



SIEBEL⁷
eBusiness

**SIEBEL ENTERPRISE
INTEGRATION MANAGER
ADMINISTRATION GUIDE**

MIDMARKET EDITION

VERSION 7.5

12-BCK98R

SEPTEMBER 2002

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2002 Siebel Systems, Inc.
All rights reserved.
Printed in the United States of America

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior agreement and written permission of Siebel Systems, Inc.

The full text search capabilities of Siebel eBusiness Applications include technology used under license from Hummingbird Ltd. and are the copyright of Hummingbird Ltd. and/or its licensors.

Siebel, the Siebel logo, TrickleSync, TSQ, Universal Agent, and other Siebel product names referenced herein are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

Supportsoft™ is a registered trademark of Supportsoft, Inc. Other product names, designations, logos, and symbols may be trademarks or registered trademarks of their respective owners.

U.S. GOVERNMENT RESTRICTED RIGHTS. Programs, Ancillary Programs and Documentation, delivered subject to the Department of Defense Federal Acquisition Regulation Supplement, are “commercial computer software” as set forth in DFARS 227.7202, Commercial Computer Software and Commercial Computer Software Documentation, and as such, any use, duplication and disclosure of the Programs, Ancillary Programs and Documentation shall be subject to the restrictions contained in the applicable Siebel license agreement. All other use, duplication and disclosure of the Programs, Ancillary Programs and Documentation by the U.S. Government shall be subject to the applicable Siebel license agreement and the restrictions contained in subsection (c) of FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987), or FAR 52.227-14, Rights in Data—General, including Alternate III (June 1987), as applicable. Contractor/licensor is Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404.

Proprietary Information

Siebel Systems, Inc. considers information included in this documentation and in Siebel eBusiness Applications Online Help to be Confidential Information. Your access to and use of this Confidential Information are subject to the terms and conditions of: (1) the applicable Siebel Systems software license agreement, which has been executed and with which you agree to comply; and (2) the proprietary and restricted rights notices included in this documentation.

Contents

Introduction

How This Guide Is Organized	10
Additional Documentation	10
What's New	10
Revision History	12

Chapter 1. Siebel Enterprise Integration Manager: An Overview

EIM Functions	14
Import New and Revised Data into Siebel Base Tables	14
Delete Data from Siebel Base Tables	14
Export Data from Siebel Base Tables	15
Merge Data in Siebel Base Tables	15
Process Flow Between EIM and Other Databases	16

Chapter 2. Siebel Interface Tables

Interface Tables	18
Interface Table Columns	19
File Attachment Columns	20
Organization Columns	21
Interface Table and Column Mappings	21
Explicit Primary Mappings	23
Viewing Interface Table Mappings to Base Tables	26
Viewing Interface Column Mappings to Base Tables	27

Viewing Base Table Mappings to Interface Tables	29
Interface Table Mappings to Base Tables Without User Keys	31
Party Model	34

Chapter 3. EIM Configuration File

Preparing the EIM Configuration File	38
EIM Configuration File Parameters	39
Syntax for INSERT ROWS and UPDATE ROWS	43
Header Section	44
Process Sections	45
Extended Parameters	50
Sample SQL Scripts	54
DB2 Sample SQL Script	54
MS SQL Sample SQL Script	55
Oracle Sample SQL Script	56

Chapter 4. Importing Data

General Steps to Import Data	57
Import Process	61
Preparing the Interface Tables for Import Processing	63
Recommended Import Order	63
Initial Values for Special Columns	64
Initial Values for File Attachment Columns	65
Adjusting the Case of Values	65
Editing the Configuration File for Import Processing	66
Header Section	66
Process Section	66
Header and Process Parameters	68
Working with EIM for Import Processing	73
Importing Initial Data	73
Importing Changes to Existing Data	75
Checking for Duplicate Reporting Relationships	77

Importing Marketing Responses	77
Importing Private Contacts	77
Importing Solutions	77
Importing Call Lists	78
Importing Addresses	78
Importing Industry Codes	79
Importing Mobile Client Data	79
Importing Opportunities	79
Opportunity (S_OPTY) Columns Manually Denormalized from Revenue (S_REVN)	80
Child LOVs are Bounded Consistent with the Application	81
Importing File Attachments	81
Updating File Attachments	82
Importing into Base Tables that Contain the BU_ID Column	83
Importing Large Databases	84
Running an Import Process	86
Checking Import Results	86

Chapter 5. Exporting Data

Export Process	89
Preparing the Interface Tables for Export Processing	91
Editing the Configuration File for Export Processing	93
Header Section	93
Process Section	93
Header and Process Parameters	94
Working with EIM for Export Processing	96
Exporting All Data Rows	96
Exporting Selected Data Rows	96
Running an Export Process	97
Exporting Names from S_BU	97
Checking Export Results	98
Extracting Data from the Interface Tables	98

Chapter 6. Deleting Data

Delete Process Overview	99
Delete Process	100
Preparing the Interface Tables for Delete Processing	102
Editing the Configuration File for Delete Processing	103
Header Section	103
Process Section	103
Header and Process Parameters	104
Working with EIM for Delete Processing	107
Deleting All Data Rows	107
Deleting Data Rows Identified by User Key Values	108
Deleting File Attachments	108
Deleting from S_NOTE and S_*_SKILL_IT Tables	109
Handling Aborts of EIM Delete Processing	110
Cascade Behavior of EIM Deletion	110
Creating Temporary Indexes	111
Running a Delete Process	124
Checking Delete Results	124

Chapter 7. Merging Data

Merge Overview	125
Merge Process	126
Preparing the Interface Tables for Merge Processing	127
Editing the Configuration File for Merge Processing	129
Header Section	129
Process Section	129
Header and Process Parameters	130
Working with EIM for Merge Processing	131
Creating Temporary Indexes	131
Updating Affected Rows	131
Handling Aborts of EIM Merge Processing	131

Enabling Transaction Logging for Merge Processing	132
Specifying Survivor Records	132
Running a Merge Process	133
Checking Merge Results	133

Chapter 8. Running EIM

Running an EIM Process	135
Viewing the Task Info Log	140
Error Flags	140
SQL Trace Flags	142
Trace Flags	142
Log File Detail Levels	147
Optimizing Performance	148
Improving the Database Layout	150

Chapter 9. Frequently Asked Questions

General Questions	153
Import Process Questions	155
Export Process Questions	161
Delete Process Questions	162
Merge Process Questions	164
Performance Questions	165
Troubleshooting Questions	167

Appendix A. Enterprise Integration Manager Error Messages

EIM Error Codes	171
---------------------------	-----

Appendix B. Siebel File System Cleanup Utility

Cleaning up the File Attachment Directory	181
---	-----

Index

Introduction

As part of the System Administration documentation set, this guide provides information necessary to implement, configure, and administer Siebel Enterprise Integration Manager, MidMarket Edition.

NOTE: All Siebel MidMarket product names include the phrase MidMarket Edition to distinguish this product from other Siebel eBusiness Applications. However, in the interest of brevity, after the first mention of a MidMarket product in this document, the product name is given in abbreviated form. For example, after Siebel Call Center, MidMarket Edition, has been mentioned once, it is referred to simply as Siebel Call Center. Such reference to a product using an abbreviated form should be understood as a specific reference to the associated Siebel MidMarket Edition product, and not any other Siebel Systems offering. When contacting Siebel Systems for technical support, sales, or other issues, note the full name of the product to be sure of its proper identification and handling.

The audience for this guide consists of:

Call Center Administrator	Persons responsible for setting up and maintaining a call center; duties include designing and managing Computer Telephony Integration and SmartScripts.
Database Administrators	Persons who administer the database, including data loading; monitoring, backup, and recovery; space allocation and sizing; and user account management.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons responsible for planning, implementing, and configuring Siebel applications.
Siebel System Administrators	Persons responsible for the whole application implementation, including installing, maintaining, and upgrading Siebel products.

The user should possess skills in SQL, RDBMS, and network connectivity using TCP/IP. Previous experience with application and database software is helpful.

How This Guide Is Organized

Information that is common to every EIM process is in the first few chapters. Each major EIM function (import, export, delete, and merge) has its own chapter.

Additional Documentation

While reading this guide, you should refer often to *Siebel Interface Tables Reference*, as it contains detailed descriptions of the interface tables you need.

What's New

The following functionality is new in this release.

- **S_PARTY Table.** S_PARTY table has been introduced into Siebel Data Model in Siebel 7. The S_PARTY table is the target base table, while S_ORG_EXT, S_CONTACT, S_USER, and S_POSTN now become extension tables of the S_PARTY table. The S_EMPLOYEE table is obsolete in version Siebel 7. These schema changes have a direct effect on EIM behavior. For more information, read [“Party Model” on page 34](#).
- **MISC SQL Parameter.** Siebel 7 introduces a new parameter, MISC SQL. This is used to set certain Primary Child Foreign Keys, such as S_CONTACT.PR_OU_ADDR_ID and S_POSTN.PR_EMP_ID. When using MISC SQL in Siebel 7 to set Primary Child Foreign Keys, EIM does NOT log any transactions for mobile users. Use this parameter only for initial data loading. For more information, read [“Header and Process Parameters” on page 68](#).
- **utleimdiff.exe Utility.** The Siebel data model changes from release to release. Schema changes have an effect on EIM behavior. Use the utleimdiff utility to find differences in all interface tables between two repositories. It also generates a selective report of the EIM tables. The results can be used to help you prepare interface tables for EIM data loading.

- **EIM Table Mapping Wizard.** Siebel Tools includes an EIM Table Mapping wizard to assist in adding extensions to the Data Model:
 - Add new customer columns to existing Siebel tables.
 - Add new extension tables.
 - Add new intersection tables.
- **EIM_PROD_INT_UK.** EIM_PROD_INT_UK in Siebel 7 can be used to update user key columns in S_PROD_INT, such as NAME and VENDR_OU_ID. INTEGRATION_ID is an alternative user key in S_PROD_INT. The EIM engine uses this new user key to update traditional user key columns.
- **LOG TRANSACTIONS TO FILE.** Siebel 7 introduces a new parameter, LOG TRANSACTIONS TO FILE. EIM can now log transactions into DX files stored in the File_System\EIM directory. A marker transaction is created in the S_DOCK_TXN_LOG table. For more information, read [“Header Section” on page 44](#).
- **DELETE MATCHES and EXPORT MATCHES Behavior Changed.** The behavior of these parameters has changed as part of the new S_PARTY model. These parameters can now affect extension tables. These parameters also have a new argument. For more information on DELETE MATCHES, read [“Header and Process Parameters” on page 104](#), and for more information on EXPORT MATCHES, read [“Header and Process Parameters” on page 94](#).
- Previously you could not delete from the S_NOTE* and S_*_SKILL_IT tables because they did not have primary user key. Now you can delete records from S_NOTE* and S_*_SKILL_IT tables without deleting records from the parent tables using EIM_NOTE_DEL and EIM_SKLI_DEL, respectively. For more information, read [“Deleting from S_NOTE and S_*_SKILL_IT Tables” on page 109](#).
- Delete and Merge performance is improved if you create some specific temporary indexes first. For more information, read [“Creating Temporary Indexes” on page 111](#).
- The SUM_* columns on S_OPTY are manually denormalized from the Summary Revenue Item (S_REVN). For more information, read [“Opportunity \(S_OPTY\) Columns Manually Denormalized from Revenue \(S_REVN\)” on page 80](#).

- Previously, if a column is based on a hierarchical LOV type, you would populate the LOV type and also BOUND it for the parent level column. In this release, you populate the LOV type in both parent and child level columns and BOUND both the parent and child level LOVs, consistent with the application. For more information, read [“Child LOVs are Bounded Consistent with the Application” on page 81](#).
- Oracle INSERT APPEND MODE. This new parameter helps avoid deadlocks when running parallel EIM processes. For more information, read [“Process Section” on page 66](#).
- ATTACHMENT DIRECTORY. Specifies the directory to be used for importing attachments. For more information, read [“Header and Process Parameters” on page 68](#).
- CASCADE DELETE ONLY. This new parameter determines how child records are handled when the parent record is deleted. For more information, read [“Header and Process Parameters” on page 104](#).
- EIM SCHEMA CACHE. This caches the column relations. For more information, read [“Header and Process Parameters” on page 104](#).

Revision History

Siebel Enterprise Integration Manager Administration Guide, MidMarket Edition, Version 7.5

Siebel Enterprise Integration Manager: An Overview

1

Siebel Enterprise Integration Manager (EIM) manages the exchange of data between Siebel database tables and other corporate databases. You must use EIM to perform bulk imports, exports, merges, and deletes, because loading data directly into Siebel base tables (the tables targeted to receive the data) is not supported. Due to the complexity of table relationships and Mobile Web Client requirements (if applicable to your implementation), you must use EIM to import data into Siebel base tables.

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, use Siebel VB to insert, update, or delete large amounts of data. For more information on VB methods, read *Siebel Tools Online Help, MidMarket Edition*.

EIM Functions

This guide explains how to configure and use EIM to perform the following functions. Each function is discussed separately in other chapters in this guide.

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, you can load the Siebel database tables with data and file attachments created by external applications. For example, you might import information about product lines and products from an Inventory Control database into the Siebel database.
- As part of maintaining the database, you can update it with information created by external applications. For example, Inventory Control might add a product line using another application, and you might import the information into the Siebel database.
- 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 databases.

For a detailed discussion of the import function, read [Chapter 4, “Importing Data.”](#)

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.

For a detailed discussion of the delete function, read [Chapter 6, “Deleting Data.”](#)

Export Data from Siebel Base Tables

When initially implementing a non-Siebel application, you can export data from the Siebel database tables for use by that application. For example, you might export employee information to a corporate sales commission application.

For a detailed discussion of the export function, read [Chapter 5, “Exporting Data.”](#)

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.

For a detailed discussion of the merge function, read [Chapter 7, “Merging Data.”](#)

Process Flow Between EIM and Other Databases

For each EIM process, complete the following sequential steps:

- 1 Prepare the interface tables.** For inserts and merges, load the data into the interface tables. For deletes and exports, make sure the interface tables are clear. For more information, read [Chapter 2, “Siebel Interface Tables.”](#)
- 2 Edit the EIM configuration file.** The configuration file contains the information EIM needs so it can process the data. For more information, read [Chapter 3, “EIM Configuration File.”](#)
- 3 Run EIM.** The EIM process runs on a Siebel Server. You can start the process from a command line or from within the Siebel application. For more information, read [Chapter 8, “Running EIM.”](#)
- 4 Check results.** Check the Task Info log to make sure the process ran as expected.

[Figure 1](#) shows the EIM data flow of data. The Siebel database contains Siebel interface tables and Siebel base tables. EIM transports the data between the interface tables and the base tables. To transport the data from the interface tables to a non-Siebel database, use any SQL ancillary program utility.

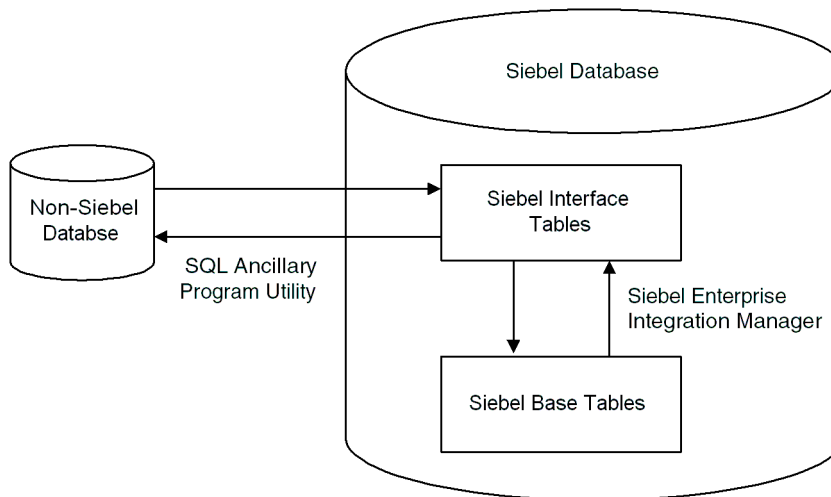


Figure 1. Process Flows Between Siebel Database and Other Databases

This chapter discusses Siebel interface tables and how EIM uses them. Siebel interface tables are intermediate database tables that act as a staging area between the base tables in the Siebel Database and other databases. Before EIM can be used in most cases, you or a database administrator must populate the interface tables with data to be processed by EIM. You then invoke EIM to process this data. EIM makes multiple passes through the tables to complete the specified process. 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.

For information on specific data and file attachments that EIM can process, the names of the interface tables, the target base tables mapped to the interface tables, and any secondary tables associated with the target tables, read *Siebel Interface Tables Reference*.

Interface Tables

All interface tables used by EIM have the prefix EIM_ (such as EIM_ACCOUNT). These interface tables support organizations, so they can be used for all EIM processes. For information on the names of the interface tables, the target base tables mapped to these interface tables, and any secondary tables associated with the target tables, read *Siebel Interface Tables Reference*.

When using interface tables for EIM processes, keep in mind the following points:

- Previous versions of EIM used a different set of interface tables, identified by the prefix S_ and the suffix _IF. These tables are still present 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.
- If you are importing into base tables that use the UTC (Universal Time Coordinate), you or a database administrator must convert the local time in the data into UTC before loading data into the interface tables.

Interface 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 interface table. These columns are described in this section.

NOTE: For all EIM processes, you must populate the ROW_ID, IF_ROW_STAT, and IF_ROW_BATCH_NUM columns in the interface tables. For merge processes, you also need to populate the IF_ROW_MERGE_ID column. Do not populate these required columns with spaces.

ROW_ID. For an interface 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 interface tables are not the ROW_ID values that are assigned to the row when it is loaded into the base table. The EIM-generated ROW_ID has a ##-##-#### format. Rows created through the Siebel application or from an upgrade process have 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 interface table.

IF_ROW_MERGE_ID. EIM uses the value in this row during merge processing.

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 interface 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. For more information, read [“Checking Import Results” on page 86](#).

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.

After processing, if a row was successfully processed to completion the row contains a zero (0).

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 interface tables.

For detailed information about each interface table (including column names, required initial values, and data types), read *Siebel Interface Tables Reference*.

File Attachment Columns

Three interface table columns must be populated to import file attachments. [Table 1](#) describes these columns and uses the attachment file budget99.doc as an example.

Table 1. 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 URL. Otherwise the rows can not be imported.	FILE_SRC_TYPE = "FILE"

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.

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.

Interface Table and Column Mappings

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

- Interface table mappings to Siebel base tables
- Interface column mappings to Siebel base table columns
- Siebel base table mappings to interface tables

Some base tables may not be mapped to a corresponding interface table. In such cases, use Siebel VB to load data into these base tables and inform Siebel Technical Services regarding the missing mapping. For information on using Siebel VB, read *Siebel Tools Online Help, MidMarket Edition*.

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 interface tables only if these columns share the same base table. To map interface table extensions to base table extensions, you must specify which column the extended field points to in the base table. For more information on Database Extensibility, read *Siebel Tools Reference, MidMarket Edition*.

Some EIM table mappings (usually to the target base table) are provided only as a common parent to nontarget EIM table mapping. In such EIM table mappings, only the user key columns are mapped. Except for updating primary child foreign key columns, EIM does not support inserting and updating rows using these EIM Table Mappings. For stability of EIM when using these interface 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

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

An example of this type of EIM table mapping is mapping from the EIM_OPTY_DTL interface table to the S_OPTY base table. One exception is when you want to update the primary child 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 child foreign keys in S_ORG_EXT if the explicit primary mappings exist, such as in S_ORG_EXT.PR_INDUST_ID, the explicit primary mapping is contained in the table mapping of S_ORG_INDUST. For more information, read [“Explicit Primary Mappings” on page 23](#). In this case, you do not want to set UPDATE ROWS = S_ORG_EXT, FALSE in the EIM configuration file. Otherwise, if you do not need to update primary child foreign keys in S_ORG_EXT, then you should set UPDATE ROWS = S_ORG_EXT, FALSE in the EIM configuration file.

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

- Mappings already exist from the interface table into the base table.

- The base table is an extension table and mappings already exist from the interface table into the corresponding base table.

You can map extension columns in interface tables to extension columns in either base tables or in extension tables only if the condition is true.

NOTE: Mappings to base columns are not supported.

For example, you could create a new column EIM_ACCOUNT.X_CUST_NUM and map this 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. For more information, read *Siebel Tools Reference, MidMarket Edition*.

For information on interface table mappings, read *Siebel Interface Tables Reference*.

Explicit Primary Mappings

The Siebel data model uses primary child foreign keys (or primaries) to point from a parent base table to a child base table. Primaries allow 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.

Primary child 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 child foreign key are mapped to the same interface table, then you should see explicit primary mapping for this primary child foreign key under the table mapping of the primary child table.

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

Siebel Interface Tables

Interface Table and Column Mappings

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 interface table. For example, if you are importing a new account with three addresses using the EIM_ACCOUNT interface table, you can explicitly set one of these addresses as the primary business address by populating the primary flag column ACC_PR_ADDR with Y, or as the primary billing address by populating the primary flag column ACC_PR_BL_ADDR with Y. [Table 2](#) shows an example of setting the primary business address for A. K. Parker Distribution to Menlo Park, and its billing address to San Francisco.

NOTE: The flag columns for explicit primary mappings usually follow the XXX_PR_XXX naming convention.

