SIEBEL 7
eBusiness

SIEBEL CONNECTOR FOR PEOPLESOFT

VERSION 7.0, REV. H

12-BCK987

JUNE 2002

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

Contents

Chapter 3. Database Adapter

Chapter 4. Database Wizard

Chapter 5. Prebuilt Integration Objects

Appendix A. Prebuilt Integration Object DTDs

Index

Introduction

This guide explains how to install, configure, maintain, and use the Siebel Connector for PeopleSoft. Because these activities involve both Siebel components and PeopleSoft components, members of both teams should use this guide. The Siebel administrator, Siebel business analyst, Siebel administrator, various end users, PeopleSoft administrator, and developers who work with the PeopleSoft Connector should read this guide. These specialists are responsible for the Connector activities listed in the next section.

How This Guide Is Organized

This guide is organized to provide the information needed by specific groups of readers. Chapters focus upon Connector installation, set up and configuration, administration, end user activities, and developer procedures. Appendix A, "Prebuilt Integration Object DTDs," provides a listing of the XML DTDs included in the Connector.

The following table indicates the range of Connector-related information provided in the guide. The left column identifies the reader while the column on the right lists activities described in the guide.

Siebel System Administrator	The Siebel System Administrator should refer to this guide for Connector-related instructions for: ■ Installation activities (see Chapter 2, "Installing the Siebel Connector for PeopleSoft," including: ■ installing and configuring the Siebel Server ■ creating the ODBC data source
Siebel Business Analyst	The Siebel Business Analyst should work with the Siebel Administrator to define eAI value maps.
Developers	Developers working with the PeopleSoft Connector team should refer to this guide for Connector-related instructions for: ■ Developer Activities including: ■ working with integration objects ■ running the Database Wizard

Revision History

Siebel Connector for PeopleSoft, Version 7.0, Rev. H

1 Overview

This chapter describes the Siebel Connector for PeopleSoft architecture.

Siebel Connector for PeopleSoft

The Siebel Connector for PeopleSoft allows data integration between Siebel applications and PeopleSoft. It helps you to serve your customers by synchronizing critical information essential to serving customers and by automating business processes that cross application boundaries. The Siebel Connector for PeopleSoft also helps organizations manage the total cost of deploying Siebel eBusiness Applications by facilitating data exchange definition, data conversion, cross-application business process modeling, and data communication through the standard Siebel eBusiness Applications and PeopleSoft supported interfaces.

Connector Features

The Siebel Connector for PeopleSoft includes the following features:

- Integration Infrastructure
- Prebuilt Integration Objects and Workflow Processes
- Mobile User Support

These features and associated functions are discussed below.

Integration Infrastructure

Siebel Connector for PeopleSoft provides tools for defining and maintaining integration points. In addition to design-time tools, the Siebel Connector for PeopleSoft provides run-time components that you can use to manage data exchanges between the two applications.

Prebuilt Integration Objects and Workflows

The Siebel Connector for PeopleSoft provides prebuilt integration objects and workflows.

- **Integration Objects.** These objects represent the common data exchanged between Siebel applications and PeopleSoft applications. External integration objects (PeopleSoft Integration Objects) model the data represented by PeopleSoft. They specify the data structure of the PeopleSoft interfaces. For example, a PeopleSoft customer integration object represents the table structure in PeopleSoft that is used to import customer information into PeopleSoft applications. Internal integration objects (Siebel Integration Objects) define the data structure used by the Siebel application. For instance, a Siebel Account integration object defines the data structures of the Account business components used in a particular integration flow.

- **Siebel Integration Workflows.** These workflows determine the sequence of processing steps involved in data exchanges between the Siebel application and PeopleSoft. You can modify and extend workflows with the Siebel Workflow Designer. This allows you to add business logic that closely models your business processes.

Mobile User Support

The Siebel Connector for PeopleSoft allows Siebel eBusiness Applications users to access and update PeopleSoft while they are disconnected from the network. It uses Siebel Remote to synchronize information stored on the Siebel Server with local databases maintained by the Siebel Mobile Web Client, and maintain a queue of transaction data destined for PeopleSoft.

Incoming transactions from PeopleSoft to Siebel eBusiness Applications are routed to mobile users according to visibility rules defined in the Siebel Repository for Siebel Remote. Outgoing transactions, such as account updates from mobile users, are queued in their local databases. When these mobile users synchronize with the Siebel Server, their transactions are placed in the server queue and subsequently routed to PeopleSoft through the interfaces described in this guide.

