NULL versus a Valid ROW_ID

There are three possible values that EIM can define for primary columns (foreign key columns) when it processes a batch:

- NO MATCH ROW ID
- NULL
- A valid ROW_ID

NO MATCH ROW ID. EIM sets the foreign key columns to NO MATCH ROW ID if the primary value cannot be found when EIM processes Step 10 on page 55. EIM does this because the primary key is missing in the linked table.

NOTE: The following are special considerations regarding NO MATCH ROW ID:

S_CONTACT. The export function will update the BU_ID on the S_CONTACT table to NO MATCH ROW ID when there is no record existing in the S_CONTACT_BU table for a given contact. To avoid this, every contact should have a corresponding record in the S_CONTACT_BU table.

S_PRI_LST_BU. The S_PRI_LST_BU table must be loaded to avoid having the UI set S_PRI_LST.BU_ID to NO MATCH ROW ID.

NULL. If the foreign key columns allow a NULL value in the parent table, EIM carries the NULL value.

A valid ROW_ID. If a valid ROW_ID is not defined, EIM uses the value in the primary column to determine the ROW_ID.

Example of Using the NUM_IFTABLE_LOAD_CUTOFF Parameter

When the NUM_IFTABLE_LOAD_CUTOFF parameter is enabled, EIM loads all schema mappings if the value is less than the number of EIM tables used in the run process. To enable this parameter, set the value to a positive number that is less than the number of EIM tables used in the run process. For example, if the EIM process is using one EIM table, then the setting should be NUM_IFTABLE_LOAD_CUTOFF = 0.

When this parameter is disabled, EIM loads only mappings for the EIM tables used in the run process. This speeds up the dictionary loading process in EIM. To disable this parameter, set the value to -1.

NOTE: NUM_IFTABLE_LOAD_CUTOFF is disabled by default.

EIM does not necessarily look at all of the EIM tables in the IFB file. EIM counts only the number of EIM tables being used in the running process.

For example, in the .IFB file that follows, there are three EIM tables: EIM_ACCOUNT, EIM_CONTACT, and EIM_OPTY. But there are only two EIM tables (EIM_ACCOUNT, EIM_CONTACT) for the process to be run (Import Objects). So with a NUM_IFTABLE_LOAD_CUTOFF value of 2, EIM does not load all of the schema mappings. If you want EIM to load all of the schema mappings in this example, set the NUM_IFTABLE_LOAD_CUTOFF value to 1 (or 0).

By setting the parameter to 2 in this example, you are disabling it because the number is equal to, not less than, the number of EIM tables used in the run process.

Sample .IFB file:

```
[Siebel Interface Manager]

   PROCESS = Import Objects

[Import Objects]
```

```
   TYPE = SHELL
   INCLUDE = Import Accounts
   INCLUDE = Import Contacts
[Import Accounts]
   TYPE = IMPORT
   BATCH = 100
   TABLE = EIM_ACCOUNT
[Import Contacts]
   TYPE = IMPORT
   BATCH = 100
   TABLE = EIM_CONTACT
[Export Opty]
   TYPE = Export
   BATCH = 100
   TABLE = EIM_OPTY
```

Example: Transaction Logging with Row-by-row Processing versus Set-based Processing

Transaction logging is enabled and disabled from within the System Preferences view in the Administration - Application screen. The preference is called Docking: Transaction Logging. You can also adjust the transaction logging system preference setting by changing the LOG TRANSACTIONS parameter in the EIM configuration file. For more information, see "Process Section Parameters Generic to All EIM Processes" on page 39.

LOG TRANSACTIONS and SET BASED LOGGING Parameters

EIM performs row-by-row (RBR) transaction logging when LOG TRANSACTIONS is set to TRUE and SET BASED LOGGING is set to FALSE in the .IFB file. In row-by-row logging mode, EIM fetches all the data to the client.

Most of the time, SET BASED LOGGING is not explicitly set. When SET BASED LOGGING is not explicitly set, the Docking: Transaction Logging system preference is used to determine the processing method.