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 business address—then EIM ignores this explicit primary mapping and sets the primary implicitly, such as choosing the address with min(ROW_ID) as the primary business address.

For information on the explicit primary columns for each EIM_ type interface table, read *Siebel Interface Tables Reference*.

Table 2. Explicit Primary Mapping for an Account

NAME	LOC	ACCNT_BU	ADDR	City	ACC_PR_ADDR	ACC_PR_BL_ADDR
A. K. Parker Distribution	CA	Americas	1000 Industrial Way	Menlo Park	Y	
A. K. Parker Distribution	CA	Americas	322 Arkansas Street	San Francisco		Y
A. K. Parker Distribution	CA	Americas	888 El Camino Real	San Mateo		

Table 3 shows an excerpt from *Siebel Interface Tables Reference*. It shows that when you use the EIM_ACCOUNT interface table, you can use the ACC_PR_ADDR column to mark an address as the primary address and the ACC_PR_BL_ADDR column to mark an address as the primary bill to address.

NOTE: Table 3 is provided as an example only; for a full listing of the mappings supported by the EIM_ACCOUNT interface table, read *Siebel Interface Tables Reference*.

Table 3. EIM_ACCOUNT¹

Base Table	Base Column	UK	Req	Base Column Description	IF Source Column	Data Type	Length
S_ORG_EXT*	NAME	1	Y	Name	NAME	Varchar	100
	LOC	2	N	Site	LOC	Varchar	50
	BU_ID	3	Y	Business Unit Id	ACCNT_BU	Varchar	50
	PR_ADDR_ID		N	Primary Address	ACC_PR_ADDR	Char	1
	PR_BL_ADDR_ID		N	Primary Bill To Address	ACC_PR_BL_ADDR	Char	1
	PR_BL_OU_ID		N	Primary Billing Organization Id	ACC_PR_BL_OU	Char	1
S_ADDR_ORG	ADDR		Y	Address	ADDR_ADDR	Varchar	200
	CITY		Y	City	ADDR_CITY	Varchar	50
	STATE		N	State	ADDR_STATE	Varchar	10
	ZIPCODE		N	Zipcode	ADDR_ZIPCODE	Varchar	30
	OU_ID	2	Y	Account	ACCNT_BU	Varchar	50
					LOC	Varchar	50
					NAME	Varchar	100

1. Excerpt from *Siebel Interface Tables Reference* showing explicit primary mapping for the EIM_ACCOUNT interface table.

Viewing Interface Table Mappings to Base Tables

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

To view interface table mappings to base tables

- 1** Start Siebel Tools.
- 2** In Object Explorer, click the Types tab.
- 3** Click EIM Interface Table.
- 4** In the EIM Tables window, select the interface 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 now displays all base table mappings for the selected interface 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 interface tables. For more information, read *Siebel Tools Reference, MidMarket Edition*.

Figure 2 shows an example of viewing the interface table mappings for the EIM_ACCOUNT interface table. This screen image from Siebel Tools shows a list of EIM tables, with the EIM_ACCOUNT table selected. Below that list is a list of the fields within the selected tables. The field list includes some of the mappings.

The screenshot displays two windows from Siebel Tools. The top window, titled 'EIM Tables', contains a table with columns: W, Name, Changed, Project, User Name, and Alias. The bottom window, titled 'EIM Table Mappings', contains a table with columns: W, Name, Changed, Destination Table, Second Row, EIM Exists Proc Column, and EIM_ROW_ID Proc Column.

W	Name	Changed	Project	User Name	Alias
>	EIM_ACCOUNT		EIM Accounts and Q...	EIM_ACCOUNT	
	EIM_ACCOUNT1		EIM Accounts and Q...	EIM_ACCOUNT1	
	EIM_ACCOUNT2		EIM Accounts and Q...	EIM_ACCOUNT2	

W	Name	Changed	Destination Table	Second Row	EIM Exists Proc Column	EIM_ROW_ID Proc Column
>	S_ACCNT_POSTN		S_ACCNT_POSTN		T_ACCNTPOST_EXS	T_ACCNTPOST_RID
	S_ADDR_ORG		S_ADDR_ORG		T_ADDR_ORG_EXS	T_ADDR_ORG_RID
	S_ORG_BU		S_ORG_BU		T_ORG_BU_EXS	T_ORG_BU_RID
	S_ORG_EXT		S_ORG_EXT		T_ORG_EXT_EXS	T_ORG_EXT_RID
	S_ORG_REL		S_ORG_REL		T_ORG_REL_EXS	T_ORG_REL_RID
	S_PARTY		S_PARTY		T_PARTY_EXS	T_PARTY_RID
	S_PARTY_PER		S_PARTY_PER		T_PARTY_PER_EXS	T_PARTY_PER_RID

Figure 2. Viewing Interface Table Mappings to Base Tables

Viewing Interface Column Mappings to Base Tables

Use Siebel Tools to view column mappings to base tables.

To view interface column mappings to base tables

- 1 Complete [Step 1](#) through [Step 6](#) of “[To view interface 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.

5 To map columns, navigate to the source of the information and fill in the target information.

For more information, read *Siebel Tools Reference, MidMarket Edition*.

Figure 3 shows an example of viewing column mappings for the S_ADDR_ORG base table. This screen image from Siebel Tools shows a list of EIM table mappings, with the S_ADDR_ORG table selected. Below that list is a list of attribute mappings within the selected table.

The screenshot displays two windows from Siebel Tools. The top window, titled "EIM Table Mappings", shows a list of mappings for the selected S_ADDR_ORG table. The bottom window, titled "Attribute Mappings", shows a list of attribute mappings for the selected table.

W	Name	Changed	Destination Table	Second Row	EIM Exists Proc Column	EIM ROW_ID Proc Column
	S_ACCNT_POSTN		S_ACCNT_POSTN		T_ACCNTPOST_EXS	T_ACCNTPOST_RID
▶	S_ADDR_ORG		S_ADDR_ORG		T_ADDR_ORG_EXS	T_ADDR_ORG_RID
	S_ORG_BU		S_ORG_BU		T_ORG_BU_EXS	T_ORG_BU_RID
	S_ORG_EXT		S_ORG_EXT		T_ORG_EXT_EXS	T_ORG_EXT_RID

W	Name	Changed	Interface Table Data Column	Base Table Attribute Column	Export Only	Inactive	Comm
▶	ACTIVE_FLG		ADDR_ACTIVE_FLG	ACTIVE_FLG			Attribu
	ADDR		ADDR_ADDR	ADDR			Attribu
	ADDR_LINE_2		ADDR_ADDR_LINE_2	ADDR_LINE_2			Attribu
	ADDR_LINE_3		ADDR_ADDR_LINE_3	ADDR_LINE_3			Attribu
	ADDR_NAME		ADDR_ADDR_NAME	ADDR_NAME			Attribu
	ADDR_NUM		ADDR_ADDR_NUM	ADDR_NUM			Attribu
	ADDR_TYPE_CD		ADDR_ADDR_TYPE_CD	ADDR_TYPE_CD			Attribu
	BL_ADDR_FLG		ADDR_BL_ADDR_FLG	BL_ADDR_FLG			Attribu
	CITY		ADDR_CITY	CITY			Attribu
	COMMENTS		ADDR_COMMENTS	COMMENTS			Attribu
	COUNTRY		ADDR_COUNTRY	COUNTRY			Attribu
	COUNTY		ADDR_COUNTRY	COUNTY			Attribu
	DFLT_SHIP_PRIO_CD		ADDR_DFLTSHIPPRIOC	DFLT_SHIP_PRIO_CD			Attribu
	DISA_CLEANSE_FLG		ADDR_DISACLEANSEFL	DISA_CLEANSE_FLG			Attribu
	EMAIL_ADDR		ADDR_EMAIL_ADDR	EMAIL_ADDR			Attribu
	FAX_PH_NUM		ADDR_FAX_PH_NUM	FAX_PH_NUM			Attribu
	INTEGRATION2_ID		ADDR_INTEGRATION2I	INTEGRATION2_ID			Attribu
	INTEGRATIONS_ID		ADDR_INTEGRATION3I	INTEGRATIONS_ID			Attribu
	INTEGRATION_ID		ADDR_INTEGRATIONID	INTEGRATION_ID			Attribu
	LATITUDE		ADDR_LATITUDE	LATITUDE			Attribu
	LONGITUDE		ADDR_LONGITUDE	LONGITUDE			Attribu
	MAIN_ADDR_FLG		ADDR_MAIN_ADDR_FLG	MAIN_ADDR_FLG			Attribu
	NAME_LOCK_FLG		ADDR_NAME_LOCK_FLG	NAME_LOCK_FLG			Attribu

Figure 3. Viewing Column Mappings

Viewing Base Table Mappings to Interface Tables

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

To view base table mappings to base tables

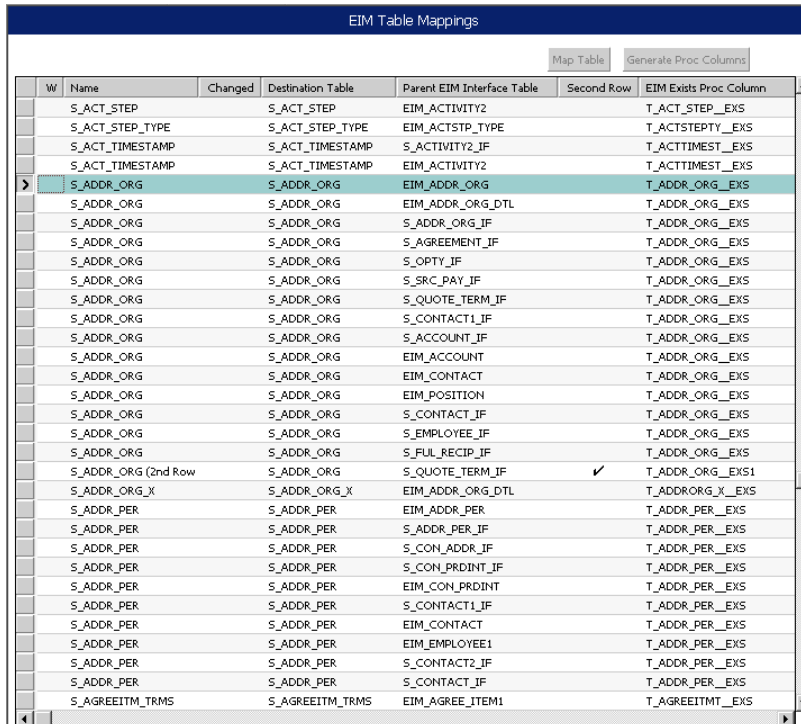
- 1** Start Siebel Tools.
- 2** In Object Explorer, click the Flat tab.
- 3** Click EIM Table Mapping.
- 4** In the EIM Table Mappings window, select a base table.
- 5** The interface table to which the base table is mapped is shown in the Parent EIM Interface Table field.

Some base tables may be mapped to more than one interface table.

Siebel Interface Tables

Interface Table and Column Mappings

Figure 4 shows an example of viewing the base table mappings for the S_ADDR_ORG base table. Note that the S_ADDR_ORG base table maps to many interface tables.



The screenshot displays the 'EIM Table Mappings' window. At the top, there are two buttons: 'Map Table' and 'Generate Proc Columns'. Below the buttons is a table with the following columns: 'W', 'Name', 'Changed', 'Destination Table', 'Parent EIM Interface Table', 'Second Row', and 'EIM Exists Proc Column'. The table lists various mappings, with 'S_ADDR_ORG' highlighted in the 'Name' column. The 'Parent EIM Interface Table' column shows 'EIM_ADDR_ORG' for the highlighted row, and 'EIM_ADDR_ORG_DTL' for the row below it. The 'EIM Exists Proc Column' column shows 'T_ADDR_ORG_EXS' for most rows, and 'T_ADDRORG_X_EXS1' for the row with 'S_ADDR_ORG (2nd Row)'.

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_ACTSTEPY_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 Interface Tables

Interface Table Mappings to Base Tables Without User Keys

Some interface tables contain table mappings to base tables without users keys. When using these interface tables, you should note the EIM behavior for the relevant process as described in this section:

- Import processes, read [“Importing Data into Base Tables Without User Keys” on page 33.](#)
- Update processes, read [“Updating Data in Base Tables Without User Keys” on page 33.](#)
- Export processes, read [“Exporting Data from Base Tables Without User Keys” on page 33.](#)
- Delete processes, read [“Deleting Data from Base Tables Without User Keys” on page 34.](#)
- Merge processes, read [“Merging Data in Base Tables Without User Keys” on page 34.](#)

[Table 4](#) lists the interface tables containing table mappings to base tables without user keys.

Table 4. Interface Tables Containing Table Mappings to Base Tables Without User Keys

Interface Table	Target Base Table Without User Key
EIM_ACC_SRC_DTL	S_NOTE_ACC_SRC
EIM_ACCNT_DTL	S_NOTE_ACCNT
	S_ORG_SKILL_IT
EIM_ACCSRCPIDTL	S_NOTE_ACCSRCPI
EIM_ACT_DTL	S_ACT_SKILL_IT
	S_NOTE_ACT
EIM_ASGN_GRP	S_ASGN_RESULT
EIM_ASSET_DTL	S_NOTE_ASSET
EIM_BASELN_DTL	S_NOTE_BASELINE

Table 4. Interface Tables Containing Table Mappings to Base Tables Without User Keys

Interface Table	Target Base Table Without User Key
EIM_BU	S_BU_SKILL_IT
EIM_CON_DTL	S_CON_SKILL_IT
	S_NOTE_CON
EIM_CON_PI_DTL	S_NOTE_CON_PI
EIM_CONSUM_DTL	S_NOTE_CONSUME
EIM_DCP_DTL	S_DCP_SKILL_IT
	S_NOTE_DCP
EIM_DEFECT_DTL	S_NOTE_DEFECT
EIM_EMP_DTL	S_EMP_SKILL_IT
EIM_GROUP_DTL	S_NOTE_ORGGROUP
	S_ORGGRP_SKLI
EIM_INVC_DTL	S_NOTE_INVOICE
EIM_NOTE	S_NOTE
EIM_OPTY_DTL	S_NOTE_OPTY
	S_OPTY_SKILL_IT
EIM_ORDER_ITEM1	S_NOTE_ORDER_IT
EIM_ORDER1	S_NOTE_ORDER
EIM_PDSHIP_DTL	S_NOTE_SHIPMENT
EIM_POSTN_DTL	S_POS_SKILL_IT
EIM_PRDINT_DTL	S_NOTE_PROD_INT
EIM_PROJECTDTL	S_NOTE_PROJ
	S_PROJ_SKILL_IT
EIM_PROJITMDTL	S_NOTE_PROJITEM

Table 4. Interface Tables Containing Table Mappings to Base Tables Without User Keys

Interface Table	Target Base Table Without User Key
EIM_PROJRSRCDTL	S_NOTE_PROJRSRC
	S_PROJRSRC_SKLI
EIM_QUO_IT_DTL	S_NOTE_QUOTE_IT
EIM_QUOTE_DTL	S_NOTE_QUOTE
EIM_SR_DTL	S_NOTE_SR
	S_SR_SKILL_IT
EIM_SRC_DTL	S_CAMP_SKILL_IT
	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

Importing Data into Base Tables Without User Keys

Importing 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

Updating on base tables without user keys does not work, because EIM cannot identify the record to update.

Exporting Data from Base Tables Without User Keys

Exporting works using the EXPORT MATCHES and EXPORT ALL ROWS parameters. If you set the EXPORT ALL ROWS parameter to TRUE, all rows are exported from the target and secondary base tables.

Deleting Data from Base Tables Without User Keys

Deleting works if you set the DELETE ALL ROWS parameter to TRUE. All rows are deleted from the target and secondary base tables (and children tables). If you are using the DELETE MATCHES parameter, delete works.

Merging Data in Base Tables Without User Keys

Merging does not work on base tables without user keys.

Party Model

The party model is a means of unifying all access to data about relationships. This covers relationships between your company and people (contacts, employees, partner employees, users) and other businesses (accounts, divisions, organizations, partners). The base table for all such access is S_PARTY. Related tables are implicitly joined as extension tables. [Table 5](#) lists the extension tables and their corresponding EIM interface tables.

Table 5. S_PARTY Extension Tables and Corresponding EIM Interface Tables

Data Type	Extension Table to S_PARTY	EIM Interface Table
Accounts	S_ORG_EXT	EIM_ACCOUNT
Business Units	S_BU	EIM_BU
Contacts	S_CONTACT	EIM_CONTACT
Employees	S_CONTACT	EIM_EMPLOYEE
Groups	S_ORG_GROUP	EIM_GROUP
Organizations	S_ORG_BU	EIM_ORG_BU
Positions	S_POSTN	EIM_POSITION
Users	S_USER	EIM_USER

Because the extension tables are implicitly joined to S_PARTY, you do not need to configure anything to access them through S_PARTY. For more detail on data types and related tables, read the appendix on Siebel Interface Table Mappings in *Siebel Interface Tables Reference*.

Some data types have a many to many relationship. For example, any contact can be associated with multiple accounts or partners. To model these relationships there are preconfigured intersection tables: S_PARTY_PER and S_PARTY_REL. Use S_PARTY_REL to implement relationships between parties in the S_PARTY table. In this case, records in S_PARTY are both parent (PARTY_ID) and child (REL_PARTY_ID).

Use S_PARTY_PER to implement relationships between members:

- Access groups and members
- Accounts and contacts
- Employees and positions
- User lists and users

Siebel Interface Tables

Party Model

The chapter covers the generic use of EIM configuration files and is organized into the following sections:

- [“Preparing the EIM Configuration File” on page 38](#)
- [“EIM Configuration File Parameters” on page 39](#)
- [“Sample SQL Scripts” on page 54](#)

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

Preparing the EIM Configuration File

EIM reads a 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 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 GUI, or the configuration file. You must specify the correct code for the target database in one of these locales.

EIM accepts parameter values from three sources:

- The DOS command line (either entered by the user or read from a BAT file) that invokes the import 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 parameter discussed later in this chapter. Alternatively, you can create multiple EIM configuration files (IFB) and specify which one EIM should use.

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. 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, read the relevant chapter for each process:

- For an import process, read [“Editing the Configuration File for Import Processing” on page 66](#).
- For an export process, read [“Editing the Configuration File for Export Processing” on page 93](#).
- For a delete process, read [“Editing the Configuration File for Delete Processing” on page 103](#).
- For a merge process, read [“Editing the Configuration File for Merge Processing” on page 129](#).

[Table 6 on page 40](#) lists the parameters used in the header and process sections of the EIM configuration file. Each parameter is categorized based on the specific type of EIM process in which it is used:

- General process parameters may be used in all EIM processes.
- Import process parameters may be used only in an import process.
- Export process parameters may be used only in an export process.
- Delete process parameters may be used only in a delete process.

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

Table 6. EIM Configuration File Parameters

Command	Description
Header	
CONNECT	ODBC System Data Source
LOG TRANSACTIONS TO FILE	Log transactions to file or table; TRUE/FALSE toggle; default is TRUE
PASSWORD	Database password ¹
PROCESS	Initial/main process section to run
TABLEOWNER	Database tableowner, defined during installation
USERNAME	Database/Employee logon ¹
[Siebel Interface Manager]	Header section must use this reserved name
General Processes	
BATCH	IF_ROW_BATCH_NUM value to run against
COMMIT EACH PASS	Commit after each EIM pass; default is TRUE
COMMIT EACH TABLE	Commit after each base table; default is TRUE
IGNORE BASE COLUMNS	Do not process these columns
IGNORE BASE TABLES	Do not process these tables
INCLUDE	Subprocess to execute
LOG TRANSACTIONS	Default value depends on system preference
ONLY BASE TABLES	Process only these tables
ROLLBACK ON ERROR	Error rollback behavior; default is FALSE
SESSION SQL	Preprocess SQL statement
SKIP BU_ID DEFAULT	If virtual null key is to be skipped for the BU_ID column; default is FALSE
TABLE	Interface (IF) table for process

Table 6. EIM Configuration File Parameters

Command	Description
TRANSACTION SQL	Post-commit SQL statement
TYPE	IMPORT, EXPORT, DELETE, MERGE, SHELL
UPDATE STATISTICS	Updates statistics of interface tables; default is TRUE
USE ESSENTIAL INDEX HINTS	TRUE/FALSE toggle; For MS SQL and Oracle only; default is TRUE
USE INDEX HINTS	TRUE/FALSE toggle; For Oracle only; default is FALSE
USE SYNONYMS	TRUE/FALSE toggle; default is TRUE
Import Process	
COMMIT OPERATIONS	Docking Log row commit frequency; default is 0
DEFAULT COLUMN	Default for IF column; DEFAULT COLUMN = ORG_CD, "Federal"
FILTER QUERY	SQL pre-process filter query fragment; FILTER QUERY = (ACCNT_NUM "1500")
FIXED COLUMN	Set IF column to this literal; FIXED COLUMN = ORG_CD, "Commercial"
INSERT ROWS	Optional base table, TRUE/FALSE toggle; default is TRUE. For more information, read "Syntax for INSERT ROWS and UPDATE ROWS" on page 43.
MISC SQL	Set primaries in Step 11 on page 62 of import process
NET CHANGE	Do not update with NULL values; default is TRUE
ONLY BASE COLUMNS	Process only these columns
TRIM SPACES	Toggles space trimming; default is TRUE
UPDATE ROWS	Optional base table, TRUE/FALSE toggle; default is TRUE. For more information, read "Syntax for INSERT ROWS and UPDATE ROWS" on page 43.
Export Process	
CLEAR INTERFACE TABLE	Preprocess deletion flag; default is TRUE
EXPORT ALL ROWS	Export all rows in tables; default is FALSE

Table 6. EIM Configuration File Parameters

Command	Description
EXPORT MATCHES	WHERE clause fragment; EXPORT MATCHES = (NAME LIKE "GEN%")
Delete Process	
DELETE ALL ROWS	Used for deleting all rows in table; use with caution; default is FALSE
DELETE EXACT	Delete using user key matching algorithm with rows in IF table; default is FALSE
DELETE MATCHES	SQL WHERE fragment deletion criteria; DELETE MATCHES = S_ORG_EXT, (NAME LIKE "TST_ACCT%")
DELETE ROWS	Prevents deletion of rows; DELETE ROWS = S_ADDR_ORG, FALSE
UPDATE ROWS	Prevents updating of foreign key references; UPDATE ROWS = S_OPTY, FALSE. For more information, read "Syntax for INSERT ROWS and UPDATE ROWS" on page 43.