Connector Architecture

The Siebel Connector for PeopleSoft includes the following design and run-time tools and components:

Table 1. Siebel Connector for PeopleSoft Components

Function	Design Time Component	Design Time Tool	Run Time Component
Data Definition	Integration Objects	■ Integration Object Editor ■ Integration Object Wizards	
Data Transformation	Data Transformation Maps	■ Data Mapper ■ Business Service Editor	Data Transformation Engine (DTE)
Business Process Integration	Integration Workflow	Workflow Process Designer	Business Integration Manager (BIM)
Data Transport			■ Siebel Adapter ■ Database Adapter ■ HTTP Adapter ■ HTTP Receiver ■ XML Converter

These Connector components are discussed in the following pages.

Integration Objects

Integration Objects are data containers that hold the information that is being exchanged between Siebel eBusiness Applications and PeopleSoft. When instantiated during run-time, they are used to hold the content of the data being sent from one application to the other. Two integration objects are typically defined for each data flow—an internal data object that represents Siebel eBusiness Applications data and an integration object that represents the corresponding external data object in PeopleSoft.

For example, to import employee data from PeopleSoft into Siebel eBusiness Applications, you must define an external integration object that represents the PeopleSoft's Employee data and another internal integration object that represents the corresponding Siebel Employee business component. Several predefined Integration Objects are provided by the connector. These objects can be used in defining integration points.

Integration Object Editor

Integration Objects can be defined manually using the Integration Object Editor in Siebel Tools. The procedure for defining an integration object is similar to the procedure for creating a Siebel Business Object.

- See *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II* for instructions on defining integration objects manually.

Integration Object Wizards

You can use Wizards within Siebel Tools to create integration objects. These wizards can be used to ascertain the metadata definitions of applications or data sources including customizations to data model, and generate appropriate integration objects. You can use the Integration Object Editor to further edit the generated objects.

The following Integration Object Wizards are provided for integrating Siebel eBusiness Applications with PeopleSoft:

Siebel Wizard

This wizard is used to examine metadata definitions of Siebel eBusiness Applications and generate internal integration objects that are based on Siebel Business Objects.

- See *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II* for instructions on using this wizard.

XML DTD Wizard

The XML DTD Wizard is used to generate external integration objects based on PeopleSoft's XML DTD.

- See *XML Integration Reference* for instructions on using this wizard.

Database Wizard

The Database Wizard can be used to examine the data dictionary of the database that PeopleSoft uses to store data and extract the data definition. These data definitions can be used to create external integration objects to represent PeopleSoft data. The Database Wizard supports Oracle, Microsoft SQL Server, and IBM DB2 databases.

- See Chapter 3, "Database Adapter," for instructions on using this wizard.

Data Transformation Maps and Data Transformation Engine

After the internal and external integration objects are created, a Data Transformation Map is used to transform data contained in the two objects. During run-time, a data transformation map is passed to the Siebel Data Transformation Engine (DTE) along with an internal and an external integration object. The map defines the relationships of data elements among the two integration objects. The DTE transforms the data following the map definitions.

Depending on the direction of data transfer, the internal and external integration objects may serve as either the DTE input or output. For example, when information flows inbound from PeopleSoft to Siebel, the external integration object is the DTE input and the output goes to the internal integration object. The arrangement is reversed for an outbound data flow.

You may define Data Transformation Map by using the Business Service Editor to write an eScript or by using the Siebel Data Mapper.

Business Service Editor

You can write a Siebel Business Service in Siebel eScript to process the integration object input to the Siebel Data Transformation Engine (DTE) and invoke DTE data transformation functions that put the transformed data into the DTE output.

- See *Siebel Tools Reference* for more about the Business Service Editor.

Data Mapper

The Siebel Data Mapper is used to define data transformation maps declaratively. Instead of writing eScripts to traverse through integration objects and transform the data, the Data Mapper allows you to define the relationships between the components and fields of the internal and external integration objects. The Siebel Data Transformation Engine (DTE) interprets the relationships and performs the data transformation on your behalf. You may access the Data Mapper through the Siebel Web Client.

- See *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV* for instructions on using the Siebel Data Mapper.