When Docking: Transaction Logging is disabled, all operations (insert, update, and delete) are performed in set-based mode. If you explicitly set LOG TRANSACTIONS = FALSE in the .IFB file, then EIM does not log any transactions into the Master Transaction Log table.

When Docking: Transaction Logging is enabled, all inserts and deletes are performed in set-based mode, while updates are performed in RBR mode. When Docking: Transaction Logging is set to TRUE, EIM logs transactions into either the .DX files or the S_DOCK_TXN_LOG table, depending on the setting for LOG TRANSACTIONS TO FILE in the .IFB file.

When SET BASED LOGGING is explicitly set, EIM uses the value of this parameter to determine the processing mode. When SET BASED LOGGING is TRUE, all operations (insert, update, delete) are performed in set-based mode. When SET BASED LOGGING is FALSE, all operations are performed in RBR mode. For import and delete processes, it is not recommended that the SET BASED LOGGING parameter be set to TRUE because in most cases, there is no reason to set this parameter explicitly. For merge processes, SET BASED LOGGING must be set to FALSE for transaction logging to work properly.

To log every transaction separately, EIM changes its operation mode to RBR.

Logging Locations for LOG TRANSACTIONS and LOG TRANSACTIONS TO FILE

With RBR processing, data is logged according to one of three scenarios:

1. EIM logs transactions into S_DOCK_TXN_LOG table or .DX files if:
   - System Preference "Docking: Transaction Logging" is set to TRUE
     - LOG TRANSACTIONS = TRUE in the .IFB file (default)
     - LOG TRANSACTIONS TO FILE = TRUE in the .IFB file (default)

   If LOG TRANSACTIONS = TRUE and LOG TRANSACTIONS TO FILE = TRUE (which are the default values), EIM logs a transaction into the .DX files, which are stored in <*SiebelFileSystem**eim*> folder. EIM creates the "marker" transaction in the S_DOCK_TXN_LOG table.

   S_DOCK_TXN_LOG.LOG_DATA1 (or LOG_DATA_2, 3, 4) stores the name and the location of the .DX file as in the following example:

   N128*d:\15051sia\FS\eim\1-CP-1.dx

   In this example, 1-CP-1.dx is the name of the .DX file and it is located in the \\15051sia\FS\eim folder.

   Once the data is created in the .DX file and the marker transactions have been created in the S_DOCK_TXN_LOG table, Transaction Processor (TxnProc) captures the .DX files from the \eim folder and brings them into the TxnProc folder for Transaction Router (TxnRouter) to process.

2. EIM logs transactions into the S_DOCK_TXN_LOG table, and eventually, TxnProc or TxnMerge creates the .DX file in the <*SiebSrvr**Docking**TxnProc*> folder if:
   - LOG TRANSACTIONS = TRUE in the .IFB file (default)
   - LOG TRANSACTIONS TO FILE = FALSE in the .IFB file

3. EIM does not log any transactions into the S_DOCK_TXN_LOG table or in the .DX file if:

- LOG TRANSACTIONS = FALSE in the .IFB file

  This is the case regardless of what value is defined for LOG TRANSACTIONS TO FILE.

  The LOG TRANSACTIONS parameter is explicitly used to control whether changes made by EIM to the Siebel database should be visible to remote clients (by logging transactions for use by the Docking Manager, which synchronizes the Siebel database with remote clients).

When to Use Row-by-Row Processing

For import and delete processes, you should let EIM determine which mode to use. EIM will use the method with the best performance for the functionality requested. For initial data loading, you can disable transaction logging for improved performance (EIM will use set-based mode for all operations). For ongoing operations with Mobile Web Clients, transaction logging should be enabled (EIM will choose set-based logging for inserts and deletes, and RBR for updates).

For merge processes with transaction logging enabled, you must explicitly set EIM to run in RBR mode in order for transaction logging to work properly.

The following are examples of cases when RBR logging should be used:

- Running an EIM import task using the COMMIT OPERATIONS parameter.

  NOTE: COMMIT OPERATIONS is useful only for RBR logging.

- Running an EIM merge task. To enable transaction logging for an EIM merge process, the EIM merge process runs in ongoing (row-by-row) mode.