1. This value is not used for access authentication or as a security measure. EIM acquires this value from the component parameters. These values are not used unless the parameters are not set at the enterprise or component level.

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

CAUTION: If continuing a parameter definition to multiple lines in the IFB file, make certain that the backslash character (\) is the last character on the line. If the backslash is followed by a space, the space character is "escaped" and the new line character then terminates the parameter definition. You may get an error message indicating the parameter definition is incomplete.

Syntax for INSERT ROWS and UPDATE ROWS

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_ADD_ORG, TRUE
```

Header Section

This section describes the parameters that appear in the header section of the EIM configuration file.

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

```
[Siebel Interface Manager]
```

PROCESS. Identifies the specific process to run during this invocation of EIM. The named process must be defined in the process section of this file.

CONNECT. The ODBC source name for connecting to the database server.

USERNAME. The database logon name for this process. 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.

PASSWORD. The database password for this process. 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 `svrmgr`. 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, it then looks in the IFB file. If it cannot find the username and password in those places, it 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.

TABLEOWNER. The database logon name that owns the tables to be operated on; used as the prefix for table names.

LOG TRANSACTIONS TO FILE. Controls where transactions are logged. This parameter must be in the header section. The default value is TRUE, which logs transactions into a file and also places one marker transaction in the S_DOCK_TXN_LOG table. Log files have a DX extension and are saved in the file system's eim directory. Set this parameter to FALSE to log transactions in the S_DOCK_TXN_LOG 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, read *Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition*.

TIP: If you set LOG TRANSACTIONS TO FILE to TRUE, then SET BASED LOGGING should be set to FALSE. If SET BASED LOGGING is set to TRUE, then the log contains only one row per set. There is not much value in writing that one row to a file.

Process Sections

This section describes only the general process parameters generic to all EIM processes 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.

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 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:

_ : - \$ % / +

TYPE. Specifies the type of process being defined (IMPORT, EXPORT, DELETE, MERGE, or SHELL). 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.

BATCH. Specifies a required batch number for this process. Use this number to identify the set of rows to load from the interface tables for this specific process. It corresponds to the value in the interface table column IF_ROW_BATCH_NUM and must be a positive integer between 0 and 999999999999999 (up to 15 digits, 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
```

TABLE. Specifies the name of an interface table used in this process. Multiple TABLE parameters may be used to define a process using more than one table:

```
TYPE = EXPORT
BATCH = 101
TABLE = EIM_ACCOUNT
TABLE = EIM_ACCNT_DTL
EXPORT MATCHES = S_ORG_EXT, (NAME > 'A')
```

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

LOG TRANSACTIONS. If this parameter is TRUE, EIM logs remote transactions when mobile clients synchronize. If this parameter is set to FALSE, changes are not logged. In general, when users load data into the HQ database for the first time, this parameter should be set to FALSE. Default value depends on system preference setting.

SKIP BU_ID DEFAULT. Specifies if the virtual null key is to be skipped for the BU_ID column. The 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.

SESSION SQL. Specifies a user-defined SQL statement to be sent to the database server before other SQL statements for this process. This string is sent uninterpreted 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 uninterpreted to the database and may cause data loss for Siebel Remote.

TRANSACTION SQL. 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:

- Adding the TRANSACTION SQL parameter in the configuration file
- Using 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:

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

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

To define the rollback segment in the configuration file

- Add the following line 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 the Components screen.
- 2 Click the Component Parameters view tab.
- 3 In the Server Components list, select Enterprise Integration Mgr.
- 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, read *Siebel Server Administration Guide, MidMarket Edition*.

UPDATE STATISTICS. Controls whether EIM dynamically updates the statistics of interface tables. The default setting is TRUE.

If you are running EIM on a DB2 database, the account under which EIM runs must have the DB2 CONTROL table privilege on the interface tables. The database installer automatically grants this privilege when creating the tables. However, it may be necessary to regrant this privilege if the interface tables have been modified or recreated. To regrant the CONTROL privilege, use the script named grantstats.bat 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 interface table simultaneously. To avoid this potential problem, set the UPDATE STATISTICS parameter to FALSE.

USE ESSENTIAL INDEX HINTS. For MS SQL and Oracle only. Default value is TRUE. This parameter enables a subset of index hints.

USE INDEX HINTS. Controls whether EIM issues optimizer hints to the underlying database to improve performance and throughput. This parameter is for Oracle only. The default setting is FALSE.

NOTE: These are the suggested settings of the index hints parameters for each RDBMS.

For MS SQL: USE ESSENTIAL INDEX HINTS = TRUE, USE INDEX HINTS = FALSE

For Oracle: USE ESSENTIAL INDEX HINTS = TRUE, USE INDEX HINTS = TRUE.

For DB2: Not applicable.

USING SYNONYMS. 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 setting is TRUE.

Extended 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 Enterprise Operations.
- 2** Click the Component Requests view tab.
- 3** In the Component Requests form, click the menu button and then New Record.
- 4** In the Component/Job field, click the Select button.

The Component/Jobs window opens.

- 5** In the Component/Jobs window, select the Enterprise Integration Mgr component and 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, read *Siebel Server Administration Guide, MidMarket Edition*.

- 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.

The Job Parameters window opens.

- 9** In the Job Parameters window, select Extended Parameters and click OK.

- 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"
```

- Click Save.
- In the Component Requests form, click the menu button and then Submit request.

Figure 5 shows an example of defining extended parameters as described in “To define extended parameters using the GUI” on page 50. This image of the Siebel application shows the Component Requests list with a record selected. Below that is the Components Requests form showing the fields for the selected record.

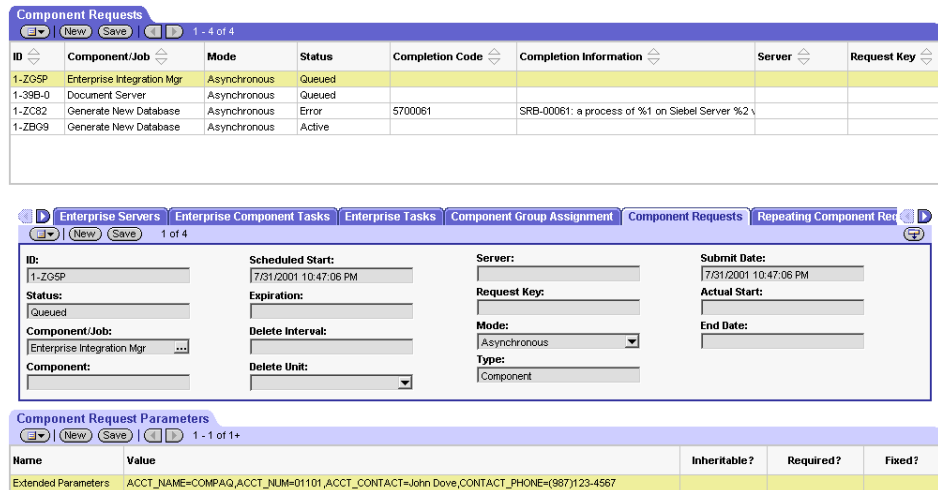


Figure 5. Defining Extended Parameters

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 7](#) lists these predefined extended parameters.

Table 7. Predefined Extended Parameters

Parameter	Description	Example
CURRENT_USER	Logon name of current user.	CURRENT_USER = Customer1
PASSWORD	Password of current user.	PASSWORD = ABC
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 is 10.	MAX_NEST_SUBST = 10

Table 7. Predefined Extended Parameters

Parameter	Description	Example
NUM_IFTABLE_LOAD_CUTOFF	<p>When enabled, EIM loads schema mappings if the value is less than the number of interface tables used in the current IFB file. To enable, set the value to a positive number.</p> <p>When disabled, EIM loads only mappings for interface tables used in the current IFB file. This speeds up the dictionary loading process in EIM. To disable, set the value to -1. This feature is disabled by default.</p>	NUM_IFTABLE_LOAD_CUTOFF = -1
IfbFileName	Name of the IFB file where resolved parameters are stored. (Used for debugging only.)	IfbFileName = TEST
TraceFlags	Contains logs of various EIM operations. Available TraceFlags include 1, 2, 4, 8, and 32. For descriptions of available TraceFlags, read “Trace Flags” on page 142 .	TraceFlags = 2

Sample SQL Scripts

Use the following sample SQL scripts as a starting point for your own scripts. Sample scripts are provided for the following RDBMSs:

- [“DB2 Sample SQL Script”](#)
- [“MS SQL Sample SQL Script” on page 55](#)
- [“Oracle Sample SQL Script” on page 56](#)

DB2 Sample SQL Script

The following sample SQL script is for DB2 users.

```
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','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_PTSHPKKEYCONFLG, 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','Y','Default Organization',
'CrossRoads', 'Account2', 'ACD1', 'AUID1', 'Y', '2000-05-17-
15.40.55.000000');
```

MS SQL Sample SQL Script

The following sample SQL script is for MS SQL users.

```
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_CLEANSER_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','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_PTSHPKKEYCONFLG, CON_SUPPRESSEMAILF, CON_SUPPRESSEMAILF,
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','Y','Default Organization',
'CrossRoads', 'Account2', 'ACD1', 'AUID1', 'Y', '02-FEB-2002')
```

Oracle Sample SQL Script

The following sample SQL script is for Oracle users.

```
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', '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_PTSHPKYCONFLG, 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');
```


This chapter discusses importing data with EIM. Importing data into Siebel base tables is a multistep process. Before you can prepare to import data, your database administrator must first load data from an external database into the interface tables. Subsequently, you need to run an EIM process to read the data in these interface tables and import them into the appropriate Siebel base tables.

General Steps to Import Data

This section describes the general steps to import data into the Siebel database using EIM.

NOTE: This effort may require the time of key personnel, as well as time and resources.

To import data

- 1 Identify and validate the data to be imported:
 - 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:

- Identify the mapping between the data and Siebel base columns. For information on Siebel base table columns, read *Siebel Data Model Reference*.
- Identify the interface table columns that map to these base table columns. To view mappings between interface table columns and base table columns, read [“Interface Table and Column Mappings” on page 21](#), and for information on interface table columns, read *Siebel Interface Tables Reference*.
- Identify the user key columns and make sure that they are populated with uniqueness. For information on user key columns, read *Siebel Data Model Reference*.
- The data to insert in the Interface table can be classified as either special columns (read [“Preparing the Interface Tables for Import Processing” on page 63](#)) and the data you want to import. Data to import can be either new records or updates to existing records. When importing new data, make sure to populate the columns marked “Required” in the interface 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, read *Siebel Interface Tables Reference*.

3 Make sure that your hardware and software environments are ready:

- Before you use Siebel interface tables to import data, the 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, read [“Importing Large Databases” on page 84](#).

4 Back up your existing database.

Before undertaking a 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 directory named input.

If you wish to import file attachments, copy the files to the input directory under the Siebel server root directory. Siebel interface 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, read [“File Attachment Columns” on page 20](#).

6 Load and verify the interface tables.

Request that your database administrator 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 interface tables.

NOTE: Siebel interface tables contain several special columns that must be populated before rows can be imported. For more information, read [“Interface Table Columns” on page 19](#).

- After the interface 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 interface tables for data import, read [“Preparing the Interface Tables for Import Processing” on page 63](#).

7 Edit the EIM configuration file (default.ifb).

This file customizes the behavior of EIM by defining the data to import and the identifying batch number to use.

For information on editing the EIM configuration file for data import, read [“Editing the Configuration File for Import Processing” on page 66](#).

8 Test your import process.

Run a small test batch (perhaps 100 records) to verify that the interface tables load correctly, and that the correct parameters are set in the configuration file and on the `srvmgr` command line.

For information on testing your import process, read [“Importing Initial Data” on page 73](#).

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. If a batch does not import correctly, using smaller batches simplifies the effort of isolating the condition, correcting the problem, and rerunning the batch.

For information on running the import process, read [“Running an Import Process” on page 86](#).

10 Verify results.

EIM provides several diagnostic tools that let you verify the success of import processing. For a description of these tools and how to use them, read [“Checking Import Results” on page 86](#).

11 Repeat [Step 8](#) through [Step 10](#) for each batch you are importing.

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 this section, although the scope of the import is usually significantly smaller than that of an initial data import.

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 interface table included in the process. Depending on the type of import process, EIM may repeat several tasks. All tasks are performed for each column in the interface table.

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

- 1 Initializes any temporary columns:
 - Compares values in IF_ROW_BATCH_NUM with the batch number provided by the Component task that initiated this import process.
 - 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 fails at this step. Rows are not imported or updated.

- 2 Applies any DEFAULT_COLUMN and FIXED_COLUMN values defined for this import process.
- 3 Applies any filter queries defined for this import process. If a row fails the filter query, EIM eliminates the row from further processing.
- 4 Generates foreign key references for rows with corresponding existing rows in the Siebel base tables; writes these foreign key values into interface 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 the business component level. For more information on bounded and unbounded picklists, read *Siebel Tools Reference, MidMarket Edition*.

- 5 Writes the appropriate ROW_ID values in the interface table rows' temporary columns, for rows with corresponding base table rows.
- 6 Creates a ROW_ID with a unique value in the base table, for interface table rows without corresponding rows in the base tables.

- 7 Eliminates rows with invalid values for user keys from further processing.

NOTE: EIM does not support modification of existing user key columns.

Generates foreign key references for rows without corresponding rows in the Siebel database tables, and writes these foreign key values into interface table temporary columns:

- If foreign keys fail for required columns, EIM eliminates these rows from further processing.
 - For interface table rows with data that resides in multiple destination tables, EIM fails rows with foreign keys that cannot be generated.
- 8 Updates contents of existing base table rows with contents from corresponding interface 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 interface table rows have the same user primary key for a base table, EIM uses only the first interface table row to update the base table, and ignores the data in other rows.
 - 9 Inserts any new interface table rows that have successfully passed all earlier steps in the Siebel database tables:
 - If Docking Transaction Logging is enabled, then EIM writes new rows to the Master Transaction Log or in a DX file in the file system's eim directory (controlled by the LOG TRANSACTIONS TO FILE parameter).
 - If multiple interface table rows use the same user primary key for a base table, EIM uses only the first interface table row to update the base table, and ignores the data in other rows.
 - 10 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.
 - 11 Runs optional miscellaneous SQL statements. For more information, read about the MISC SQL parameter in [“Header and Process Parameters”](#) on page 68.

Preparing the Interface Tables for Import Processing

This section explains how to prepare the interface tables for subsequent import into a Siebel database.

To import data, EIM reads data in the interface tables and writes data in the appropriate Siebel base tables by making multiple passes through the interface tables to:

- Set initial values for some columns in the interface tables
- Apply filter logic to select rows for importing
- Generate foreign key references and internal values
- Add or update relevant Siebel database rows
- Update each interface table row to indicate its import status

For general information on interface tables, read [Chapter 2, “Siebel Interface Tables.”](#)

Recommended Import Order

Import data in the following order to make sure that relationships between dependent data elements are established correctly:

- 1** Reference Data (Regions, Territories, and so on)
 - List of Values
- 2** Organizations
- 3** Business Unit
- 4** Products
- 5** Positions
- 6** Persons (Contacts and Employees)
- 7** Households
- 8** Accounts
- 9** Opportunities

- 10** Service Requests
- 11** Activities and Appointments
- 12** Personal Accounts
- 13** Quotes
- 14** Documents
- 15** Fulfillment
- 16** Campaigns
- 17** Product Defects
- 18** Notes
- 19** File Attachments

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

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 line in the configuration file

For more information on special columns, read [“Interface Table Columns”](#) on page 19.

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 (DOC, XLS, or TXT).

FILE_SRC_TYPE. This column requires the value FILE or URL.

For more information on file attachment columns, read [“File Attachment Columns” on page 20](#).

Adjusting the Case of Values

Prior to importing data into base table columns, EIM also adjusts the case of values in interface 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 interface table to adjust the case of column values and update the row accordingly. If this occurs, you should expect slower import processing. To change the case mode, consult Siebel Expert Services because this requires changing read-only properties defined at the table level.

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 86](#).

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, read [Chapter 3, “EIM Configuration File.”](#)

Header Section

Parameters in the header section generally apply to all processes. For a description of the necessary contents in this section, read [“Header Section” on page 44.](#)

Process Section

Parameters in the process section apply only to that 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, read [“Process Sections” on page 45.](#)

[Table 7 on page 52](#) lists the parameters specific to an import process that appear in the process section of the EIM configuration file.

FILTER QUERY. (IMPORT only.) Names a query that runs before the import process. The query prescreens certain rows in the import batch, using data values in the interface tables. The query expression should be a self-contained WHERE clause expression (without the WHERE keyword) and should use only unqualified column names from the interface table or literal values (such as NAME IS NOT NULL). By default, the FILTER QUERY parameter is not used.

ONLY BASE TABLES. Specifies and restricts selected base tables for the import process. Use commas to separate table names. Target tables for interface tables must be included. The default is to process all base tables into rows that can be imported from the interface tables. Use this parameter to improve performance when updating only a few tables. This parameter affects all interface tables used in the import process. For example:

```
ONLY BASE TABLES = S_CONTACT, S_ORG_EXT
```

IGNORE BASE TABLES. Specifies base tables to be ignored by the import process. Use commas to separate table names. Target tables for interface 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 interface 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. For example:

```
ONLY BASE COLUMNS= S_ORG_EXT.NAME, S_ORG_EXT.LOC, S_ORG_EXT.BU_ID
```

ORACLE INSERT APPEND MODE. The default value is FALSE, which means EIM does not use append mode for insert at EIM process step 9, avoiding a deadlock when running parallel EIM processes. If you need to improve step 9 performance, you can set the value to TRUE.

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 interface columns.

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

Header and Process Parameters

This section 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, read [“Process Sections” on page 45](#).

[Table 7 on page 52](#) lists the parameters specific to an import process that appear in the process section of the EIM configuration file.

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 it. Example:

```
ATTACHMENT DIRECTORY = SIEBEL_HOME\INPUT
```

DEFAULT COLUMN. (IMPORT only) Specifies a default value for an interface column table. The syntax is COLUMN NAME, VALUE, as in the following example:

```
DEFAULT COLUMN = CURCY_CD , "USD"
```

EIM uses the given value only if the column is NULL in the interface table.

FIXED COLUMN. (IMPORT only.) Specifies the value for a column from the interface table. The syntax is the same as for DEFAULT COLUMN. EIM loads the given value into the Siebel base table, overriding the value in the interface table column.

INSERT ROWS. Specifies that nonexistent rows in the interface table be inserted into the base table. The default is INSERT ROWS = TRUE. A table name can be specified with INSERT ROWS as the first value, separated by a comma, as in the following example:

```
INSERT ROWS = EIM_ACCOUNT, FALSE
```

If the named table is an interface table, as in the example, the setting applies to all base tables imported from this interface table. If the table is a base table, the setting is applied when data is imported from any interface table.

The INSERT ROWS parameter has an optional element of its syntax. 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:

```
INSERT 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:

```
INSERT ROWS = FALSE  
INSERT ROWS = S_CONTACT, TRUE  
INSERT ROWS = S_ADD_ORG, TRUE
```

NET CHANGE. (IMPORT only.) Specifies the handling of NULL (non-key) column values when importing a row that already exists in the Siebel database table. If NET CHANGE = TRUE, EIM ignores the null value; otherwise, EIM updates the column in the base table with NULL. The default is NET CHANGE = TRUE; EIM ignores NULL attribute values for existing rows by default. This parameter is ignored if UPDATES ROWS = FALSE.

TRIM SPACES. (IMPORT only.) Specifies whether the character columns in the interface tables should have trailing spaces removed before importing. Possible values are TRUE or FALSE. The default value is TRUE.

COMMIT EACH PASS. Specifies whether a separate transaction should be used for each EIM pass through each interface table. The default is COMMIT EACH PASS = 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.

COMMIT EACH TABLE. Specifies whether a separate transaction should be used for each interface table. The default is COMMIT EACH TABLE = TRUE, which invokes commits after each table. This setting helps to reduce the database resources required for the import process.

NOTE: The above two commit parameters work cumulatively. If you set both COMMIT EACH PASS and COMMIT EACH TABLE to TRUE, a commit occurs 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.

MISC SQL. Sets specific explicit or implicit primaries, as mentioned in [Step 11 on page 62](#) of 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.

Table 8 lists the fields that can be set using the MISC SQL parameter. The table lists the value of 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). For this parameter use underscores (not periods) to separate values.