Integration Workflow

An Integration Workflow, a type of Siebel Workflow Process, defines the processing steps for data that will be exchanged between Siebel eBusiness Applications and PeopleSoft. For example, an integration workflow for outbound account data from Siebel eBusiness Applications to PeopleSoft may contain steps that invoke the Siebel Adapter to query the account object from the Siebel Object Manager, use the DTE to transform the data, convert the data into XML, and call the HTTP Adapter to send information to PeopleSoft through its interfaces.

- See *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV* for instructions on defining Integration Workflow.

Workflow Process Designer

Integration Workflows are created and maintained with the Siebel Workflow Process Designer, a graphical tool accessed through the Siebel Web Client.

Workflow Process Manager

The Siebel Workflow Process Manager is the run-time component that executes integration workflows. It can handle both interactive and batch data exchanges.

- See *Siebel Workflow Administration Guide* for instructions on using the Workflow Process Manager and on using the Designer.

Siebel Adapter

Siebel Adapter moves data into and out of the Siebel Object Manager. When outbound data flows from Siebel eBusiness Applications to PeopleSoft, Siebel Adapter is the first step of the workflow. It queries the Object Manager for data and puts it into an instantiated integration object for additional processing. When inbound data flows from PeopleSoft to a Siebel application, Siebel Adapter is the last step of the workflow. It inserts or updates the incoming data into the Siebel Object Manager.

- See *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II* for instructions for using the Siebel Adapter.

Database Adapter

The Database Adapter allows data exchanges between Siebel eBusiness Applications and PeopleSoft database tables. During an outbound data flow, the Database Adapter writes data to PeopleSoft's interface tables. During an inbound data flow, the Database Adapter can read data from PeopleSoft's interface tables or base tables and passes the data into Siebel eBusiness Applications. The Database Adapter can be used for both PeopleSoft 7.5 and 8.

- See Chapter 3, "Database Adapter," for instructions on using this Adapter.

HTTP Adapter and HTTP Receiver

The HTTP Adapter and HTTP Receiver allow Siebel eBusiness Applications to send and receive XML messages with PeopleSoft using HTTP as the communication protocol. The HTTP Adapter and HTTP Receiver support integrating Siebel eBusiness Applications with PeopleSoft 8.

- See *Transports and Interfaces: Siebel eBusiness Application Integration Volume III* for instructions for using the adapter and receiver.

XML Converter

XML Converter converts data between Siebel-native representation to XML. It supports integration between Siebel eBusiness Applications with PeopleSoft 8 by exchanging XML messages. During an outbound dataflow, the XML Converter converts data from Siebel-native format to XML before the data is sent to PeopleSoft using the HTTP Adapter. During an inbound flow, the converter converts XML-encoded data from PeopleSoft 8 back to Siebel-native representation.

Run-Time Architecture

The run-time architecture of the Siebel Connector for PeopleSoft involves outbound and inbound data processes.

Outbound Data

For outbound data sent from Siebel eBusiness Applications to PeopleSoft:

1. A user action in a Siebel Applet, a Siebel Workflow Event, or a Business Integration Manager batch run invokes an Integration Workflow to execute the data transfer.
2. The workflow invokes the Siebel Adapter to query the updated business object through Siebel Object Manager and put the data into an instantiated internal integration object.
3. The integration object is passed to the Data Transformation Engine (DTE). The DTE transforms the object content from Siebel eBusiness Applications representation into an equivalent PeopleSoft representation and puts it into an instantiated external integration object.
4. The external integration object is passed to the Database Adapter or to the XML Converter and then to the HTTP Adapter, which sends the information contained in the object to PeopleSoft through its interfaces.

Inbound Data

For inbound data sent from PeopleSoft to Siebel eBusiness Applications:

1. Database Adapter reads data from PeopleSoft and places it in an instantiated external integration object.
2. XML-encoded data is sent from PeopleSoft 8 to Siebel HTTP Receiver and the receiver places the data into an instantiated external integration object. The object is then converted from XML representation back into a Siebel-native representation through the XML Converter.
3. The external integration object is passed to the Data Transformation Engine. The DTE transforms the object content from PeopleSoft's representation into an equivalent Siebel representation and puts it into an instantiated internal integration object.

4 The internal integration object is passed to the Siebel Adapter. The Adapter "upserts" (inserts or updates) the data to the Siebel Object Manager and puts it into the Siebel Database.