Advantages and Disadvantages of Using RBR Logging

Advantages of RBR logging include the following:

- Since EIM inserts all the information (data as well as column info) into the S_DOCK_TXN_LOG table, in cases for which mobile clients bring about synchronization problems, RBR logging is beneficial in making troubleshooting easier.
- The row-by-row mode is required for logging update transactions. If the SET BASED LOGGING parameter is not set, then EIM runs in RBR for update operations when transaction logging is enabled. RBR is also required for merge processes when transaction logging is enabled.

  NOTE: When running merge processes with transaction logging, SET BASED LOGGING = FALSE is required. If EIM performs a merge with set-based logging instead of RBR, transactions are not written to S_DOCK_TXN_LOG even though TRANSACTION LOGGING is set to TRUE. The merge works correctly, but remote clients cannot get the transactions and are out of synch as a result.

A disadvantage of RBR logging is that RBR logging affects performance, especially with large imports and deletes.

Example of Implementing a Multi-Organization Hierarchy

If you use multi-org, this means that a single record is shared across multiple organizations. For overview information on multi-org, see the section on access control in *Security Guide for Siebel eBusiness Applications*.

In this example, you are adding a new organization to convert a single-org to a multi-org. The process of converting a single-org to a multi-org involves adding the additional organization and its related structure, adding positions, and then associating the data to the new organization.

NOTE: Some data, such as Accounts, has a many-to-many relationship to organizations, while other data, such as Contacts and Service Requests do not.

To convert from single-org to multi-org

1. Add a new organization (New Org) into the Organization table.
2. Assign records to the new organization.

   You can assign records through the GUI or using EIM.

   For example, assign an Employee record (Emp1). The Employee records are stored in the S_CONTACT table. There is a many-to-many relationship between the employee and the organization, so the intersection table S_CONTACT_BU holds the relationship between the organization and the employee.

   You add a new record in the S_CONTACT_BU intersection table to hold the relationship between New Org and Emp1. Now Emp1 is available to both the original organization and New Org.
3. Verify that you can see the record in both organizations.

Example of Adding a Position to a Party Table

This example shows how positions are added to party tables, such as Account, Contact, and Employee. You are adding positions to the Account table.

You can use the EIM_ACCOUNT table to populate S_ACCNT_POSTN, which is an intersection table between Accounts and Position.

In the S_ACCNT_POSTN table, you provide information about the position you are trying to add (POSITION_ID) and the account you are trying to associate with the position (OU_EXT_ID).

In the EIM_ACCOUNT table, you provide information about the account.

To populate the EIM table, you must always include the target base table: in this case, S_PARTY. Since EIM_ACCOUNT is for account information, S_PARTY should also be filled with account information. So you set the S_PARTY.PARTY_TYPE_CD = 'Organization' since Account belongs to the Organization type. (PARTY_TYPE_CD = ‘Person’ is only used for Contact, User, Employee, or Partner.)

The .IFB file looks like this:

```
[Add Position]
      TYPE = IMPORT
      BATCH = 2002
      TABLE = EIM_ACCOUNT
      ONLY BASE TABLES = S_PARTY, S_ACCNT_POSTN
      INSERT ROWS = S_PARTY, FALSE
      INSERT ROWS = S_ORG_EXT, FALSE
      INSERT ROWS = S_ACCNT_POSTN, TRUE
      UPDATE ROWS = S_ACCNT_POSTN, TRUE
```

Example of Using the EIM_ASSET Interface Table

Table 3 shows an example of how to populate the EIM_ASSET interface table for an import process in order to properly display product and part number information in the Siebel application's Asset Management - Assets View.

Table 3. Import Example of How to Populate EIM_ASSET

Field to Populate	Description
OWNER_ACCNT_BU	The organization of which the account is part.
OWNER_ACCNT_LOC	The account site for the related asset.
OWNER_ACCNT_NAME	The account’s actual name.
AST_ASSET_NUM	The product’s serial number.
AST_PROD_BU	Can be specified as “Default Organization” if necessary.
AST_PROD_NAME	The product’s actual name.

Index

D

E

F

G

H

I

P

Q

R

S

T

U

V

W

X