Table 8. Fields Used in MISC SQL Parameter

Table and Column	Explicit Value
S_CONTACT.PR_HELD_POSTN_ID	EXPR_S_CONTACT_PR_HELD_POSTN_ID
S_CONTACT.PR_OU_ADDR_ID	EXPR_S_CONTACT_PR_OU_ADDR_ID
S_CONTACT.PR_USERROLE_ID	EXPR_S_CONTACT_PR_USERROLE_ID
S_OPTY.PR_OU_ADDR_ID	EXPR_S_OPTY_PR_OU_ADDR_ID
S_OPTY.PR_OU_INDUSTRY_ID	EXPR_S_OPTY_PR_OU_INDUSTRY_ID
S_ORG_EXT.PR_BL_PER_ID	EXPR_S_ORG_EXT_PR_BL_PER_ID
S_ORG_EXT.PR_SHIP_PER_ID	EXPR_S_ORG_EXT_PR_SHIP_PER_ID
S_POSTN.PR_EMP_ID	EXPR_S_POSTN_PR_EMP_ID
S_POSTN.PR_POSTN_ADDR_ID	EXPR_S_POSTN_PR_POSTN_ADDR_ID
S_PROJ.PR_OU_ADDR_ID	EXPR_S_PROJ_PR_OU_ADDR_ID

MISC SQL can only be used with specific interface tables. For example, MISC SQL = EXPR_S_CONTACT_PR_OU_ADDR_ID can only be used with EIM_CONTACT, and MISC SQL = EXPR_S_CONTACT_PR_HELD_POSTN_ID can only be used with EIM_EMPLOYEE. The default.ifb file contains a detailed explanation and examples on how to use MISC SQL to set primary child foreign keys.

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 use explicit primaries on the records where you have specified a primary, and to automatically use implicit 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 this parameter, see the sample default.ifb file located in the Siebel Server/admin directory.

ROLLBACK ON ERROR. Specifies whether the current transaction should be rolled back (aborted) when an error, such as a SQL database failure, is encountered. The default is `ROLLBACK ON ERROR = 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.

Working with EIM for Import Processing

This section explains the procedures for performing the following import processes:

- Importing initial data
- Importing changes to existing data
- Importing private contacts
- Importing solutions
- Importing call lists
- Importing contact address usage
- Importing industry codes
- Importing file attachments

Importing Initial Data

When you first import data into Siebel database tables, you should adhere to the following guidelines:

- **Perform a database backup.** To ease recovery in the event of an unexpected problem, back up your existing database before you begin an import process.
- **Follow the recommended import order.** Siebel interface tables do not map one-to-one with Siebel target database tables. To make sure that the necessary data is present to establish relationships between data entities, use the recommended sequence outlined in [“Recommended Import Order” on page 63](#).

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

- **Use small batch sizes.** To provide better control over import status, consider running small batches (1,000 to 5,000 rows). This allows you to understand the status of an individual import batch. When you have completed a successful import with a small batch size, you can use larger batch sizes to minimize administration and optimize performance. For more information, read [“Optimizing Performance” on page 148](#).

To import initial batches of data

- 1 In the interface 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.

CAUTION: If you have active mobile Web clients, do *not* disable Docking Transaction Logging. Otherwise, the server database and mobile Web client databases do not synchronize after the import. To resynchronize them, you need to reextract the database for each mobile Web client and rerun database initialization on each client.

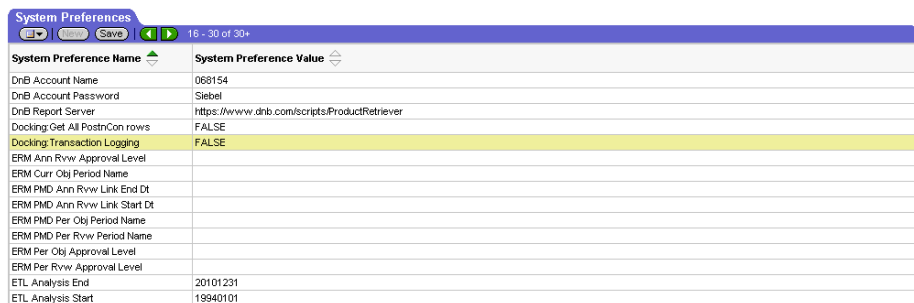
- a Navigate to the System Preferences screen.
- b Select Docking: Transaction Logging.
- c In the System Preference Value field, type FALSE.

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, read [“EIM Configuration File Parameters” on page 39](#).

The following illustration shows an example of disabling the Docking: Transaction Logging Preference using Server Manager. It displays the System Preferences list with the Docking Transaction Logging preference selected. The Value of the preference is FALSE.



System Preference Name	System Preference Value
DnB Account Name	068154
DnB Account Password	Siebel
DnB Report Server	https://www.dnb.com/scripts/ProductRetriever
Docking Get All PostmCon rows	FALSE
Docking Transaction Logging	FALSE
ERM Ann Rvw Approval Level	
ERM Cur Obj Period Name	
ERM PMD Ann Rvw Link End Dt	
ERM PMD Ann Rvw Link Start Dt	
ERM PMD Per Obj Period Name	
ERM PMD Per Rvw Period Name	
ERM Per Obj Approval Level	
ERM Per Rvw Approval Level	
ETL Analysis End	20101231
ETL Analysis Start	19940101

- 3 Start an EIM task for each batch number.

For information on running an EIM process, read [“Running an Import Process” on page 86](#).

- 4 Review your import processes by using the EIM trace file (eim_task.log).

This file contains comprehensive status and diagnostic information about the import processes. By default, this trace file is located in the admin directory of the Siebel Server directory.

Importing Changes to Existing Data

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. However, situations may arise in which you want to perform only inserts or updates.

NOTE: To update user key columns in S_ORG_EXT and S_PROD_INT tables use EIM_ORG_EXT_UK and EIM_PROD_INT_UK. EIM does not support modification of existing user key columns.

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 of unmatched rows

- 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.

NOTE: Because suppressing updates prevents updating primaries in [Step 10 on page 62](#), 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
```

Checking for Duplicate Reporting Relationships

If you are importing positions, 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 get an error when you click Generate Reporting Relationships after the import.

Importing Marketing Responses

For the rows to be displayed in the Response views, populate the CAMP_MEDIA_ID column in the S_COMMUNICATION base table with valid values from the S_SRC_DCP base table.

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. Furthermore, to import contacts, you must have this section in the EIM configuration file.

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, EIM reports a low-level error. Additionally, the Internal Publish flag (INTR_PUBLISH_FLG) has to 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, EIM reports a low-level error.

Importing Addresses

When importing into the S_ADDR_ORG base table, make sure the address name, plus the account or person ID form a unique combination. When the source of the address data provides a unique name for the address (within the Account or Contact), this name can be mapped to the new address name field. For example, if a contact has two addresses, already named “Primary Residence” and “Vacation Home” in the existing data source, those names can be retained when importing by setting the NAME_LOCK_FLG in the EIM_ADDR_ORG table to Y.

To import account addresses where the Account ID and the address name should be unique, use S_ADDR_ORG. To import contact addresses where the Contact ID and the address name should be unique, use S_ADDR_PER. You can import into S_ADDR_ORG using EIM_ACCOUNT or EIM_ADDR_ORG. You can import into S_ADDR_PER using EIM_CONTACT or EIM_ADDR_PER.

Both S_ADDR_ORG and S_ADDR_PER have a NAME_LOCK_FLG column. This column maps to the ADDR_NAME_LOCK_FLG column in EIM_ACCOUNT, EIM_ADDR_ORG, and EIM_CONTACT, and it maps to the AP_NAME_LOCK_FLG column in EIM_ADDR_PER.

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 Mobile Client Data

To load mobile client data into S_NODE table, follow these rules to make sure the remote environment works properly.

- S_NODE.PAR_NODE_ID is a required column.
- S_NODE.NODE_TYPE_CD should be set to REMOTE. When users register a mobile client through Siebel Remote Administration > Mobile Client, this column is populated with a default value REMOTE.
- The parent node should contain one of these two values:
 - HQ (if mobile clients are registered against HQ). HQ is from the Siebel seed data which should be loaded during Siebel standard installation process.
 - The name of the Regional Node (if mobile clients are attached to a Regional Node). Register a Regional Node through Siebel Remote Administration > Mobile Client, and then use that value as the parent node when loading Nodes through EIM.

Importing Opportunities

Because EIM_OPTY_DTL has S_OPTY_PROD mapped as User key only, do not use EIM_OPTY_DTL to import into S_OPTY_PROD. Instead, use EIM_OPTY1 for importing into S_OPTY_PROD.

Do not use an EIM table for importing records into a base table that has “User key only” mapping.

Opportunity (S_OPTY) Columns Manually Denormalized from Revenue (S_REVN)

The SUM_* columns on S_OPTY are manually denormalized from the Summary Revenue Item (S_REVN). You must make sure that these columns are kept in synchronization with the corresponding column on S_REVN (for the summary revenue item row). EIM does not automatically handle the manual denormalization of columns. To use EIM_OPTY for importing data into S_REVN, read the Forecasting chapter in *Applications Administration Guide*, as well as the table and column comments for those SUM_* columns in S_OPTY in *Siebel Data Warehouse Data Model Reference*. Make sure the data for the interface table columns OPTY_SUM_* are the same as the data for the corresponding columns in S_REVN that are referenced by S_OPTY.SUM_REVN_ITEM_ID. Also refer to the comments at the attribute mapping level of these interface tables columns in Siebel Tools.

To see the comments of the attribute mapping

- 1** In Siebel Tools, navigate to the Types tab.
- 2** Click EIM Interface Table Object type.
- 3** Query for EIM_OPTY.
- 4** Click EIM Table Mapping Object.
- 5** Select the Table Mapping = S_OPTY.
- 6** Click on Attribute Mapping.
- 7** Scroll to SUM_* columns and look at the comments for these SUM_* attribute mappings.

Child LOVs are Bounded Consistent with the Application

Previously, if a column is based on a hierarchical LOV type, you would populate the LOV type and also BOUND it for the parent level column. You usually would not populate the LOV type for the child level column and would not BOUND it, even though the application binds both the parent and the child LOVs. Identifying the parent column required understanding the application functionality and usage of columns. In this release, you populate the LOV type in both parent and child level columns and BOUND both the parent and child level LOVs, consistent with the application. Hence, the child LOV is validated against all values in the specific column. The EIM engine checks whether the imported data exists in the S_LST_OF_VAL table, even though it may not be able to check that the parent and child are inconsistent.

When loading new records using EIM_LST_OF_VAL in a MLOV (multi-lingual list of values) situation, make sure to populate the NAME column with the display value (found in Application Administration > List of Values and in the S_LST_OF_VAL table). Do not populate the NAME column with the language independent code.

NOTE: EIM does not support MLOV for Hierarchical LOVs.

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).

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 interface tables support all file attachment formats.

- 4 Populate interface tables with rows matching the file attachments.
- 5 Run EIM.

NOTE: All three file attachment columns (FILE_NAME, FILE_EXT, and FILE_SRC_TYPE) must be populated to import file attachments. The FILE_SRC_TYPE column must be set to FILE or URL. Although these columns can be listed as nullable in the interface tables, the import process returns 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.

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, clean the file attachment directory of any unused file attachments using the Siebel File System Maintenance Utility, sfscleanup.exe, during hours when the network is least active.

To update file attachments

- 1 Update the file attachment by completing the steps in [“Importing File Attachments” on page 81](#).
- 2 When 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, read [Appendix B, “Siebel File System Cleanup Utility.”](#)

NOTE: EIM does not support merging of file attachments.

Importing into Base Tables that Contain the BU_ID Column

Base tables in the Siebel Data Model that are enabled for multi-org 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 an overview of multi-org, read *Siebel Data Model Reference*.

During the import process, if the value supplied in the interface table does not resolve to a valid business organization, EIM by default continues to import the record with the BU_ID set to the default value as 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 configuration file (the default value for this parameter is FALSE).

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

If you are using multiple organizations, be aware that S_ORG_EXT.BU_ID does not identify the organization, but rather it indicates the organization's visibility. S_ORG_EXT.PAR_BU_ID identifies the organization and should refer to the same organization as S_POSTN.BU_ID.

Importing Large Databases

Before importing a large database, you should thoroughly test your import processes. When 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 and resources. 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. EIM uses database server space for the interface 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:

- **Siebel 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, read the documentation for your RDBMS or consult your database administrator.
- **Secondary tables.** You may be importing data from a single interface table into multiple destination tables. For each interface 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 interface 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.

Running an Import Process

You can run an import process when you have:

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

Run the import process by completing the procedures in [Chapter 8, “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.

To view a list of imported rows

- Query the appropriate interface tables for rows whose `if_row_batch_num` equals the batch number for the import.

These columns in each interface 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 following:

IN_PROGRESS. In Step 1, Integration Manager sets `IF_ROW_STAT` to this initial value for all rows in the batch. If rows still have this status value after Integration Manager exits, a failure occurred that aborted processing for this table.

SQL_ERROR. A SQL error occurred during an attempt to import this row. This error occurs for rows processed when transaction logging is `TRUE`.

IMPORT_REJECTED. A user-specified filter query failed for this row. This error occurs in Step 3 if the user has specified `FILTER QUERY` expressions.

FOREIGN_KEY. A required foreign key column in the target table could not be resolved. This error occurs in Step 4.

PICKLIST_VALUES. 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 61](#).

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 62](#), or when inserting new rows in [Step 9 on page 62](#).

DUP_RECORD_EXISTS. The row exactly matches rows that already exist in the destination tables. This error occurs in Step 8. Note that a row may have a duplicate in the target base table, but not in other destination base tables. In this situation, Integration Manager adds the new relation (a child or intersection table) in the other destination base tables, and does not mark the interface table row as a duplicate.

The `IF_ROW_STAT` column can also be set to the value `DUP_RECORD_EXISTS` when the same record (same user key) exists with the same EIM batch number with a lower `ROW_ID`. In this case, `MIN(ROW_ID)` is the record processed, and the other records with the same user key are marked as `DUP_RECORD_EXISTS`.

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.

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 interface 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.

NOT_ALLOWED. EIM 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 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. To get a detailed log file, set Component Event Tracing for the EIM component. For more information on viewing the trace file, read [“Viewing the Task Info Log” on page 140](#).

This chapter explains how to export data from the Siebel base tables into the interface tables. Upon completion of the EIM process, your DBA can access the interface tables and extract the data for use in a non-Siebel application.

Export Process

To export data, EIM reads the data in the Siebel database tables and places the information in the appropriate interface tables. You can then copy data from the interface tables into another database.

During its multiple passes through the interface tables, EIM does the following actions.

- Initializes the interface tables for export.
- Applies filter logic to select rows for exporting.
- Updates interface 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.

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

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

1 Initialize interface 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 Use 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, locate child table rows and export them to their corresponding interface tables.

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

Preparing the Interface Tables for Export Processing

Unlike other Open Interfaces processes, an export process requires minimal preparation of the interface tables. During the first step of export processing, EIM inspects each interface table involved in the process. If EIM finds a row whose IF_ROW_BATCH_NUM matches the batch number for this export process, it will do 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, read [“Header and Process Parameters” on page 94](#).

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.

The values for the LAST_UPD and CREATED columns in the interface tables will 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 will contain the data from the S_CONTACT.LAST_UPD and S_CONTACT.CREATED columns, respectively.

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_PCMP_DTL
- EIM_IC_PERF_HST
- EIM_MDF

For more information on special columns, read [“Interface Table Columns” on page 19](#), and for general information on interface tables, read [Chapter 2, “Siebel Interface Tables.”](#)

Editing the Configuration File for Export Processing

This section describes the header and process sections that you will need in the EIM configuration file to properly configure EIM for an export process. For general information on the EIM configuration file, read [Chapter 3, “EIM Configuration File.”](#)

Header Section

Parameters in the header section generally apply to all processes. For a description of the necessary contents in this section, read [“Header Section” on page 44.](#)

Process Section

Parameters in the process section apply only to that process and override any corresponding value in the header section for the specific process. 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 accounts table.

```
[Export Accounts]
TYPE = EXPORT
BATCH = 2
TABLE = EIM_ACTIVITY
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.

For generic parameters that can be used in all EIM processes, read [“Process Sections” on page 45.](#)

Header and Process Parameters

This section describes the parameters that can appear in 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, read [“Process Sections” on page 45](#).

[Table 7 on page 52](#) lists the parameters specific to an export process that appear in the process section of the EIM configuration file.

ATTACHMENT DIRECTORY. (Default = 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 interface table for the given batch number should be deleted. Valid values are TRUE (the default) and FALSE.

EXPORT MATCHES. Specifies a base table and a WHERE clause expression for filtering base table rows. The WHERE clause expression (without the WHERE) should use only literal values or unqualified column names from the base table. Separate the operator from the operand using a space.

There are several variations of this parameter. Use the variation that matches your situation. In these examples, notice the space after the > operator.

To export data based on criteria from the S_PARTY table, use this form:

```
EXPORT MATCHES = S_PARTY, (CREATED > '2002-05-01'
```

EIM exports the data into the EIM_PARTY table. You can use ONLY BASE TABLES to control exporting the target base table's children (such as S_PARTY_PER for this example).

To export data based on criteria on an extension table of S_PARTY table, use this form:

```
EXPORT MATCHES = S_ORG_EXT, (CREATED > '2002-05-01')
```

EIM exports the data into the EIM_ACCOUNT table.

For tables that are not S_PARTY or its extension tables, use one of these forms:

```
EXPORT MATCHES = (CREATED > '2002-05-01')
```

OR

```
EXPORT MATCHES = target base table name, (CREATED > '2002-05-01')
```

EXPORT ALL ROWS. Specifies that all rows in the target base table and secondary tables are to be exported. Valid values are TRUE and FALSE (the default). Existing values in the interface table and EXPORT MATCHES expressions are ignored.

Working with EIM for Export Processing

This section explains how to perform the following types of export processes:

- Exporting all data rows
- Exporting selected data rows

Exporting All Data Rows

To export all rows from the tables that are mapped in an interface 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_ACTIVITY
EXPORT ALL ROWS = TRUE
```

Prior to exporting, make sure that your DBA has allocated enough space for the interface 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.

Running an Export Process

You may run an export process when you have:

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

Run the export process by completing the procedures in [Chapter 8, “Running EIM.”](#)

Exporting Names from S_BU

If you are exporting data that pertains to organizations and divisions, it may be necessary to run additional SQL statements against the interface 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 interface 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.

To view a list of exported rows

- Query the appropriate interface 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, read [“Viewing the Task Info Log” on page 140](#).

Extracting Data from the Interface Tables

Upon completion of an export process, the database administrator can use appropriate tools to extract data from the interface tables for subsequent use by an external application.

This chapter covers the process of deleting selected data from the Siebel database. It covers preparing interface tables, editing the configuration file, running the EIM process, and checking the results.

Delete Process Overview

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

During its multiple passes through the interface tables, EIM does the following:

- Initialize the interface tables for deletion
- Apply filter logic to select rows for deleting, insert interface tables rows that correspond to matching base table rows, or select rows with matching user keys in the interface tables
- Update 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.

EIM uses a combination of interface 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 interface 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 interface 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

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

The DELETE EXACT parameter specifies the base table rows to delete by using user key values specified in the interface table. By default, DELETE EXACT is set to 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. 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.

To delete data, EIM performs the following steps:

1 Initialize interface 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 interface table values to identify rows for deletion.

2 Delete 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 interface table.

Only use this parameter to delete rows from nontarget base tables that contain user keys. EIM does not delete rows in nontarget base tables that do not contain user keys.

- 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.

Only use this parameter to delete rows from target base tables. EIM deletes rows from the target base table even if the DELETE ROWS parameter is set to FALSE for that table.

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

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

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

For information on configuration file parameters to use in a delete process, read [“Header and Process Parameters” on page 104](#).

- 3** Set IF_ROW_STAT to deleted for rows that are successfully processed.
- 4** 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: If you have mobile Web clients, transaction logging should be in effect during delete operations. This captures the appropriate transactions for later docking.

Preparing the Interface Tables for Delete Processing

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

You must make sure that each interface 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 interface 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

For more information on special columns, read [“Interface Table Columns” on page 19](#), and for general information on interface tables, read [Chapter 2, “Siebel Interface 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. For general information on the EIM configuration file, read [Chapter 3, “EIM Configuration File.”](#)

Header Section

Parameters in the header section generally apply to all processes. For a description of the necessary contents in this section, read [“Header Section” on page 44.](#)

Process Section

Parameters in the process section apply only to that process and override any corresponding value in the header section for the specific process. 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 is 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
```

Header and Process Parameters

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, read [“Process Sections” on page 45](#).

[Table 7 on page 52](#) lists the parameters specific to a delete process that appear in the process section of the EIM configuration file.

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

CASCADE DELETE ONLY. (Default = FALSE). Set 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. Specifies whether existing rows in the interface 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 = TRUE).

DELETE EXACT. Specifies the base table rows to delete by using user key values specified in the interface table. For example, you can use DELETE EXACT to delete any of the nontarget base tables such as S_ADDR_ORG and S_ACCNT_POSTN using the EIM_ACCOUNT table. Load the EIM_ACCOUNT table with records that specifically identify the S_ACCNT_POSTN or the S_ADDR_ORG records to be deleted. 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.

DELETE MATCHES. Specifies a table and a WHERE clause expression for filtering table rows. The WHERE clause expression (without the WHERE) should use only literal values or unqualified column names from the base table. Separate the operator from the operand using a space.