These processes are represented in Figure 1:

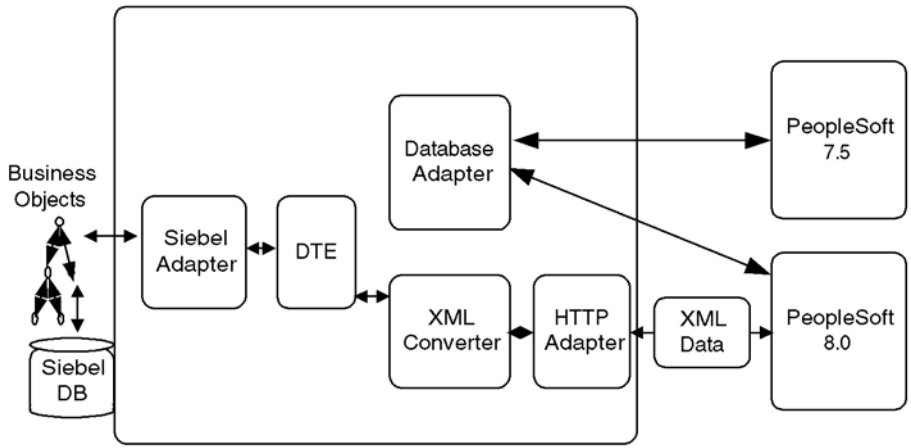


Figure 1. Inbound Data Processes

Installing the Siebel Connector for PeopleSoft 2

This chapter describes how to install Siebel Connector for PeopleSoft.

Overview

This overview summarizes all the main installation issues. For more details, refer to the later sections of this chapter.

All necessary components of the Siebel Connector for PeopleSoft are automatically installed during the installation of Siebel software. The integration objects and the workflow processes that are part of this Connector will be installed during the regular installation of Siebel software.

If your Siebel installation and the PeopleSoft installation (or the external database that you want to integrate Siebel with) are running on the same RDBMS platform, all required driver software is automatically installed. However, if you are using different platforms, you must install and register appropriate database drivers and set up required ODBC data sources.

Whether your installation is heterogeneous or not, you must create ODBC data sources. This will facilitate the connections with the external database.

There is no separate installer to run for the Siebel Connector for PeopleSoft. All of the components of the connector are copied onto the appropriate system as part of Siebel Server, Siebel Tools, and Siebel Web Client install. You may have to follow different steps when you run the Siebel Software Configuration Utility, Siebel Tools, and Siebel Web Client installer.

NOTE: For more information about the Siebel Server installation, see *Siebel Server Installation Guide for Microsoft Windows*.

ODBC Configuration

If you plan to use the Database Adapter and Database Wizard in defining integration points between Siebel eBusiness Applications and PeopleSoft, you need to install and register ODBC driver, and set up the proper ODBC data source in order to obtain the necessary connection. The procedure is different depending on the operating system environment that you run your Siebel Server on, and the database system that you use for storing PeopleSoft data.

Table 2 summarizes the procedures.

Table 2. ODBC Driver Installation and Registration

Installation	PeopleSoft Running on Oracle Database	PeopleSoft Running on IBM DB2 Database	PeopleSoft Running on Microsoft SQL Server
Siebel Server (Microsoft Windows)	Run the Siebel Software Configuration Utility after you install the Siebel Server package	Install ODBC Driver provided by the DB2 Client package separately	Install ODBC Driver provided by SQL Server's MDAC package separately
Siebel Server (Solaris)	Run the Siebel Software Configuration Utility after you install the Siebel Server package	Follow instructions on Siebel Server Installation Guide for UNIX for installing and registering DB2 ODBC driver	Not Supported
Siebel Server (AIX)	Run the Siebel Software Configuration Utility after you install the Siebel Server package	Follow instructions on Siebel Server Installation Guide for UNIX for installing and registering DB2 ODBC driver	Not Supported
Siebel Tools	Select Custom Install when you run the Siebel Tools installer	Install ODBC Driver provided by the DB2 Client package separately	Install ODBC Driver provided by SQL Server's MDAC package separately
Siebel Mobile Web Client / Siebel Dedicated Web Client	Select Custom Install when you run the Siebel Web Client installer.	Install ODBC driver provided by the DB2 Client package separately.	Install ODBC driver provided by SQL Server's MDAC package separately.

NOTE: You still have to use the operation system's native utility to set up the ODBC data source after you install and register the driver.

Systems

ODBC Driver setup is performed on the following environments:

- Siebel Server
- Siebel Dedicated Web Client or Siebel Mobile Web Client (if you plan to deploy these types of clients)
- Siebel Tools

Siebel Clients and Siebel Tools

Siebel Web Clients and Siebel Tools operate under Windows 98, Windows NT, and Windows 2000 operating systems. The following sections explain how to install and register ODBC drivers in order for the Database Wizard and Database Adapter to access PeopleSoft data stored in Oracle, IBM DB2 and Microsoft SQL Server databases.

PeopleSoft Running on an Oracle Database

The ODBC driver from Merant required for an Oracle database is included in the Siebel Tools and Web Client packages. If you are running PeopleSoft on an Oracle database, follow the directions summarized as follows.

To install the driver

- Run the client install in the custom install mode to select ODBC Driver for External Oracle DB to copy the driver onto the system and register it with Microsoft Windows.

PeopleSoft Running on an IBM DB2 Database

The IBM ODBC driver from IBM is not bundled in as part of Siebel Tools and Web Client packages. If your are running PeopleSoft on a DB2 database, following the directions summarized below.

To install the driver

- Install DB2 ODBC driver from the DB2 CD-ROM. The DB2 installation process registers the ODBC driver with Windows.

PeopleSoft Running on a Microsoft SQL Server Database

The Microsoft ODBC driver is not bundled in as part of Siebel Tools and Web Client packages. If you are running PeopleSoft on an SQL Server database, you may use the ODBC driver that is included in the Third-Party software directory of the Siebel installation CD-ROM. Follow the directions summarized below.

To install the driver

- Install the Microsoft MDAC driver from the MDAC package. The MDAC installation registers the ODBC driver with Microsoft Windows.

Siebel Server on Windows NT and Windows 2000

The following sections explain how to install and register ODBC drivers for Siebel Server running on Windows NT and Windows 2000 operating systems.

PeopleSoft Running with an Oracle Database

The Merant ODBC driver is bundled in as part of Server package. If you are running PeopleSoft with an Oracle database, follow the directions summarized below.

To register the driver

1. Start the Siebel Software Configuration Utility. The utility is automatically launched at the conclusion of the Siebel Server install. You may also start it manually by selecting Microsoft Windows Start Menu > Programs > Siebel Enterprise Server > Configure Siebel Server.
2. When the dialog box “Configure EAI Connectors: Register External Oracle DB ODBC Driver” appears, check Yes under the “Register External Oracle DB ODBC Driver” prompt.

PeopleSoft Running with an IBM DB2 Database

The IBM ODBC driver from IBM is not bundled in as part of Siebel Server package. If your are running PeopleSoft with a DB2 database, following the directions summarized below.

To install the driver

- Install DB2 ODBC driver from the DB2 CD-ROM. The DB2 installation process registers the ODBC driver with Windows.

PeopleSoft Running on a Microsoft SQL Server Database

The Microsoft ODBC driver is not bundled in as part of Siebel Tools and Web Client packages. If you are running PeopleSoft on an SQL Server database, you may use the ODBC driver that is included in the Third-Party software directory of the Siebel installation CD-ROM. Follow the directions summarized below.

To install the driver

- Install the Microsoft MDAC driver from the MDAC package. The MDAC installation registers the ODBC driver with Microsoft Windows.

Siebel Server on Solaris

The following sections explain how to install and register ODBC drivers for Siebel Server running on Sun Solaris operating environment.

NOTE: If you run your PeopleSoft application on a Microsoft SQL Server database, you cannot use the PeopleSoft Connector on a Siebel Server running on the Solaris operating environment. The appropriate driver for connecting to Microsoft SQL Server from Solaris is not available at this time.

PeopleSoft Running with an Oracle Database

The Merant ODBC driver is bundled in as part of Server package. If you are running PeopleSoft with an Oracle database, follow the directions summarized below.

To register the driver

- The Server installation process automatically copies the ODBC driver. Use the Server configuration tool to edit the configuration files and set up an environment variable to register the ODBC driver with Solaris. Refer to *Siebel Server Installation Guide for UNIX* for more instructions.

Siebel Server on AIX

The following sections explain how to install and register ODBC drivers for Siebel Server running on IBM AIX operating system.

NOTE: If you run your PeopleSoft application on a Microsoft SQL Server database, you cannot use the PeopleSoft Connector on a Siebel Server running on the IBM AIX. The appropriate driver for connecting to Microsoft SQL Server from AIX is not available at this time.