There are two variations of this parameter; one used for S_PARTY and its extension tables, and one used for all other base tables.

For S_PARTY and its extension tables, use this form:

```
DELETE MATCHES = S_PARTY, (CREATED > '2001-10-01')
```

For all other base tables, use this form:

```
TABLE = EIM_ACCOUNT  
DELETE MATCHES = S_ORG_EXT, (CREATED > '2001-10-01')
```

EIM finds the rows in S_ORG_EXT that match Created > '2001-10-01' and their corresponding parent rows in S_PARTY. The EIM engine deletes from S_PARTY and does a cascade delete or cascade update. EIM Delete handles all children tables of the target table, not only those tables mapped by the interface table.

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

NOTE: Use this parameter only to delete rows from target base tables. Rows are deleted from the target base table even if the DELETE ROWS parameter is set to FALSE for that table.

DELETE ALL ROWS. 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 interface table and DELETE MATCHES expressions are ignored.

This parameter will only write the user keys values of the deleted target table rows to the interface table columns. It will not write values of non-user keys columns or nontarget table rows column values to the interface table. The deleted rows cannot be reimported using the interface 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. To selectively delete rows, use the DELETE EXACT or DELETE MATCHES expressions.

DELETE ROWS. 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 prevents deletion of rows from the S_ADDR_ORG table:

```
DELETE ROWS = S_ADDR_ORG, FALSE
```

UPDATE ROWS. Specifies if foreign key references can be updated. The default is UPDATE ROWS = TRUE, which affects all tables. To affect only specific tables, you can specify a table name. For example:

```
UPDATE ROWS = S_CONTACT, TRUE
```

This parameter 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 settings for DELETE ROWS and UPDATE ROWS carefully. Inappropriate use can result in dangling foreign key pointers.

EIM SCHEMA CACHE. This caches the column relations. The default value is TRUE, which results in faster processing. If the value is set to FALSE then the cache is not used.

Working with EIM for Delete Processing

This section explains configuration file settings that direct EIM to perform delete processing:

- Deleting all data rows for a target base table
- Deleting data rows identified by user key values
- Deleting file attachments
- Handling aborts

Deleting All Data Rows

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_ACTIVITY
  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

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

- 1 Set the DELETED 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 interface tables from the accounts table:

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

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

- Territory Items
- Fulfillment Items

Deleting File Attachments

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

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 the file attachments you want to delete.
- 2 When file attachments have been deleted, run the Siebel File System Maintenance Utility, `sfscleanup.exe`, to clean up the file attachment directory.

For information on using `sfscleanup.exe`, read [Appendix B, “Siebel File System Cleanup Utility.”](#)

Deleting from S_NOTE and S_*_SKILL_IT Tables

Previously you could not delete from the S_NOTE* and S_*_SKILL_IT tables because they did not have primary user key. Now you can delete records from S_NOTE* and S_*_SKILL_IT tables, without deleting records from the parent tables, by using EIM_NOTE_DEL and EIM_SKLI_DEL respectively.

EIM_NOTE_DEL should only be used for EIM DELETE from S_NOTE_* tables (such as S_NOTE_ACT, S_NOTE_ASSET, and so on). Similarly, EIM_SKLI_DEL should only be used for deleting skill items from S_*_*SKILL* tables (such as S_ACT_SKILL_IT). To use these tables, populate ROW_IDs of records to be deleted into the corresponding columns in the interface table and then use DELETE EXACT to delete the records.

For example, to delete records from S_NOTE_ACT, populate EIM_NOTE_DEL.ACT_ROW_ID with the ROW_IDs of the records in S_NOTE_ACT to be deleted. Then run EIM DELETE EXACT process with the following parameter specified in the IFB file:

```
DELETE EXACT = TRUE", "ONLY BASE TABLES = S_NOTE_ACT
```

Here is a sample from an IFB file:

```
[DELETE EIM_NOTE_DEL]
TYPE = DELETE
BATCH = 1
TABLE = EIM_NOTE_DEL
DELETE EXACT = TRUE
ONLY BASE TABLES = S_NOTE_ACT
```

For more details, see the table comments in *Siebel Interface Tables Reference*.

Handling Aborts of EIM Delete Processing

If an EIM delete process is aborted, base tables associated with deleted 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, you should set the following parameters in the IFB configuration file to make sure that the EIM delete process only performs one commit and rollback when aborted:

```
COMMIT EACH TABLE = FALSE
```

```
COMMIT EACH PASS = FALSE
```

```
ROLLBACK ON ERROR = TRUE
```

Cascade Behavior of EIM Deletion

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.

Creating Temporary Indexes

Delete and Merge performance is improved if you create some specific temporary indexes first.

To create temporary indexes

- 1 Make a list of the base tables (S_*) that are to be affected by the delete or merge.
 - a If you specify “ONLY BASE TABLES = ” in your IFB file, then these are the only tables affected.
 - b If you do not specify “ONLY BASE TABLES = ” in your IFB file, then review mappings to determine this list of tables.
 - c Do not include S_USER, as this table has many foreign keys and does not aid the search.
- 2 Using the list of base tables, use the following query to capture the list of table and column pairs on which indexes should be created.

```
select T.NAME,C.NAME

from S_COLUMN C, S_TABLE T, S_TABLE KT, S_REPOSITORY R

where T.ROW_ID=C.TBL_ID AND C.FKEY_TBL_ID=KT.ROW_ID AND
KT.NAME=upper('{list of tables}')

AND KT.REPOSITORY_ID=R.ROW_ID and R.NAME='Siebel Repository'

and C.PR_KEY != 'Y' and C.INACTIVE_FLG != 'Y' and T.INACTIVE_FLG
!= 'Y'

AND not exists (select '1' from S_INDEX_COLUMN IC,S_INDEX I where
I.ROW_ID = IC.INDEX_ID and IC.COL_ID=C.ROW_ID and IC.COL_POSTN=1
and IC.INACTIVE_FLG != 'Y' and I.INACTIVE_FLG != 'Y');
```

- 3 Review the list. Do not create indexes on tables with fewer than 1000 rows.
- 4 If you are using DB2 or MS SQL Server, update statistics for the tables on which you create indexes.
- 5 Run your delete or merge process.
- 6 When the delete or merge process completes, drop the indexes you created. If you are using DB2 or MS SQL Server, update statistics on those tables.

Table 9 shows the primary child columns for which there is EIM support. Some columns already have an index. For the others, you can create a temporary index if needed.

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_ACCNT_ISSUE	PR_SRV_REQ_ID	EIM_ACCNT_ISS	
S_ACCNT_SRC	PR_ACCNT_SRC_PI_ID	EIM_ACCNT_SRC	
	PR_ACCNT_SRCPLN_ID	EIM_ACCNT_SRC1	
S_ASSESS_VAL	PR_CERTATTR_PER_ID	EIM_ASSESS	
S_ASSET	PR_ACCNT_ID	EIM_ASSET	Yes
	PR_ASSET_WRNTY_ID	EIM_ASSET1	
	PR_CON_ID	EIM_ASSET	
	PR_EMP_ID	EIM_ASSET1	
S_AUC_ALRT_DEF	PR_CTLG_CAT_ID	EIM_AUC_ALRT	
S_AUC_ITEM	PR_CTLG_CAT_ID	EIM_AUC_ITEM	
S_BRDCST_MSG	PR_RECIP_ORG_ID	EIM_BRDCST_MSG	
S_CALL_LST	PR_POSTN_ID	EIM_CALL_LST	
	PR_SRC_ID	EIM_CALL_LST	
S_CMPT_MTRC	PR_PROD_LN_ID	EIM_CMPT_MTRC	
S_CON_PRDINT	PR_CON_ID	EIM_CON_PRDINT	Yes

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_CONTACT	PR_ACT_ID	EIM_CONTACT2	
	PR_ALT_PH_NUM_ID	EIM_CONTACT3	
	PR_BL_PER_ADDR_ID	EIM_CONTACT and EIM_EMPLOYEE1	
	PR_CON_ADDR_ID	EIM_CONTACT1	
	PR_EMAIL_ADDR_ID	EIM_CONTACT3	
	PR_GRP_OU_ID	EIM_CONTACT	
	PR_HELD_POSTN_ID	EIM_EMPLOYEE	
	PR_MKT_SEG_ID	EIM_CONTACT2	
	PR_NOTE_ID	EIM_CON_DTL	
	PR_OPTY_ID	EIM_CONTACT1	
	PR_OU_ADDR_ID	EIM_CONTACT	Yes
	PR_PER_ADDR_ID	EIM_CONTACT and EIM_EMPLOYEE1	
	PR_PER_PAY_PRFL_ID	EIM_CONTACT1	
	PR_POSTN_ID	EIM_CONTACT and EIM_CONTACT1	Yes
	PR_RESP_ID	EIM_EMPLOYEE	
	PR_SH_PER_ADDR_ID	EIM_CONTACT and EIM_EMPLOYEE1	
	PR_SYNC_USER_ID	EIM_CONTACT	
	PR_TERR_ID	EIM_CONTACT2	
PR_USERROLE_ID	EIM_USER		
S_CP_GDLN_SET	PR_GDLN_TBL_ID	EIM_COMP_GDLN	
S_CP_GDLN_TBL	PR_GDLN_ID	EIM_COMP_GDLN	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_CRSE_TST	CRSE_ID	EIM_CRSE_TST	
	PR_CRSE_OFFR_ID	EIM_CRSE_TST	
	PR_LANG_ID	EIM_CRSE_TST	
S_CTLG_CAT	PR_AUC_ITEM_ID	EIM_CTLG_CAT1	
	PR_COMPETITOR_ID	EIM_CTLG_CAT1	
	PR_CRSE_ID	EIM_CTLG_CAT1	
	PR_DECISION_ISS_ID	EIM_CTLG_CAT	
	PR_EVENT_ID	EIM_CTLG_CAT1	
	PR_LIT_ID	EIM_CTLG_CAT1	
	PR_PROD_ID	EIM_CTLG_CAT	
	PR_RES_ITM_ID	EIM_CTLG_CAT1	
	PR_SOLN_ID	EIM_CTLG_CAT1	
S_CURRCLM	PR_DESC_LIT_ID	EIM_CURRCLM	
	PR_TRGT_AUD_ID	EIM_CURRCLM	
S_DMND_CRTN_PRG	PR_CS_PATH_ID	EIM_DMNDCRTPRG	
	PR_DEAL_ID	EIM_DMNDCRTPRG	Yes
	PR_LIT_ID	EIM_DMNDCRTPRG	
	PR_OFFER_MTRL_ID	EIM_DMNDCRTPRG	
S_DOC_AGREE	SALES_REP_POSTN_ID	EIM_AGREEMENT and EIM_AGREEMENT1	
S_DOC_QUOTE	PR_QUOTE_SOLN_ID	EIM_QUOTE1	
S_EMP_PER	PR_WRK_EXP_ID	EIM_EMPLOYEE2	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_EVT_ACT	PR_ATT_ID	EIM_ACT_DTL	
	PR_CON_ID	EIM_ACTIVITY1	
	PR_EXP_RPT_ID	EIM_ACTIVITY2	
	PR_ORDER_ID	EIM_ACTIVITY1	
	PR_PRDINT_ID	EIM_ACTIVITY2	Yes
	PR_SYMPTOM_CD	EIM_ACTIVITY2	
	PR_TMSHT_LINE_ID	EIM_ACTIVITY1	
	TARGET_PER_ID	EIM_ACTIVITY1	
S_EVT_FUL_REQ	PR_DOC_ID	EIM_FUL	
	PR_LABEL_ID	EIM_FUL	
S_EXP_ITEM	PR_ADJ_ITEM_ID	EIM_ACTIVITY2, EIM_EXP_ITEM, and EIM_EXP_RPT	
	PR_INCL_CON_ID	EIM_EXP_ITEM	
	PR_INCL_EMP_ID	EIM_EXP_ITEM	
S_EXP_RPT	PR_APRV_BY	EIM_EXP_RPT	
S_IC_FLTR	PR_ACCNT_ID	EIM_IC_FILTER	
	PR_PROD_ID	EIM_IC_FILTER	
	PR_PROD_LN_ID	EIM_IC_FILTER	
S_IC_TXN	PR_POSTN_ID	EIM_IC_TXN	
S_INV_TXN	PR_ASSET_ID	EIM_INV_TXN	
S_INVOICE	PR_PAYMENT_ID	EIM_INVOICE	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_LIT	PR_ACCNT_ID	EIM_LITERATURE	Yes
	PR_CMPT_OU_ID	EIM_LITERATURE	Yes
	PR_INDUST_ID	EIM_LITERATURE	Yes
	PR_ISS_THEME_ID	EIM_LITERATURE	Yes
	PR_LIT_FORM_ID	EIM_LITERATURE	
	PR_PROD_INT_ID	EIM_LITERATURE	Yes
	PR_STG_ID	EIM_LITERATURE	
S_MKT_SEG	PR_PRI_LST_ID	EIM_MKT_SEG	
S_NODE	EMP_ID	EIM_NODE	
S_ONL_OBJECT	PR_CATEGORY_ID	EIM_ONL_OBJECT	
	PR_PARTY_ID	EIM_ONL_OBJECT	
S_ONL_PROJECT	PR_PARTY_ID	EIM_ONL_PROJECT	
S_OPTY	PR_CMPT_OU_ID	EIM_OPTY1	Yes
	PR_CON_ID	EIM_OPTY	
	PR_MKT_SEG_ID	EIM_OPTY1	
	PR_OPTY_INDUST_ID	EIM_OPTY1	Yes
	PR_POSTN_ID	EIM_OPTY	
	PR_PROD_ID	EIM_OPTY1	
	PR_PROJ_ID	EIM_OPTY1	
	PR_PRTNR_ID	EIM_OPTY1	Yes
	PR_QUOTE_ID	EIM_QUOTE	
	PR_SRC_ID	EIM_OPTY1	
	PR_TERR_ID	EIM_OPTY1	
	SUM_REVN_ITEM_ID	EIM_OPTY and EIM_REVN	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_ORDER	PR_PAYMENT_ID	EIM_ORDER1	
	PR_POSTN_ID	EIM_ORDER and EIM_ORDER1	
	PR_SHIPMENT_ID	EIM_ORDER1	
S_ORDPART_MVMT	PR_ASSET_ID	EIM_PART_MVMT	
S_ORG_BU	PR_ADDR_ID	EIM_ACCOUNT	
	PR_BL_ADDR_ID	EIM_ACCOUNT	
	PR_BL_OU_ID	EIM_ORG_BU	
	PR_PAY_OU_ID	EIM_ORG_BU	
	PR_SHIP_ADDR_ID	EIM_ACCOUNT	
	PR_SHIP_OU_ID	EIM_ORG_BU	
S_ORG_EXT	PR_ADDR_ID	EIM_ACCOUNT	Yes
	PR_ADDR_PER_ID	EIM_ACCOUNT2	
	PR_AGREE_ID	EIM_ACCOUNT2	
	PR_BL_ADDR_ID	EIM_ACCOUNT	
	PR_BL_OU_ID	EIM_ACCOUNT	
	PR_BL_PER_ID	EIM_ACCOUNT, EIM_CONTACT, EIM_EMPLOYEE, and EIM_USER	
	PR_CO_MSTR_ID	EIM_ACCOUNT1	
	PR_COMPETITOR_ID	EIM_ACCOUNT	
	PR_CRDT_AREA_ID	EIM_ACCOUNT3	
	PR_DISCNT_ID	EIM_ACCOUNT1	
PR_INDUST_ID	EIM_ACCOUNT1		

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
	PR_LOGO_ID	EIM_ACCNT_DTL	
	PR_MKT_SEG_ID	EIM_ACCOUNT1	
	PR_ORG_TRGT_MKT_ID	EIM_ACCOUNT3	
	PR_OU_TYPE_ID	EIM_ACCOUNT1	
	PR_PAY_OU_ID	EIM_ACCOUNT	
	PR_POSTN_ID	EIM_ACCOUNT	
	PR_PRI_LST_ID	EIM_ACCOUNT1	
	PR_PRTNR_OU_ID	EIM_ACCOUNT	
	PR_PRTNR_TIER_ID	EIM_ACCOUNT3	
	PR_PRTNR_TYPE_ID	EIM_ACCOUNT3	
	PR_PTSHM_MKTSEG_ID	EIM_ACCOUNT3	
	PR_SHIP_ADDR_ID	EIM_ACCOUNT	
	PR_SHIP_OU_ID	EIM_ACCOUNT	
	PR_SHIP_PER_ID	EIM_ACCOUNT, EIM_CONTACT, EIM_EMPLOYEE, and EIM_USER	
	PR_SRV_AGREE_ID	EIM_ACCOUNT2	
	PR_SVC_POSTN_ID	EIM_ACCOUNT	
	PR_SYN_ID	EIM_ACCOUNT1	
	PR_TERR_ID	EIM_ACCOUNT2	
S_ORG_FUL	PR_PER_ID	EIM_ORG_FUL	
S_ORG_GROUP	PR_CON_ID	EIM_GROUP	
	PR_POSTN_ID	EIM_GROUP	
	PR_TERR_ID	EIM_GROUP	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_ORG_PRTNR	PR_PRTNR_LOGO_ID	EIM_ACCOUNT3	
S_PARTY_GROUP	PR_COLB_RSRC_ID	EIM_PARTY_GRP	
S_PER_PRTNRAPPL	PR_EXPRTSE_ID	EIM_PRTNRAPPL	
	PR_INDUST_ID	EIM_PRTNRAPPL	
	PR_MKT_SEG_ID	EIM_PRTNRAPPL	
	PR_REGN_ID	EIM_PRTNRAPPL	
S_POSTN	PR_EMP_ID	EIM_POSITION	
	PR_OU_EXT_ID	EIM_POSITION	
	PR_POSTN_ADDR_ID	EIM_POSITION	
	PR_PROD_ID	EIM_POSITION	
	PR_PROD_LN_ID	EIM_POSITION	
	PR_TERR_ID	EIM_POSITION	
S_PREPAY_BAL	PR_AUTH_PARTY_ID	EIM_PREPAY_BAL	
S_PROD_DEFECT	PR_AFF_PRDINT_ID	EIM_DEFECT	Yes
	PR_FIXED_PROD_ID	EIM_DEFECT	
	PR_INT_EMP_ID	EIM_DEFECT	
	PR_SYMPTOM_ID	EIM_DEFECT	
	PR_TAG_ID	EIM_DEFECT	
	PR_TRGTD_PROD_ID	EIM_DEFECT	
	PROJ_ITEM_ID	EIM_PROJITEM	
S_PROD_EXT	PR_EQUIV_PROD_ID	EIM_PROD_EXT	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_PROD_INT	PR_EQUIV_PROD_ID	EIM_PROD_INT2	
	PR_POSTN_ID	EIM_PROD_INT1	
	PR_PROD_LN_ID	EIM_PROD_INT1 and EIM_PROD_INT2	
	PR_SEASON_ID	EIM_PROD_INT2	
	PR_TRGT_MKT_SEG_ID	EIM_PROD_INT2	
S_PROD_INT_CRSE	PR_CRSE_TST_ID	EIM_COURSE	
	PR_LANG_ID	EIM_COURSE	
S_PROD_LN	PR_POSTN_ID	EIM_PROD_LN	
	PR_PROD_ID	EIM_PROD_LN	
	PR_PROD_LN_ID	EIM_PROD_LN	
S_PROJ	PR_AGREE_ID	EIM_PROJECT1	
	PR_POSTN_ID	EIM_PROJECT	
	PR_PROJ_RSRC_ID	EIM_PROJECT1	
	PR_PRTNR_OU_ID	EIM_PROJECT	
	PR_SRC_ID	EIM_PROJECT1 and EIM_PROJECT	
	PR_SRC_OPTY_ID	EIM_PROJECT	
S_PROJ_ATT	PR_ACT_PER_CRT_ID	EIM_PROJECTDTL	
	PR_ACT_TYP_CRT_ID	EIM_PROJECTDTL	
	PR_ITEM_PER_CRT_ID	EIM_PROJECTDTL	
	PR_ITEM_TYP_CRT_ID	EIM_PROJECTDTL	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_PROJITEM	PR_ENG_CONTACT_ID	EIM_PROJITEM	
	PR_ENG_GROUP_ID	EIM_PROJITEM	
	PR_ENG_OWNER_ID	EIM_PROJITEM	
	PR_EXP_RPT_ID	EIM_PROJITEM	
	PR_INTL_CON_ID	EIM_PROJITEM	
	PR_PM_CONTACT_ID	EIM_PROJITEM	
	PR_PM_GROUP_ID	EIM_PROJITEM	
	PR_PM_OWNER_ID	EIM_PROJITEM	
	PR_PRDCSR_ITEM_ID	EIM_PROJITEM	
	PR_PROJ_RSRC_ID	EIM_PROJITEM	
	PR_QA_GROUP_ID	EIM_PROJITEM	
	PR_QA_MGR_ENG_ID	EIM_PROJITEM	
	PR_QA_MGR_INTL_ID	EIM_PROJITEM	
	PR_QA_OWNER_ID	EIM_PROJITEM	
	PR_REL_ENG_TASK_ID	EIM_PROJITEM	
	PR_REL_FTR_ID	EIM_PROJITEM	
	PR_REL_MRD_ID	EIM_PROJITEM	
	PR_REL_TECH_DOC_ID	EIM_PROJITEM	
	PR_REL_TST_PLAN_ID	EIM_PROJITEM	
	PR_REL_TSTSTRGY_ID	EIM_PROJITEM	
	PR_REQUESTNG_OU_ID	EIM_PROJITEM	
	PR_RESP_INT_OU_ID	EIM_PROJITEM	
	PR_TAG_ID	EIM_PROJITEM	
	PR_TECH_CON_ID	EIM_PROJITEM	
	PR_TMSHT_LINE_ID	EIM_PROJITEM	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_QTA_INCENTV	PR_EARNING_ID	EIM_IC_INCENTV	
S_SHIPMENT	PR_ORDER_ID	EIM_ORDER_SHIP	
S_SRC	PR_CALL_LST_ID	EIM_SRC3	
	PR_CS_PATH_ID	EIM_SRC1	
	PR_DMND_CRT_PRG_ID	EIM_SRC3	
	PR_MDF_ID	EIM_SRC3	
	PR_POSTN_ID	EIM_SRC and EIM_SRC3	Yes
	PR_PROD_INT_ID	EIM_SRC3	Yes
	PR_PROD_LN_ID	EIM_SRC1	
	PR_TERR_ID	EIM_SRC1	
	PR_TRGT_INDUST_ID	EIM_SRC1 and EIM_SRC2	
S_SRC_EVT	PR_CRSE_TST_ID	EIM_SRC_EVT1	
	PR_SPEAKER_ID	EIM_SRC_EVT1	
S_SRC_PAYMENT	PR_INVOICE_ID	EIM_PAYMENT	
S_SRV_REQ	PR_CON_ID	EIM_SRV_REQ1	Yes
	PR_SYMPTOM_CD	EIM_SRV_REQ1	
S_TASK_STEP	PR_USERLIST_ID	EIM_TASK_PROC	
S_TMPL_PLANITEM	PR_SALES_STG_ID	EIM_TMPL_PLNITM	
S_TMSHT	PR_APRV_BY	EIM_TIMESHEET	
S_TMSHT_ITEM	PR_ADJ_ITEM_ID	EIM_CONTACT1	
S_TMSHT_LINE	PR_ACTIVITY_ID	EIM_TIMESHEET	
	PR_PROJITEM_ID	EIM_TIMESHEET	

Table 9. Primary Child Columns That Have EIM Support

TABLE_NAME	PC_COL_NAME	EIM Table Mapping	Index Defined?
S_WRNTY_CVRG	PR_PROD_ID	EIM_WRNTY_CVRG	
	PR_PROD_LINE_ID	EIM_WRNTY_CVRG	
	PR_SRV_ORG_ID	EIM_WRNTY_CVRG	

Running a Delete Process

You may run a delete process after you have:

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

Run the delete process by completing the procedures in [Chapter 8, “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 interface 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 interface 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. To get a detailed log file, set Component Event Tracing for the EIM component. For more information on viewing the trace file, read [“Viewing the Task Info Log” on page 140.](#)

This chapter covers the process of merging data into the Siebel database. It covers preparing the interface tables, editing the configuration file, running the EIM process, and checking results.