PeopleSoft Running with an Oracle Database

The Merant ODBC driver is bundled in as part of Server package. If you are running PeopleSoft with an Oracle database, follow the directions summarized below.

To register the driver

- The Server installation process automatically copies the ODBC driver. Use the Server configuration tool to edit the configuration files and set up an environment manually variable to register the ODBC driver with AIX. Refer to *Siebel Server Installation Guide for UNIX* for more instructions.

Installing ODBC Driver for Siebel Server for DB2

This section explains how to install ODBC drivers for external DB2 databases.

Installing the DB2 Client

Before proceeding with the following sections, make sure that the DB2 client (and the connectivity) is installed on your system. For details, please obtain the services of a qualified DB2 database administrator and refer to the DB2 documentation.

Installing the ODBC Driver

If your external database (for example, PeopleSoft database) is on a DB2 RDBMS platform, there are no special installation requirements for the ODBC Driver for DB2. You should see the appropriate driver (IBM DB2 ODBC Driver) installed on your system once you installed the database client.

NOTE: For the supported versions of this driver, please refer to *Siebel System Requirements and Supported Platforms* on the *Siebel Bookshelf*.

Creating ODBC Data Source for a DB2 Database

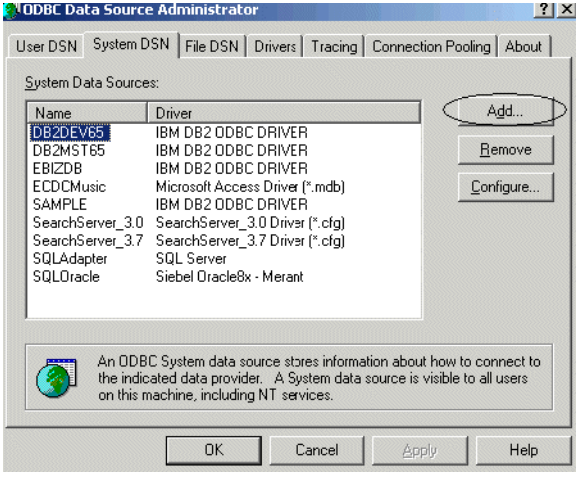
This section explains how to create ODBC data sources for external DB2 databases.

Creating an ODBC Data Source

This section explains how to create the ODBC data source.

To create an ODBC data source

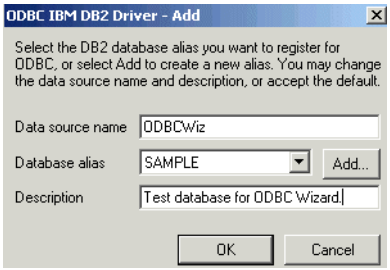
1. Select Data Sources (ODBC) menu item from the Windows Start button (Start > Settings > Control Panel > Administrative Tools > Data Sources [ODBC]) Windows 2000.
2. On Windows NT, launch the ODBC Administrator from the Control Panel by selecting ODBC Data Source.
3. Click Add in the ODBC Data Source Administrator.



4. When the Create New Data Source screen appears, select IBM DB2 ODBC DRIVER as the driver for the data source and click Finish.

5 When the ODBC IBM DB2 Driver - Add screen appears, enter a name for the ODBC data source (in this example, ODBCWiz). This data source name will be used in both tools.cfg and Siebel Web Client configuration file (such as siebel.cfg and uagent.cfg) to connect to the external database.

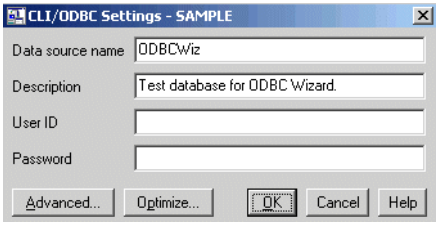
NOTE: When you use the ODBC data source wizard, you will supply the name of the server that hosts this DB2 database.



6 Click OK.

7 When the ODBC Data Source Administration screen appears, click Configure.

8 When the Connect to DB2 Database screen appears, enter the User ID and Password, then click OK. The CLI/ODBC Settings dialog box appears. For example:



9 If you accept the defaults, click OK. However, if you decide to click Advanced, a new screen will allow you to make site-specific changes.

Installing the ODBC Driver for an Oracle Database

This section explains how to install the ODBC driver for an external Oracle database.

Installing the Database Client

Check to be certain that the Oracle client (and the connectivity) is installed on your system. For details, please obtain the services of a qualified Oracle database administrator and refer to the Oracle documentation.