Merge Overview

EIM reads information from the interface tables and then identifies rows to merge in the Siebel database tables.

During its multiple passes through the interface tables, EIM will:

- Initialize the interface tables for merge
- Select for merge the rows with matching user keys in the interface tables
- Merge child rows into the replacement rows
- Update child rows containing foreign keys that point to newly deleted rows

EIM provides comprehensive status information about each merge process. When the process ends, you should review this information.

EIM uses a combination of interface 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 adjusts the intersecting table rows to refer to the remaining rows. Duplicate child records of the deleted rows are also deleted 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 are also deleted.

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

- 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.

Merge Process

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

To merge data, EIM performs the following steps:

- 1** Initialize interface tables for merge.
- 2** Delete rows from the target base table that are specified in the interface table.

For deleted rows, set T_MERGED_ROW_ID to the ROW_ID of the merged-into (surviving) row; set T_DELETED_ROW_ID to the ROW_ID of the deleted base table row.

- 3** 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).

NOTE: If you have active mobile Web clients, you should enable transaction logging during merge operations. This captures the appropriate transactions for later synchronization.

Preparing the Interface Tables for Merge Processing

This section provides assistance in loading the interface 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 interface table row to be processed contains the appropriate values in the following columns. [Table 10](#) shows a merge example.

Table 10. 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 interface 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 interface table row whose ROW_ID and IF_ROW_BATCH_NUM columns identify the surviving or merged-into row, set this value to null. For interface 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 interface table.

For more information on special columns, read [“Interface Table Columns” on page 19](#), and for general information on interface tables, read [Chapter 2, “Siebel Interface Tables.”](#)

Editing the Configuration File for Merge Processing

This section describes the header and process sections that you will need in the EIM configuration file to properly configure EIM for a merge process. For general information on the EIM configuration file, read [Chapter 3, “EIM Configuration File.”](#)

Header Section

Parameters in the header section generally apply to all processes. For a description of the necessary contents in this section, read [“Header Section” on page 44.](#)

Process Section

Parameters in the process section apply only to that process and override any corresponding value in the header section for the specific process. 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.

For generic parameters that can be used in all EIM processes, read [“Process Sections” on page 45.](#)

Header and Process Parameters

This section 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, read [“Process Sections” on page 45](#).

UPDATE ROWS. Specifies whether the foreign key(s), which 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.

EIM SCHEMA CACHE. This caches the column relations. The default value is TRUE, which results in faster processing. If the value is set to FALSE, then the cache is not used.

SET BASED LOGGING. Specifies whether set based logging is enabled. Possible values are TRUE or FALSE. The default value is TRUE.

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. If a table is only partly read-only to mobile Web clients, it is safe to execute set-based transaction logging for changes to these columns only. This can be done by using the ONLY BASE COLUMNS and IGNORE BASE COLUMNS parameters during import, and turning on set based logging explicitly using the SET BASED LOGGING parameter. Set based logging is always the default for inserting and deleting. The SET BASED LOGGING parameter must be set to TRUE to allow set based logging on update.

NOTE: EIM ignores this parameter if Docking Transaction Logging is set to FALSE in the System Preferences view.

Working with EIM for Merge Processing

This section explains configuration file settings that direct EIM to perform merge processing.

Creating Temporary Indexes

Merge performance is improved if you create some specific temporary indexes first. For information, read [“Creating Temporary Indexes” on page 111](#).

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.

Handling Aborts of EIM Merge Processing

If an EIM merge process is aborted, base tables associated with merged 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, set the following parameters in the IFB configuration file so the EIM merge process performs only one commit and rollback when aborted:

```
COMMIT EACH TABLE = FALSE
```

```
COMMIT EACH PASS = FALSE
```

```
ROLLACK 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 configuration file so the EIM merge process runs in ongoing (row-by-row) mode:

LOG TRANSACTIONS = TRUE

SET BASED LOGGING = FALSE

Specifying Survivor Records

In a merge process, data from the record you select as the surviving record are preserved. Data from the other records are lost. Do not specify the same record as both the survivor and the victim. Otherwise it would be deleted. Make sure a record is specified as a survivor only once in a batch. Otherwise the EIM task fails.

Running a Merge Process

You can run a merge process when you have:

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

Run the merge process by completing the procedures in [Chapter 8, “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 interface 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 interface 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. To get a detailed log file, set Component Event Tracing for the EIM component. For more information on viewing the trace file, read [“Viewing the Task Info Log” on page 140.](#)

Merging Data

Checking Merge Results

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

- [“Running an EIM Process” on page 135](#)
- [“Viewing the Task Info Log” on page 140](#)
- [“Optimizing Performance” on page 148](#)

Running an EIM Process

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

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

You can start an EIM process by running a server task for the Enterprise Integration Mgr component. You can run the server task using the GUI or the command-line interface. For more information on running server tasks, read *Siebel Server Administration Guide, MidMarket Edition*.

CAUTION: If you are running EIM on a DB2 database, then set the database configuration parameters as described in *Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition* or EIM does 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.

On each pass, EIM processes one interface table and performs a particular action on all rows in that table for that batch. Most passes affect only the interface table's temporary columns; for example, resolving foreign keys. Passes in [Step 8 on page 62](#) (update), [Step 9 on page 62](#) (insert), and [Step 10 on page 62](#) (primary child keys) affect the base tables. All steps are performed for all columns used in the import process.

To run an EIM process using the GUI

- 1** Navigate to Enterprise Operations.
- 2** Click the Component Requests view tab.
- 3** In the Component Requests form, add a new record.
- 4** In the Component/Job field, click the Select button.

The Component/Jobs window opens.

- 5** In the Component/Jobs window, select the Enterprise Integration Mgr component and 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, read *Siebel Server Administration Guide, MidMarket Edition*.

- 6** Complete the rest of the fields and save the record.
- 7** In the Component Request Parameters list, add or change any component parameters for the EIM process and save the record.

[Figure 6 on page 138](#) shows an example of setting parameters in the Component Request Parameters list.

- a** In the Component Request Parameters list, add a new record.
- b** In the Name field, click the Select button.

The Job Parameters window opens.

- c** To select the EIM configuration file to use, select Configuration file and click OK.

- d** In the Value field, type the name of the EIM configuration file and click Save. The default value is default.ifb.

NOTE: You can use the Uniform Naming Convention (UNC) filename when specifying the EIM configuration file if you have read permission to the path.

- e** To set the batch number for the EIM process, repeat [Step a](#) through [Step b](#). Select Batch Number and click OK. In the Value field, type the batch number and click Save. The default value is 0.
- f** To select the EIM process from the configuration file, repeat [Step a](#) through [Step b](#). Select Process and click OK. In the Value field, type the process name and click Save.
- g** Optionally, to activate error flags, repeat [Step a](#) through [Step b](#). Select Error Flags and click OK. In the Value field, type 1 and click Save. The default value is 0.

For information on error flags, read [“Error Flags” on page 140](#).

- h** Optionally, to activate SQL trace flags, repeat [Step a on page 136](#) through [Step b on page 136](#). Select SQL Trace Flags and click OK. In the Value field, type 8 and click Save. The default value is 0.

For information on SQL trace flags, read [“SQL Trace Flags” on page 142](#).

- i** Optionally, to activate trace flags, repeat [Step a](#) through [Step b](#). Select Trace Flags and click OK. In the Value field, type in the appropriate value and click Save.

For information on trace files, read [“Trace Flags” on page 142](#).

NOTE: You will need to identify at least a batch number, process name, and configuration file for the task. If the batch number component parameter is set to 0, the batch number in the EIM configuration file (if any) will be used.

- 8** In the Component Requests form, click the menu button, and then click Submit request.

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

Figure 6 shows an example of running an EIM process as described in “To run an EIM process using the GUI” on page 136. It shows the Component Requests form and below that the Component Request Parameters list.

The screenshot displays the Siebel EIM GUI. The top navigation bar includes tabs for Enterprise Servers, Enterprise Component Tasks, Enterprise Tasks, Component Group Assignment, Component Requests, and Repeating Component Requests. The 'Component Requests' tab is active, showing a form with the following fields:

- ID:** 1-ZOSW
- Scheduled Start:** 7/31/2001 11:23:09 PM
- Server:** (empty)
- Submit Date:** 7/31/2001 11:23:09 PM
- Status:** Queued
- Expiration:** (empty)
- Request Key:** (empty)
- Actual Start:** (empty)
- Component Job:** Enterprise Integration Mgr
- Delete Interval:** (empty)
- Mode:** Asynchronous
- End Date:** (empty)
- Component:** (empty)
- Delete Unit:** (empty)
- Type:** Component

Below the form is the 'Component Request Parameters' list, which is a table with the following data:

Name	Value	Inheritable?	Required?	Fixed?
Batch Number	101			
Configuration file	sample.ifb			
Error Flags	1			
Process	IMPORT ACCOUNTS			
SQL Trace Flags	8			
Trace Flags	1			

Figure 6. Running an EIM Process

To run an EIM process using the command-line interface

- 1 Start the `svrvmgr` program in the command-line interface.

For information on `svrvmgr` program, read *Siebel Server Administration Guide, MidMarket Edition*.

- 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. If you do not specify a configuration file, the `default.ifb` configuration file will be used.

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

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, read *Siebel Server Administration Guide, MidMarket Edition*.

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.

Viewing the Task Info Log

The Task Info Log contains information about the results of an EIM process. 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.
- **Rowcount 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 Task Info Log will also contain the results of each flag. To get a detailed log file, set Component Event Tracing for the EIM component.

To view the Task Info Log

- 1** Navigate to the Tasks screen.
- 2** Click Task Info Log view tab.
- 3** In the Tasks list, select the task for the EIM process.

The log is displayed in the Task Info Log list.

Error Flags

To activate error flags, you must complete [Step g on page 137](#) when running an EIM process. Setting the Error Flags parameter to 1 produces a detailed explanation of rows that were not successfully processed.

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.

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 03:47:59 4/4/01 3:47 Warning: No rows in S_ORG_EXT
matched by expressions for export.

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

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

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

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

2001-04-04 03:47:59 S_ORG_EXT (Account)

2001-04-04 03:47:59 The match expressions specified for
exporting rows through this interface table

2001-04-04 03:47:59 did not match any of the rows currently in
the target base table.

2001-04-04 03:47:59 Since there were no matches for the given
match expressions, processing for

2001-04-04 03:47:59 this interface table was discontinued.
However, processing of other interface

2001-04-04 03:47:59 tables will continue.

2001-04-04 03:47:59 Recorded 1 group of failures.
```

SQL Trace Flags

To activate SQL trace flags, you must complete [Step h on page 137](#) when running an EIM process.

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.

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 i on page 137](#) when running an EIM process. 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.

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.

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 interface table processed. The following sample shows an EIM Trace Flag 1 output:

```
Initializing
      Loading configuration file  imacct.ifb  0s
      Opening server database    ora_dev   6s
      Loading Siebel dictionary      15s
Initializing                    21s
```

```

Import Accounts      14
  Importing  EIM_ACCOUNT
    Step 1: initializing IF Table      0s
    Step 4: resolving foreign keys    S_ORG_EXT    0s
    Step 5: locating existing row    S_ORG_EXT    0s
    Step 6: assigning new row IDs    S_ORG_EXT    0s
    Step 4: resolving foreign keys    S_ACCNT_POSTN  0s
    Step 5: locating existing rows    S_ACCNT_POSTN  0s
    Step 6: assigning new row IDs    S_ACCNT_POSTN  0s
    Step 4: resolving foreign keys    S_ADDR_ORG    0s
    Step 5: locating existing rows    S_ADDR_ORG    0s
    Step 6: assigning new row IDs    S_ADDR_ORG    1s
    Step 4: resolving foreign keys    S_ORG_INDUST  0s
    Step 5: locating existing rows    S_ORG_INDUST  0s
    Step 6: assigning new row IDs    S_ORG_INDUST  0s
    Step 4: resolving foreign keys    S_ORG_PROD    1s
    Step 5: locating existing rows    S_ORG_PROD    0s
    Step 6: assigning new row IDs    S_ORG_PROD    0s
    Step 4: resolving foreign keys    S_ORG_REL    0s
    Step 5: locating existing rows    S_ORG_REL    0s
    Step 6: assigning new row IDs    S_ORG_REL    0s
    Step 4: resolving foreign keys    S_ORG_SYN    0s
    Step 5: locating existing rows    S_ORG_SYN    0s
    Step 6: assigning new row IDs    S_ORG_SYN    1s
    Step 4: resolving foreign keys    S_ORG_TYPE    0s

```

```
Step 5: locating existing rows    S_ORG_TYPE    0s
Step 6: assigning new row IDs    S_ORG_TYPE    2s
Step 4: resolving foreign keys   S_TERR_ITEM   1s
Step 5: locating existing rows   S_TERR_ITEM   0s
Step 6: assigning new row IDs    S_TERR_ITEM   0s
Step 7: finding new foreign keys      4s
Step 9: inserting new rows    S_ORG_EXT    2s
Importing    EIM_ACCOUNT    15s
Updating primaries
Step 10: updating primary keys    S_ORG_EXT    3s
Updating primaries      3s
Import Accounts    14    18s
```

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 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

```

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] -----
```

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] -----
```

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 11](#).

Table 11. 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
```

Log File Detail Levels

You can control the level of detail in the log files.

To generate a detailed log file for EIM

- 1** Navigate to View > Site Map > Server Administration > Components.
- 2** In the Components list, select Enterprise Integration Mgr.
- 3** Click the Component Event Configuration view tab.
- 4** Set the log level for the following event types.
 - For event type SQL Tracing, set log level to 4.
 - For event type SQL Summary, set log level to 4.
 - For event type Component Tracing, set log level to 3.

You do not need to restart the Siebel Server to apply the changes. These settings take effect in the next EIM task execution.

- 5** When running an EIM task, set the following flags for the task.
 - Error flags = 1
 - SQL Trace Flags = 8
 - Trace Flags = 1

The EIM engine generates the detailed log file in the Siebel_Server\log directory.

Optimizing Performance

To optimize EIM:

- **Run EIM processes 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. Concurrent EIM processes can run against the same interface table only if they are operating on different batch numbers or if your database supports row level locking.

NOTE: Running EIM processes in parallel on a DB2 database may cause a deadlock when multiple EIM processes access the same interface table simultaneously. To avoid this potential problem, set the UPDATE STATISTICS parameter to FALSE in the EIM configuration file.

- **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, and IGNORE BASE COLUMNS. The ONLY BASE COLUMNS parameter is critical for the performance of an Integration Manager process updating a few columns in many rows.
- **Ignore account synonyms.** 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, 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.
- **Limit number of records for Merge process.** Limiting the number of records in your batch to 100 will significantly improve performance.

- **Use smaller batches.** Using multiple smaller batches rather than one large batch places smaller demands on resources and improves performance. For initial data loading, around 20,000 rows per batch works well. When loading into a production environment, use a batch size of 5,000 rows or less, preferably less than 2,000 rows.

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.

- **Use batch ranges.** Using batch ranges (BATCH = #-#) allows you to use smaller batch sizes. The maximum number of batches that you can run in an EIM process is 1,000.
- **Delete batches from interface tables upon completion.** Leaving old batches in the interface table wastes space and can adversely affect performance.
- **Disable transaction logging during the EIM process.** Disabling transaction logging will definitely improve performance; however, it must be balanced with the need for mobile users to reextract afterwards. To disable transaction logging, complete [Step 2 on page 74](#).
- **Perform regular table maintenance on interface tables.** Frequent insert or delete operations on interface tables can cause fragmentation in the table. Ask your database administrator to detect and correct fragmentation in the interface tables.
- **For IBM DB2, load a few batches of data into the interface table and run EIM for just one of these batches.** This will prime the statistics in the DB2 catalogs. Afterwards, do not update statistics on the interface tables, and run EIM with the parameter UPDATE STATISTICS = FALSE in the IFB file. This helps achieve consistent performance results when running EIM.

For more tips on optimizing EIM performance, read [“Performance Questions” on page 165](#).

Improving the Database Layout

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.

- Tables and Indexes.

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 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_SRV_REQ
- S_EVT_ACT
- S_OPTY
- S_ORG_EXT

- Organizations that plan large scale use of EIM should put their key EIM tables (based on their unique business requirements) on different devices from the Siebel base tables, because all tables are accessed simultaneously during EIM operations.

This chapter covers the frequently asked questions about using EIM. This may assist you in troubleshooting and data preparation.

General Questions

Question Does EIM support case values?

Answer Yes, EIM supports various case values defined for base table columns in Siebel Tools. EIM will adjust the case value of an interface table column according to the Force Case property of the corresponding base table column. For more information on case values, read [“Adjusting the Case of Values” on page 65](#).

Question What columns in each interface table are mandatory in order for a row to be loaded? What must their values be? Which columns (such as foreign keys) are validated against other tables?

Answer Several columns are mandatory. Others are *conditionally mandatory*, depending on the conditions of your import. Your Siebel application offers two methods for determining mandatory columns. You can use Siebel Tools to view each column in an interface table and its target base table columns. You can also refer to *Siebel Interface Tables Reference*. By adhering to Siebel’s recommended import sequence, you make sure that the appropriate data dependencies are established.

Question Does EIM support importing data from ACT!?

Answer You can only use EIM to import bulk data from legacy systems into the Siebel database. Only ACT! 2.0 and ACT! 3.0 have File/Import functionality for data import into Siebel. You can use “Exporter for ACT!” to export ACT! 4.0 or 2000 Contacts, Notes/History, Activity, Group, Sales and E-Mail data into comma-delimited files. For information on ACT! products, visit their website at <http://www.actaddons.com>.

Frequently Asked Questions

General Questions

- Question** Will EIM interfere or affect Visual Basic code built into the application?
- Answer** EIM will not interfere with Siebel VB code because Siebel VB works at the business component and business object level, and EIM works at the table level.
- Question** Does EIM evaluate properties on Business Component (BC) level (such as default value or LOV type)?
- Answer** EIM does not evaluate the logic from the BC Layer (or above); it only evaluates the data layer, tables, columns and their properties. The LOV Type on the table level and BC level should thus be always consistent.
- Question** How does EIM treat empty strings?
- Answer** EIM translates empty strings into NULL.

Import Process Questions

Question We successfully imported most of our interface table rows and want to delete them. However, we want to leave rows that were not fully imported so that we can examine and correct them. How can we do this?

Answer Each interface 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 interface 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_ORD_EXT_STA = 0 and T_ADDORG_STA = 0)
```

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

Frequently Asked Questions

Import Process Questions

Question Does EIM support importing multiple addresses for accounts?

Answer Use EIM_ACCOUNT to import accounts and their multiple addresses and set ACC_PR_ADDR to Y to indicate the primary address for each account.

Question Does EIM support importing multiple team members for accounts?

Answer 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.

Question How do I populate the PR_PROD_LN_ID column in the S_PROD_INT base table?

Answer Complete 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.

Question After successfully importing quote information, my users are unable to edit their quotes. Why?

Answer You probably initialized the APPROVED_FLG field to Y for each quote. A Y makes the quote read-only and not editable by the user. Instead, set this field to N or leave it blank.

Question How can we import contacts so that they are visible in the Contact list?

Answer You need to use EIM_CONTACT to import into S_PARTY, S_CONTACT, and S_POSTN_CON. Make sure S_POSTN_CON.POSTN_ID reference 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 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.

Question How can data be imported into List of Values (LOV) tables?

Answer When importing data from interface 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 interface 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 a LOV table using the S_ORG_EXT_XM table as an example.

To import data into a LOV table

- 1** Using Siebel Tools, find the LOV type for the S_ORG_EXT_XM table.
 - 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.

NOTE: Although the TYPE column in the S_LST_OF_VAL and S_ORG_EXT_XM tables have the same name, they are not the same column and do not need to be populated with the same values.

- 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.

NOTE: If you are using the Siebel client to import data, the Siebel client will check the PickList property for each field in a certain business component. If you are using EIM to import data into base columns, EIM will check the LOV Type property defined for each column in a base table. It is possible that you could insert data into a particular column using the Siebel client, but not using EIM, because they are looking at different properties.

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. For information on importing data, read [Chapter 4, “Importing Data.”](#)

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.

NOTE: Although the `TYPE` column in the `S_LST_OF_VAL` and `S_ORG_EXT_XM` tables have the same name, they are not the same column and do not need to be populated with the same values.

Question How can I import multiline fields such as addresses?

Answer 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 multivalued field control.
```

Question How do I import exported rows into target and secondary tables?

Answer 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 interface table row will be created for every target table row, and a separate interface 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.

Question How do I import international phone numbers using EIM?

Answer 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 will be treated as an U.S.A phone Number. This will lead 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.

Question How do I import URL links into the S_LIT base table?

Answer 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.

Export Process Questions

Question When data is exported using an interface table (such as contacts), why doesn't data from related child tables (such as opportunities) get exported?

Answer For all columns to export using an interface table (both data from the base table and data from related child tables), you need to add or modify the following line in the IFB configuration file:

```
EXPORT ALL ROWS = TRUE
```

NOTE: Rows from child tables of related child tables are not exported unless they have been mapped.

Delete Process Questions

Question What is the behavior of the EIM delete process?

Answer The delete process performed by EIM is cascade delete. The data deleted is not restricted to the base tables mapped to the interface table that you specified in the delete process, but all child records as well. 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 matches this criteria to be deleted from the database.

Question How do I use the DELETE EXACT parameter to delete data from base tables other than the target base table?

Answer 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 interface 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.

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 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 configuration 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 will delete all rows from the S_ORG_PROD table that have user keys that match the rows in your interface table.

Question Is it possible to delete or merge rows that have the same primary user key and different conflict IDs using EIM?

Answer This is not possible. 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 interface table will be marked as AMBIGUOUS.

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

Question Can I remove unnecessary seed data by deleting all rows from the S_LST_OF_VAL base table?

Answer No, you should 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.

Merge Process Questions

Question What is the behavior of the EIM merge process?

Answer Data from the record you select as the surviving record are preserved. Data from the other records are 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.

Question Why is the IF_ROW_STAT column set to the value NO_SUCH_RECORD after running a merge process?

Answer If you do not correctly populate all the user key columns, the merge process will fail and the IF_ROW_STAT column in the interface 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.

Question Can EIM be used to merge rows from secondary tables?

Answer 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.

During EIM merge, EIM will merge rows from the target base table, and update the rows from the secondary tables to reference the surviving target base table rows. If you try to merge two secondary tables rows into one row by populating the interface table with user keys of the secondary table rows, the interface table victim and survivor rows will have the same target base table user keys values, which will cause the target base table row to be deleted.

Performance Questions

Question How can I tune my EIM batches to improve performance?

Answer You should try the following options to improve EIM performance:

- Verify that all indexes exist for the tables involved.
- Limit tables and columns to be processed using ONLY BASE TABLES/COLUMNS configuration parameters to minimize EIM processing.
- Consider switching off transaction logging during the EIM run. This improves performance. However, it must be balanced with the need for mobile users to reextract afterward.
- Altering batch sizes typically resolves most performance issues. The batch size is dependent upon which type of EIM process you are running. The use of larger batch sizes for certain types of EIM processes could create a performance degradation.

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 interface tables. Frequent insert or delete operations on interface tables can cause fragmentation. Consult your database administrator to detect and correct fragmentation in the interface tables.
- Delete batches from interface tables on completion. Leaving old batches in the interface table wastes space and could adversely affect performance.
- Run independent EIM jobs in parallel. EIM jobs that have no interface or base tables in common can run in parallel.

- Set the USING SYNONYMS parameter to FALSE in the IFB configuration file to indicate that account synonyms do not need to be checked.
- Avoid using the UPDATE PRIMARY KEYS parameter in the IFB configuration file.
- If all else fails, set Trace Flags = 1 and SQL Trace Flags = 8 and rerun the batch. Use the resulting task log to identify slow-running steps and queries.

Question What is the recommended number of rows that can be loaded in a single batch?

Answer 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, you should limit the number of records in the interface 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 interface table. Under no circumstances should you have more than 250,000 rows in any single interface table because this reduces performance.

NOTE: The number of rows you can load in a single batch may vary depending on your physical machine setup. To reduce demands on resources and improve performance, you should always try to use smaller batch sizes.

Question How can I speed up deletes and merges involving S_ORG_EXT?

Answer 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.

Troubleshooting Questions

Question Should EIM stop processing when multiple rows fail a pass?

Answer 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.

Question Which rows failed the pass? Did some or all of the values in these rows get loaded? If so, which values loaded successfully and which did not?

Answer Failures can occur at several steps; each of these types has a different cause. [Step 4 on page 61](#) processes foreign keys and bounded picklists.

- A row fails this step if the foreign key developed from values in the interface 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).
- 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 (EIM_CONTACT.MID_NAME) and LOC (location) for accounts (S_ORG_EXT.LOC).
- 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 interface table, Step 7 resolves foreign keys into the data to be imported.
- 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.

None of the other steps produce failures, although any step can encounter an SQL error that might halt processing. EIM imports data on a best-effort basis, loading any records it can and ignoring records that have failed.

Question Why does the EIM batch fail with the following error message:

```
Error 205: Missing Temporary Column for S_XX table from EIM_XX
table
```

Answer This error message indicates that EIM is unable to find a correct mapping from the interface table to a column in the base table. Possible causes include:

- An invalid EIM Mapping. In some cases, it is possible to build an invalid mapping. Reselecting the target table and column for the mapping using Siebel Tools will normally rectify the problem.
- An invalid setting of the column's property. It is possible to use Siebel Tools to set properties such as Foreign Key Table for your extension column, although they are not supposed to be set for extension columns.
- The EIM mapping and extension columns have been deactivated, but the repository and schema have not been synchronized. After deactivation, the Apply and Activate buttons must be used to synchronize the repository and database to prevent EIM from using them.

NOTE: Clicking the Activate button in Siebel Tools will force the dicccache.dat file in the Siebel Server bin directory to be refreshed.

- The extension column was added prior to an upgrade, and the upgrade process removed some of the EIM mapping information. If you are affected by this problem, email support@siebel.com to request a patch.
- Your database installation failed. Check the installation logs to make sure that there were no unexpected errors.

Question After importing data using EIM, why isn't the data visible in some views or applets?

Answer Some values that needed to be imported for a particular view or applet to display imported data may not have been imported. To determine what 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. 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.

Question Why does an EIM process fail with following error message:

```
Error 405: Too many batch numbers in range
```

Answer Too many batches are set for the EIM process. The maximum number of batches you can run in an EIM process is 1,000.

Question Why does an EIM import process fail with the following error message:

```
Error 413: Column S_XXX.XXX table not in process
```

Answer This error message may be generated when you have specified one or more interface tables using the TABLE parameter in the IFB configuration file, but failed to specify user key columns and required columns from all related base tables when using the ONLY BASE COLUMNS parameter.

Question Why does an EIM process fail with the following error message:

```
Error 999: Rownum range [XXX] too large
```

Answer The batch that you are trying to process contains too many rows. Typically, you should not exceed 100,000 rows in a single batch.

Question Why does an EIM export process fail with the following error message:

```
Error 999: Doubly indirect joins for XXX too complex to export
```

Answer EIM does not support doubly indirect joins. This error will be generated if you try to run an EIM export process to the following interface tables:

- EIM_ACCSRCPIDTL
- EIM_CRSE_TSTRUN
- EIM_IC_CALC
- EIM_IC_PCMP_DTL
- EIM_IC_PERF_HST
- EIM_MDF

Question Why does EIM hang when updating the S_LST_OF_VAL base table?

Answer If you are running Siebel applications on an IBM DB2 database, this problem 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.

Question Why does EIM produce wrong data when updating tables in Oracle?

Answer When there are 255 or more NVL functions in an update statement, Oracle will update 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.

Enterprise Integration Manager Error Messages

A

This appendix lists the error codes and associated text that the Siebel Enterprise Integration Manager (EIM) may generate during processing. [Table 12](#) lists these error codes, the message text, and a description of each error.

EIM Error Codes

For each error, EIM writes this information to the EIM log file (if you specified that one be used).

NOTE: The message text that appears in [Table 12](#) is generic. However, for most errors, EIM generates more specific information about the exact cause of the error. Both the generic and the specific error messages appear in the Server Process Log; only the specific error message appears in the EIM log file.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
100–199, 998, 999	Internal errors.	These errors indicate a problem with the operating environment or the Siebel installation.
101	Invalid arguments to function.	EIM detected internal inconsistencies.
102	Too little memory to perform operation.	The machine on which the process is running can no longer allocate memory. Close one or more other processes while you are running EIM.
103	Name is not a valid identifier.	EIM detected an invalid parameter name in the configuration file. If you have modified the configuration file, check and correct any spelling errors.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
104	Requested entry not found.	EIM could not locate a value referenced in the configuration file. If you have modified the configuration file, review it and make the appropriate correction.
105	Operating system error.	EIM detected an operating system error. For example, the operating system may deny access to a directory in which EIM is attempting to create a file.
106	Functionality not yet implemented.	The configuration file is indicating that EIM is to perform a task not currently supported. Make sure that you are using the most current release of EIM.
107	ODBC (SQL) error.	The connected database returns an error during execution of an SQL statement. This generally indicates a configuration or resource failure on the database.
998	Usage warning (see detail information).	EIM detected input inconsistencies that are not fatal, but that you should know about. This error does not abort processing and is always reported with more detail.
999	Internal failure (no error code).	EIM detected an unexpected condition that is not covered by another error code. This error indicates a problem with EIM itself or with the Siebel applications database installation. This error is always preceded by a more specific error message (or messages) that indicates the problems leading to this result.
200–299	Exit status errors.	These errors indicate that EIM exited abnormally. (The exit status is 0 on successful completion.)
201	Invalid command line arguments.	EIM detected one or more arguments that it could not process. Check and correct the arguments and their spelling in the command line that initiated the process.
202	Invalid SIEBELHOME directory.	EIM could not locate the directory named in the /ROOTDIR argument of the command line that initiated the process. Check and correct the directory name.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
203	Invalid file to log to.	The file named in the /LOGFILE argument did not conform to DOS file-naming conventions. Check and correct the filename.
204	Invalid configuration file to load initially.	EIM detected that the file named in the /CONFIG argument of the IFMGR command line did not have a valid name. Make sure that the filename conforms to DOS naming conventions and that the file extension is IFB.
205	Failed to load the application dictionary.	Problems with the Siebel applications dictionary prevented EIM from loading the database schema.
206	Failed to run initial process.	EIM detected an error when loading or running the initial process named in the configuration file.
207	Unable to log in to the database.	A connection could not be established using the specified ODBC source, user name and password, and tableowner.
208	Aborted due to user interrupt.	You interrupted EIM with a CTRL-BREAK from the keyboard, or some operating system event (such as shutdown) terminated the process.
209	Errors occurred during processing.	Errors occurred that were reported in the log file and the Server Process Monitor.
210	Failures occurred during processing.	Failures occurred that were reported in the log file and the Server Process Monitor.
211	Failed to parse and define extended parameters.	Error occurred in parsing and defining extended parameters. Check to make sure that the parameter follows the naming convention.
212	This extended parameter is undefined.	The parameter is present in the IFB file, but it does not have a value assigned to it. Check to make sure that the parameter is assigned before using it.
213	Failed to resolve this extended parameter.	Error occurred in resolving this parameter. Log all the process parameters into a file by specifying "IFBFileName = <i>filename</i> " and examine <i>filename</i> to determine the source of the error.
214	Failed to resolve this parameter in maximum allowed recursions.	Error occurred in resolving this parameter within the maximum allowed level of nesting (default = 10). Change the default by giving the extended parameter MAX_NEST_SUBST.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
301–399	Configuration file load errors.	
301	Invalid section beginning in configuration file.	EIM detected a syntax error at the beginning of the header section, or the section contents were invalid. Check that the header section begins with exactly [Siebel Interface Manager] and that each of its parameters is valid.
302	Configuration file does not begin with a section.	The first nonblank, noncomment line in the configuration file was not the beginning of a process section. Correct the configuration file to begin with [Siebel Interface Manager].
303	Variable name is not a legal token.	The name of a parameter in the configuration file is not valid. Note that all the text before the equal sign (=) is considered to be the parameter name. Correct any invalid variable names.
304	No value part of the assignment.	A parameter assignment was missing the equal sign (=) or had no value after the equal sign. Make sure that parameter names and values are separated by an equal sign and that each parameter has a value.
306	Invalid token for right-hand side.	The value of a parameter was not quoted, but could not be interpreted as a token. Check that values which are not simple tokens are enclosed in quotation marks.
307	Invalid string for right-hand side.	The value for a parameter began with a quotation mark, but could not to be interpreted as a string. Check that quotation marks are paired. Also check the validity of the quoted string.
308	Trailing garbage at end-of-line.	There is extra text after the parameter value. Delete the trailing text.
309	End-of-line in quoted string.	The end of the line was found before the string-terminating double-quote mark ("). Make sure the string is properly ended on this line or continued with \ at the end of the line.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
310	End-of-file in parenthesized expression.	The end of the file was found before the closing parenthesis for an expression value. Make sure each open parenthesis is matched with a closing parenthesis.
311	Unexpected number of values given.	More values were specified for the variable than were expected. Values are separated by unquoted commas.
312	No header section is found in IFB file.	The IFB file is missing the header section. For directions on setting up a header section, read “Header Section” on page 44 .
400–499	Import process load/run errors.	
401	Missing process section in configuration file.	The RUN PROCESS parameter of the configuration file header section named a process for which there is no corresponding process section. Check and correct the process name provided by the RUN PROCESS parameter or create a process section for the named process.
402	Invalid process type in section.	The PROCESS TYPE parameter of the configuration file process section contained a value other than IMPORT or SHELL. Correct the parameter value to one of the above.
403	Error resolving includes.	An INCLUDE parameter provided a process name that EIM could not find, or an error occurred when loading the process named in the INCLUDE parameter. Check all INCLUDE statements to make sure that they refer to valid sub-process sections.
404	Error getting interface table names.	One or more of the interface table names specified by the TABLE parameters were invalid, or no TABLE parameters were specified. Check and correct TABLE parameter errors.
405	Invalid batch number specified.	The batch number specified in the process was not valid or was not specified. Batch numbers must be positive integers of fewer than 15 digits.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
406	Invalid choice value in assignment.	The parameter value did not match any of the expected values. Review and correct parameter values.
407	Invalid Boolean value in assignment.	The value of the parameter should have been TRUE or FALSE, but some other value was found. Correct the parameter value.
408	Invalid numeric value in assignment.	The value of the parameter should have been a number, but some other value was found. Correct the parameter value.
409	Invalid table or column in assignment.	The value of the parameter should have been a table or column name, but the value found is not a valid SQL table or column specification. Correct the parameter value.
410	Invalid report specification.	The value of the parameter should have been a report specification consisting of an optional prefix (“TAB”, “CSV”, or “columns”), a comma, and the filename used to save the report. Correct the report specification.
411	SQL WHERE clause expression invalid.	The value was expected to be an SQL WHERE clause expression fragment but was unable to be processed as such. Make sure the value is a parenthesized SQL WHERE clause fragment.
412	Invalid ONLY/IGNORE BASE TABLES specification.	The value for the ONLY BASE TABLES or IGNORE BASE TABLES parameter should have been a list of base table names, separated by commas. Correct the parameter value.
413	Invalid ONLY/IGNORE BASE COLUMNS specification.	The value for the ONLY BASE COLUMNS or IGNORE BASE COLUMNS parameter should have been a list of base table column names, separated by commas. Correct the parameter value.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
420	Subprocess failed to execute.	One of the subprocesses identified by an INCLUDE statement failed to execute. Check the EIM Log File for an indication of why the failure occurred.
421	Interface table not in DB schema.	The TABLE parameter named a table that is not identified as a Siebel interface table. Check the spelling and syntax of all values for TABLE parameters. Siebel interface table names begin with EIM_.
422	No rows in given batch to process.	EIM detected no rows that were eligible for processing and that had the batch number specified by the BATCH NUMBER parameter. Check to be sure the correct batch number was specified. If it is correct, the rows with that batch number may have a data error.
423	Unable to register with docking log.	Docking Transaction Logging is on, but EIM could not contact the transaction log. There is an installation problem with the Siebel applications database.
424	All interface tables failed.	Importing failed for all specified interface tables. This generally indicates a data problem.
425	All rows in interface table failed.	Processing for all rows in this interface table failed. This generally indicates a data problem. Either data was not available for insert, update, delete or merge, or all rows had errors. For example, all rows may be duplicates in a merge process or no rows are based on the user key filled in the interface table for delete and merge processes.
426	All batches in run failed.	Processing for all batches specified for this run of EIM failed. This generally indicates a data problem.
430	Invalid character seen in string.	Characters were encountered in the WHERE clause fragment that could not be properly interpreted. Make sure literal string values are enclosed in quotes.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
431	Invalid comparison operation.	The comparison operator in a match expression was not understood. Make sure all comparisons in the WHERE clause fragment are simple column-to-value comparisons.
432	Invalid column name for comparison.	The left-hand side of a comparison in a match expression was not a column. Make sure all comparisons in the WHERE clause fragment are simple comparisons.
433	End-of-file in quoted string value.	A string being read was not terminated before the expression ended. Make sure all strings are properly terminated with a closing quote. To embed a quote in a string, double it.
434	Unexpected token in expression.	Unexpected value seen while parsing a matching expression. This represents a syntax error in the SQL WHERE clause fragment.
435	Invalid value for column type.	The value to be compared with this column could not be converted to the column's type. Make sure the comparison value is of the correct type. Date/time values should be in the ODBC format: yyyy-mm-dd hh:mm:ss.
440	Default/fixed column not found in table.	The configuration file specified a default column or a fixed column, but the column does not exist in the interface table. Check the spelling and syntax of the column name.
441	Invalid filter query expression.	The expression provided for the FILTER QUERY parameter did not conform to SQL WHERE clause syntax. Invalid FILTER QUERY expressions usually cause the SQL statement generated in Step 3 to fail with an ODBC error. Correct the SQL statement and resubmit the process.
442	Update of primary child columns failed.	Updating the primary child key columns failed. This indicates an error in Step 10 of EIM processing.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
450	Cannot operate on child table directly.	This interface table is a child of another. Export, delete, and merge operations must be done on the parent table instead of this child. Make sure the parent table is included in the process as well as the child.
451	Cannot operate on this IF table directly.	This interface table has no defined target and thus cannot be used in export, delete, or merge operations. This is an import-only table.
452	Match expression invalid for IF table.	A match expression uses columns that are not present in the current interface table. Make sure the match expressions are for the specific interface tables being processed.
453	Specified column is not exportable.	The requested column cannot be exported. Either this column is not being used or it does not exist. Omit this column from the list of columns to be exported.
454	Target table for IF table has no user keys.	This interface table does not support delete exactly or merge (which require user keys) because its base table has no user keys. You can delete using match expressions, but you cannot merge through this interface table.
460	Column value too long for base table.	The value in the interface table column is too long to fit into the base table column. Columns are limited to 4095 bytes.
601–699	Report errors.	These errors indicate a problem generating a report requested by a process.
601	Column does not exist in report.	The report tried to access a column that was not defined.
602	Column already exists in report.	The report tried to add a column that was already defined.
603	Row does not exist in report.	The report tried to access a row that was not defined.

Table 12. EIM Error Codes

Code	Message Text	Description/Recommendation
604	Invalid report type for operation.	The given report type could not support the requested operation.
605	Cannot create report file.	The report file could not be created in the file system, probably as a result of an invalid filename.
606	Cannot update report table.	An SQL report could not update the table indicated by the report specification, usually the result of specifying an invalid database table.

This appendix explains how to use the Siebel File System Cleanup Utility to cleanup the file attachment directory of any unused file attachments.