Installing the ODBC Driver

At first, you install the appropriate ODBC driver. This section explains how to install the driver.

The ODBC driver used by Siebel to connect with the external Oracle databases is installed during installation of Siebel software.

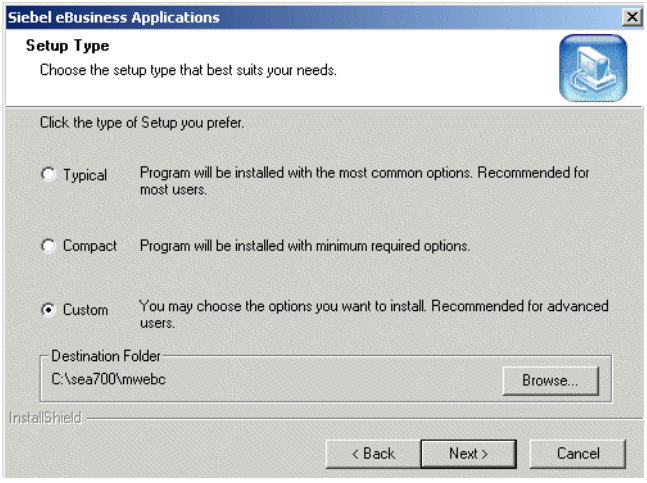
NOTE: For information about Siebel software installation, see *Siebel Server Installation Guide for Microsoft Windows* or *Siebel Web Client Administration Guide* on the Siebel Bookshelf.

From Siebel Client Installation

You can install the ODBC driver from the Siebel Web Client installation.

To install the ODBC driver from Siebel Client installation

1 Select Data Sources (ODBC) menu item from the Windows Start button (Start > Settings > Control Panel > Administrative Tools > Data Sources (ODBC)) on Windows 2000 operating system. On Windows NT, launch the ODBC Administrator from the Control Panel by selecting ODBC Data Source.



If the Custom setup option is selected, the Select Components dialog appears.

2 Select ODBC Driver for External Oracle Database.

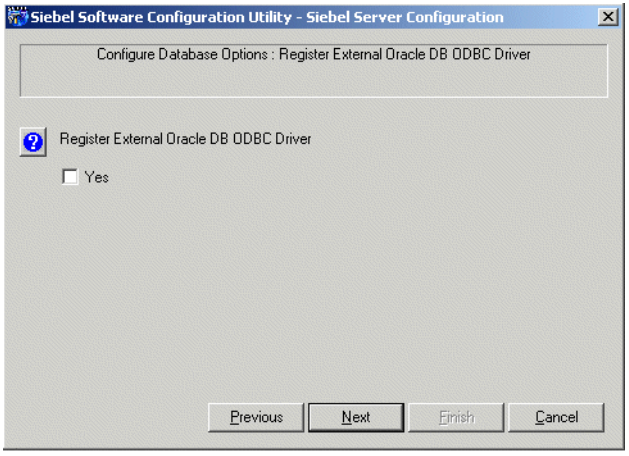
This action installs and registers the ODBC driver used by the Siebel application to connect to external Oracle databases.

3 When the ODBC IBM DB2 Driver – Add screen appears, enter a name for the ODBC data source (in this example, ODBCWiz). This data source name will be used in both tools.cfg and Siebel Web Client configuration file (such as siebel.cfg or uagent.cfg) to connect to the external database.

From Siebel Server Installation

To install the ODBC driver from Siebel Server installation

- When the screen Configure Database Options: Register External Oracle DB ODBC Driver appears, as shown in the following figure, and prompts you to select the driver, click Yes.



From Siebel Tools Installation

You can select a custom set up from the Siebel Tools installation.

To install the ODBC driver from Siebel Tools installation

1. When the Siebel eBusiness Applications dialog appears, select Custom in the Setup Type dialog. This action allows you to customize the installation process.
2. If you chose the custom setup option, the Select Components dialog appears. In this case, select ODBC Driver for External Oracle. This action installs and registers the ODBC driver used by the Siebel application to connect to the external Oracle databases.

Verifying the Driver Installation

After you finish installing the Siebel software, you will see an Oracle ODBC driver manufactured by MERANT, Inc. under the Drivers tab in the ODBC Data Source Administrator's screen, shown in Figure 2. This driver will be used (not the default ODBC driver from Oracle) to create the ODBC data sources for the external Oracle databases.