The Siebel File System Cleanup Utility is a command-line utility, named `sfscleanup.exe`, located in the `bin` directory within the Siebel Server root directory. `Sfscleanup.exe` will process all files in the file attachment directory and perform one of several operations to each file depending on the file type and the parameters that you set. For descriptions of the run-time parameters that you can set when running `sfscleanup.exe`, see [Table 13 on page 182](#). For descriptions of the file types and the associated operation that will be performed by `sfscleanup.exe` during processing, see [Table 14 on page 183](#).

Cleaning up the File Attachment Directory

As part of routine maintenance, you should use the `sfscleanup` utility to remove extraneous files from the file attachment directory.

To clean up the file attachment directory using `sfscleanup.exe`

- 1** At the command prompt, change directory to the `bin` directory within the Siebel Server root directory.
- 2** Run `sfscleanup.exe` using the parameters listed in [Table 13](#) as shown in the following example:

```
sfsfcleanup /u sadmin /p secret /f \\server1\files /x  
\\server1\logs\sfsfcleanup.log
```

Table 13. Sfsfcleanup.exe Parameters

Parameter	Value	Description	Required?
/u	<i>Username</i>	Username ID.	Y
/p	<i>Password</i>	Username password.	Y
/c	<i>ODBC data source</i>	Set this value to the ODBC data source. Default value is set to the environment variable, SIEBEL_DATA_SOURCE.	N
/d	<i>Siebel table owner</i>	Set this value to the Siebel table owner. Default value is set to the environment variable, SIEBEL_TABLE_OWNER.	N
/f	<i>Path for file directory</i>	Set this value to the path for the file attachment directory. Do not append att to the file attachment directory path.	Y
/x	<i>Path for output file</i>	Set this value to the path for the output file.	N
/m	<i>Path for move directory</i>	Set this value to the path for the directory where discarded files will be moved.	N
/n	<i>Remove old revisions</i>	Determines if old versions of file attachments will be removed. To remove old versions, set this value to Y. Default value is N.	N

If you specified an output file using the /x parameter, sfscleanup.exe generates a log file listing the operations that were performed. The output file is a tab-delimited text file that contains the following columns:

■ File Name

This column lists the names of all files that were processed.

■ File Type

This column lists the types of the files that were processed. [Table 14](#) lists the possible file types and the associated operation that will be performed by sfscleanup.exe during processing.

Table 14. File Types and Associated Operation

File Type	Description	Operation¹
CURRENT	The file has a corresponding record in the file attachment database table.	KEPT
NEW	The file is less than 5 minutes old. Sfscleanup.exe does not check for the file in the file attachment database table.	KEPT
ORPHAN	The file does not have a corresponding record in the file attachment database table.	DELETED ²
OLDREV	The file has a corresponding record in the file attachment database table but the record indicates a version different from the file.	KEPT ³
INVALID	The file (or directory) is not a file attachment. If sfscleanup.exe is attempting to delete a directory that is not empty, the operation errors out. This gives you an opportunity to review the files contained within the directory before deletion.	DELETED ²

1. For descriptions of each operation, see [Table 15 on page 184](#).
2. If you used the /m parameter to set a move directory, then the operation performed on the file will be MOVED.
3. If you set the /n parameter to Y, the operation performed on the file will be DELETED (or MOVED if you used the /m parameter to set a move directory).

- Operation

This column lists the type of operation that was performed during processing. [Table 15](#) lists the types of operation that sfscleanup.exe may have performed during processing.

Table 15. Operations

Operation	Description
KEPT	The file was kept.
DELETED	The file was deleted.
MOVED	The file was moved to the directory specified by the /m parameter. Files will only be moved if you used the /m parameter.

Index

A

- aborted processing
 - delete process 110
 - merge process 131
- accounts
 - multiple addresses 156
 - multiple team members, importing 156
- ACT!, importing data from 153
- addresses
 - accounts with multiple addresses 156
 - multiline, importing 160
- attachment files
 - columns 20
 - deleting 109
 - sfscleanup.exe, running 181, 184

B

- base tables
 - base table to interface table mapping, sample 30
 - column mappings, viewing 27, 28
 - conditions for mapping 22
 - deleting data 104
 - deleting data from target base table 100, 101
 - deleting selected rows 106
 - interface table mapping without user keys 31, 33
 - interface table mappings, about 21
 - loading data directly 13
 - multi-org, capability 83
 - nonexistent mapping, remedy 21
 - as parent in non-target table
 - mapping 22, 23
 - parent-to-child pointer 23
 - primary child foreign keys, role of 23
 - S_LST_OF_VAL, processing hung 170

- sample view 27
- viewing deleted rows 124
- viewing mappings 26, 29
- base tables, non-target
 - DELETE EXACT parameter, use of 162
 - deleting rows, parameter 100
- batch failure 168
- batch files
 - importing data 73
 - maximum number of rows 169
- BATCH parameter 40, 46
- batches, maximum number 169
- BU_ID foreign key column 83

C

- call lists 78
- child records, about merging 125
- CLEAR INTERFACE TABLE parameter
 - deleting data 104
 - deleting data, process initialization 100
 - description 41
 - export setting 94
- column mappings
 - about 21
 - between extension table and interface table columns 21
 - implicit and explicit mappings 23
 - primary child foreign keys 23
 - sample view 28
 - viewing 27, 28
- columns
 - See also* column mappings
 - common to interface tables 19
 - data deletion values 102
 - file attachments, initial values 65
 - file attachments, required columns 20
 - initial values for special columns 64

- loading problems 153
 - mandatory, determining 153
 - organization mapping, about 21
 - PR_type columns 23
 - primary child foreign keys 23
 - special column values for merge
 - process 133
 - special columns, about 19
 - special columns, export
 - considerations 91
 - temporary columns 20
 - user key columns, failure to specify 169
 - xxx_BU column 21
 - commit and rollback 110, 131
 - COMMIT EACH PASS parameter
 - description 40
 - editing for import 69
 - COMMIT EACH TABLE parameter
 - description 40
 - editing for import 69
 - COMMIT OPERATIONS parameter
 - description 41
 - editing for import 70
 - CON_PRIV_FLG column 77
 - configuration file
 - about 38
 - comment lines, specifying 42
 - commit and rollback parameter
 - settings 131
 - continuation character 42
 - DB2 caution 135
 - extended parameters 50, 53
 - header section parameters 44, 45
 - header section, export parameters 94, 95
 - parameters for exporting data 93
 - parameters, about 39
 - parameters, table of 40, 42
 - rollback segment, defining 48
 - transaction logging parameters for merge 132
 - configuration file, editing
 - delete processing 103, 106
 - export process 93
 - import process 66, 130
 - merge process 129
 - configuration file, errors
 - invalid parameter name 171
 - load errors 174, 175
 - task not supported 172
 - value not located 172
 - configuration file, process section
 - deleting all rows in table 105
 - deleting data, parameter settings 103, 104
 - deleting rows, selectively 106
 - export edits 93
 - export parameters 94, 95
 - general process parameters, described 45, 49
 - import parameters 66, 67
 - merge code example 129
 - updating rows, parameter setting 106
 - CONNECT parameter 40, 44
 - contacts
 - importing, FAQ 157
 - private contacts, importing 77
 - CURRENT_DATETIME parameter 52
 - CURRENT_USER parameter 52
- ## D
- data
 - importing. *See* importing data
 - initial values for special columns 64
 - Database Extensibility, specifying mappings 21
 - databases
 - DB2 configuration parameter settings, warning 135
 - DB2, parallel processes warning 148
 - high-level process flow diagram 16
 - server space requirements for import 84
 - DB2 databases, EIM processing
 - caution 135
 - DEFAULT COLUMN parameter
 - description 41
 - editing for import 68

- default.ifb file 22, 38
 - DELETE ALL ROWS parameter
 - deleting data 101
 - deleting data from interface tables 105
 - description 42
 - sample code 107
 - DELETE EXACT parameter
 - delete based on user keys 163
 - deleting data 100, 104
 - description 42
 - DELETE MATCHES parameter
 - code example 104
 - deleting data 101
 - description 42
 - DELETE ROWS parameter
 - code example 106
 - description 42
 - deleting data
 - aborted processing, special considerations 110
 - about 99
 - all rows 107
 - child rows 101
 - DELETE EXACT parameter and non-target base tables 162
 - deleting all rows 105
 - deleting rows selectively 106
 - editing the configuration file 103, 106
 - preparing interface tables 102
 - process overview 100, 101
 - results, verifying 124
 - row selection methods 99
 - rows successfully imported, sample code 155
 - same primary user key, different conflict IDs 163
 - special column values 102
 - transaction logging for Mobile Web Clients 101
 - unimported rows 155
 - updating rows, parameter 106
 - user keys and DELETE EXACT parameter 163
 - diccache.dat file, refreshing 168
 - documents, importing 81
 - DUP_RECORD_EXISTS import status 87
- ## E
- EIM processing
 - DB2 databases, configuration parameter caution 135
 - error codes 171, 180
 - error flags 140
 - file attachment status log, creating 146
 - inconsistency warning 172
 - interface table mappings, warning log 146
 - internal inconsistency error 171
 - multiple row failure 167
 - optimizing performance 148, 149
 - pausing or stopping, warning 137
 - performance tuning suggestions 165
 - preparations 135
 - process failure, Error 405 169
 - process failure, Error 999 169
 - recommended number of rows 166
 - results log, viewing 140
 - running from the command line 139
 - running from the GUI 136, 138
 - SQL trace flags 142
 - step-oriented task log, creating 142, 144
 - trace flags 142
 - user key override log, creating 145
 - user parameter substitution log, creating 144, 145
 - warning 137
 - EIM_type interface tables, about 21
 - EIM_IC_CALC interface table 92
 - EIM_OPTY_DTL 22
 - eim_task#.trc file 75
 - Enterprise Integration Mgr component 135
 - Error 205 168
 - Error 405 169
 - Error 413 169
 - Error 999 169
 - error codes 171, 180

- configuration file load errors 174, 175
- Error codes 200 - 209 172, 173
- Error codes 301 - 399 174, 175
- Error codes 401 - 499 175, 179
- Error codes 601 - 699 179, 180
- Errors 100 - 109 171
- exit status errors 172, 173
- import process load/run errors 175, 179
- internal errors 171
- report errors 179, 180
- error flags 140
- exit status errors 172, 173
- EXPORT ALL ROWS parameter
 - description 41
 - export setting 95
 - selected rows setting 96
- EXPORT MATCHES parameter
 - description 42
 - export setting 94
- exported rows, importing 160
- exporting data
 - about 89
 - all rows 96
 - configuration file, editing 93
 - export process 89
 - export processes, types 96
 - extracting data from interface tables 98
 - interface table preparation 91
 - to organizations 97
 - results, verifying 98
 - selected rows 96
 - starting export process 97
 - troubleshooting child tables 161
- extended parameters
 - defining 50, 52
 - predefined 52, 53
- extension tables, column mappings 21
- F**
- FAQs
 - accounts with multiple addresses 156
 - ACT!, importing data from 153
 - batch failure 168
 - case values 153
 - column loading problems 153
 - contacts, importing 157
 - deleting data from non-target base table 162
 - EIM and Visual Basic code 154
 - IF_ROW_STAT column set to NO_SUCH_RECORD 164
 - importing exported rows 160
 - LOV table imports 157, 159
 - LOV Type 154
 - maximum number of batches 169
 - maximum rows in a batch 169
 - multiline addresses, importing 160
 - multiple row failure 167
 - multiple team members, importing 156
 - performance, tuning 165
 - PR_PROD_LN_ID column, populating 156
 - quote problems 156
 - recommended number of rows (performance) 166
 - rows, deleting or merging 163
 - S_LST_OF_VAL base table, processing hung 170
 - troubleshooting child table export 161
 - unimported rows 155
 - unsuccessful merge setting 164
 - user key columns, failure to specify 169
- file attachments
 - deleting 109
 - directory cleanup 181, 184
 - importing 81
 - status log 146
 - updating 82
- FILE_EXT column 20
- FILE_EXT row, initial value 65
- FILE_NAME column 20
- FILE_NAME row, initial value 65
- FILE_SRC_TYPE column 20
- FILE_SRC_TYPE row, initial value 65
- FILTER QUERY parameter
 - description 41
 - editing for import 66
- FIXED COLUMN parameter
 - description 41
 - editing for import 68

- I**
- IF_ROW_BATCH_NUM column
 - deleting data 102
 - initial value 64
 - merge processing 127
 - processing requirements 19
 - IF_ROW_MERGE_ID column
 - merge processing 127
 - processing requirements 19
 - IF_ROW_STAT column
 - deleting data 102
 - deleting imported rows 155
 - initial value 64
 - merge processing 128
 - processing requirements 19
 - status values 87, 88
 - IF_ROW_STAT_NUM column 20
 - IfbFileName extended parameter 53
 - IGNORE BASE COLUMNS parameter
 - description 40
 - editing for import 67
 - performance 148
 - IGNORE BASE TABLES parameter
 - description 40
 - editing for import 67
 - performance 148
 - IMPORTED import status 88
 - importing data
 - about 57
 - accounts with multiple addresses 156
 - from ACT! 153
 - call lists 78
 - column loading problems 153
 - configuration file, process section
 - edits 66, 67
 - contacts 157
 - existing data, changing 75
 - exported rows, importing 160
 - file attachments 81
 - header and process parameters 68, 72
 - import process, running 86
 - imported rows, viewing list of 87
 - industry codes 79
 - initial batches, importing 74, 75
 - initial data 73
 - initial import guidelines 73
 - inserts, suppressing 75, 76
 - large databases, considerations 84
 - load and run errors 175, 179
 - LOV tables 157, 159
 - multiline addresses, importing 160
 - multiple team members 156
 - private contacts 77
 - process failure, Error 413 169
 - quote problems 156
 - results, verifying 86, 88
 - Solution business component 77
 - unimported rows 155
 - updates, suppressing 76
 - importing data, preparations
 - about 63
 - case of values 65
 - data import order 63
 - file attachment columns, initial values 65
 - initial values for special columns 64
 - importing data, processes
 - insert and update, about 75
 - overview 61, 62, 73
 - user overview 57, 60
 - IN_PROCESS import status 87
 - INCLUDE parameter 40, 46
 - industry codes, importing 79
 - INSERT ROWS parameter
 - description 41
 - editing for import 68
 - inserted data, suppressing 75
 - Integration Manager Log file 75
 - interface table mappings
 - about 21
 - base table mappings without user keys 31, 33
 - column mappings, viewing 27, 28
 - conditions for mapping 22
 - extension table columns 21
 - implicit and explicit mappings 23
 - nonexistent mapping, remedy 21
 - non-target EIM table mapping 22, 23
 - primary child foreign keys 23

- primary child foreign keys, implicit and explicit mapping 23, 24
 - sample view 27
 - viewing 26
 - warning log, creation 146
 - interface tables
 - See also* interface table mappings
 - deleting data 104, 105
 - deleting selected rows 106
 - EIM_type interface tables 18
 - export, preparations 91
 - PR_type columns 23
 - preparing for data deletion 102
 - preparing for merge processing 127
 - from prior releases 18
 - required columns 19
 - statistics, updating parameter 48
 - t_columns 20
 - temporary columns 20
 - updating rows, parameter setting 106
 - viewing deleted rows 124
 - interface tables, import preparations
 - about 63
 - case of values 65
 - data import order 63
 - file attachment columns, initial values 65
 - initial values for special columns 64
- L**
- LANGUAGE parameter 52
 - List of Values (LOV) tables, troubleshooting import 157, 159
 - log entries, SET BASED LOGGING parameter 130
 - LOV Type 154
- M**
- Marketing Response views, updating 131
 - Master Transaction Log 62
 - MAX_NEST_SUBST parameter 52
 - memory requirements 84
 - memory, resource requirements 84
 - merging data
 - aborted merge process, about 131
 - about 125
 - configuration file code, sample 129
 - configuration file, editing 129
 - IF_ROW_MERGE_ID 19
 - interface tables, preparing 127
 - merge process overview 126
 - performance 129
 - results, verifying 133
 - same primary user key, different conflict IDs 163
 - transaction logging parameters 132
 - troubleshooting merge failure 164
 - updating Marketing Response views 131
 - updating rows 131
 - Microsoft Excel spreadsheets, importing 81
 - Microsoft Word documents, importing 81
 - Mobile Web Clients, transaction logging 101
 - multi-org capability 83
- N**
- NET CHANGE parameter
 - description 41
 - editing for import 69
 - non-target table mapping 22, 23
 - null key (virtual) 47
 - NUM_IFTABLE_LOAD_CUTOFF parameter 53
- O**
- ODBC (SQL) error 172
 - ODBC_DATA_SOURCE parameter 52
 - ONLY BASE COLUMNS parameter
 - description 41
 - editing for import 67
 - ONLY BASE TABLES parameter
 - description 40
 - editing for import 66
 - sample code, deleting rows 108
 - operating system error messages 172
 - Oracle database, EIM_IC_CALC interface table 92

- organizations
 - deleting, warning 100
 - exporting data to 97
 - organization mapping 21
- P**
- parameters. *See individual parameter entries*
- PARTIALLY_IMPORTED import status 88
- PASSWORD parameter 40, 44, 52
- performance
 - database optimization configuration parameter 49
 - export considerations 93
 - import parameters for improving 67
 - log entry parameter settings 130
 - merge table limit 129
 - optimizing tips 148, 149
 - recommended number of rows 166
 - set tracing configuration parameter 47
 - tuning suggestions 165
- PR_PROD_LN_ID column, populating 156
- primaries. *See primary child foreign keys*
- primary child foreign keys, implicit and explicit mapping of 23, 24
- private contacts, importing 77
- process batch number 46
- PROCESS parameter 40, 44
- products, deleting data, warning 100
- Q**
- queries
 - account synonyms, configuration parameter 49
 - primary child foreign keys, advantage of using 23
- quotes, troubleshooting 156
- R**
- report errors 179, 180
- ROLLBACK ON ERROR parameter
 - description 40
 - editing for import 72
- rollback segment, defining 48
- ROOT_DIR parameter 52
- ROW_ID column
 - deleting data 102
 - initial value 64
 - merge processing 127
 - processing requirements 19
- rows
 - See also deleting data; exporting data; importing data; merging data*
 - deleting unimported rows 155
 - maximum number in a batch 169
 - parameter for updating 106
- S**
- S_BU base table, exporting names from 97
- S_LST_OF_VAL, processing hung problem 170
- S_OPTY 22
- S_RESITEM base table 77
- Server Manager, defining rollback segment 48
- server space requirements 84
- SESSION SQL parameter 40
- SESSION SQL parameter, description 47
- SET BASED LOGGING parameter
 - editing for import 130
- sfscleanup.exe 82, 181, 184
- SIC codes 79
- Siebel base tables. *See base tables*
- Siebel Enterprise Integration Manager (EIM)
 - field requirements 19
 - process overview 13, 16
- SIEBEL_FILE_DIR parameter 52
- SKIP BU_ID DEFAULT parameter 40, 47
- Solution business component 77
- special columns
 - data deletion values 102
 - delete process 124
 - described 19
 - export considerations 91
 - merge process 133
- spreadsheets, importing 81
- status

- deleting data 124
- exported data, checking status 98
- IF_ROW_STAT column, about 19
- IF_ROW_STAT_NUM column, about 20
- import status, verifying 86, 88
- merge process results 133
- Task Info log, viewing 140
- step-oriented task log, creating 142, 144

T

- TABLE parameter 40, 46
- TABLE_OWNER parameter 52
- TABLEOWNER parameter 40, 45
- Task Info log, viewing 140
- text files, importing 81
- trace flags
 - parameter 1 sample 142, 144
 - parameter 2 sample 144, 145
 - parameter 32 sample 146
 - parameter 4 sample 145
 - parameter 8 sample 146
- TraceFlags extended parameter 53
- transaction logging
 - disk area allocated for 85
 - Mobile Web Client 101
 - parameters for merge 132
- transaction rollback areas 85
- TRANSACTION SQL parameter 41, 47
- TRIM SPACES parameter
 - description 41
 - editing for import 69
- troubleshooting
 - batch failure 168
 - child table export 161
 - configuration file load errors 174, 175
 - Error 405 169
 - Error 413 169
 - Error 999 169
 - error flags, sample 140
 - exit status errors 172, 173
 - import process load/run errors 175, 179
 - lack of memory 171
 - mandatory columns, determining 153

- multiple row failure 167
- report errors 179, 180
- S_LST_OF_VAL base table, processing
 - hung 170
- SQL trace flags 142
- Task Info Log, viewing 140
- trace flag parameter 1, sample
 - output 142, 144
- trace flag parameter 2, sample
 - output 144, 145
- trace flag parameter 32, sample
 - output 146
- trace flag parameter 4, sample
 - output 145
- trace flag parameter 8, sample
 - output 146
- unsuccessful merge 164
- TYPE parameter 41, 46

U

- UPDATE ROWS parameter
 - description 41, 42
 - in merge processing 130
 - sample 106
- UPDATE STATISTICS parameter 41, 48
- updated data, suppressing 76
- USE INDEX HINTS parameter 41, 49
- USE SYNONYMS parameter 41
- user key override log, creating 145
- user keys
 - deleting data, sample code 108
 - interface to base table mappings 31, 33
- user parameter substitution log,
 - creating 144, 145
- USERNAME parameter 40, 44
- USING SYNONYMS parameter 49

V

- virtual null key 47
- Visual Basic code 154

X

- xxx_BU columns 21