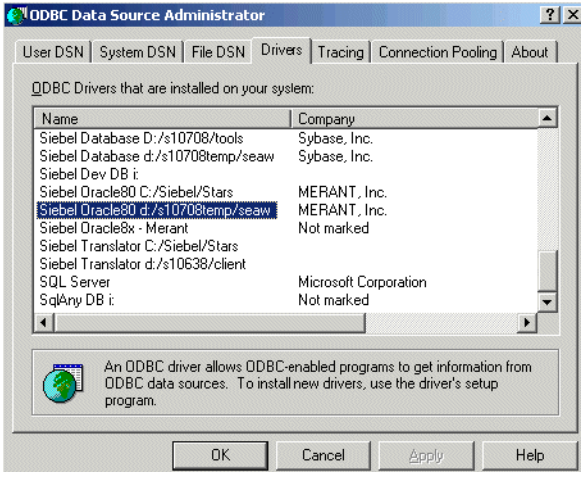


Figure 2. Verifying the ODBC Driver Installation

Creating an ODBC Data Source for an Oracle Database

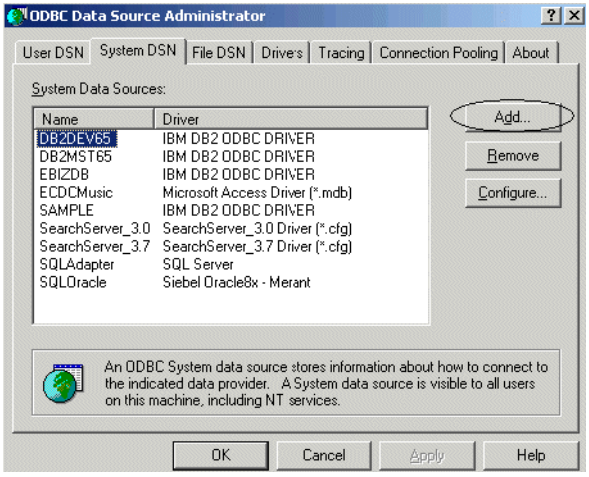
This section explains how to create an ODBC data source for an external Oracle database.

Creating an ODBC Data Source

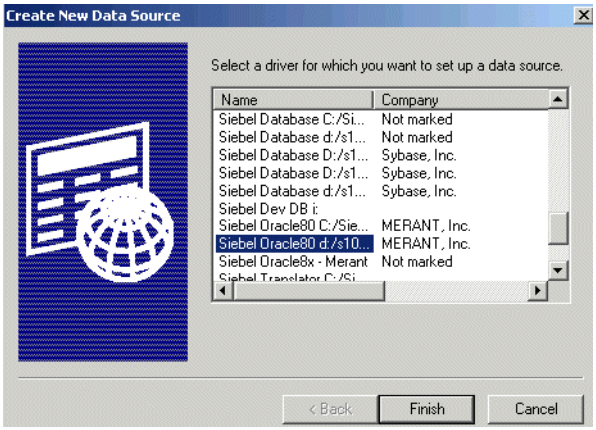
Next, you must create an ODBC data source.

To create an ODBC data source

1. From the Windows 2000 Start button, select Settings > Control Panel > Administration Tools > Data Sources (ODBC).
2. When the ODBC Data Source Administrator dialog appears, under the System DSN tab, select the driver and click Add.

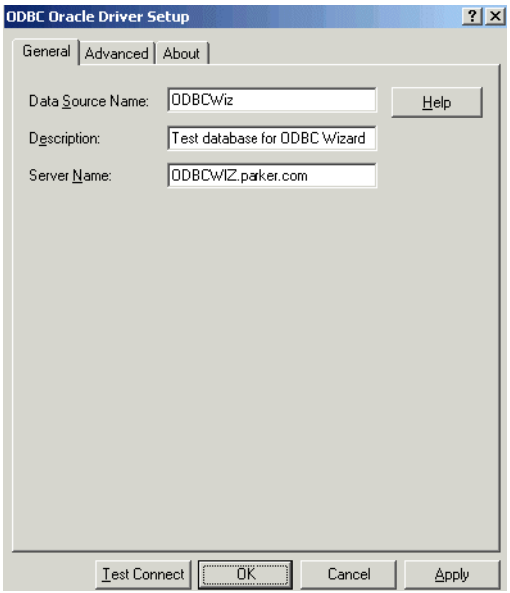


3 When the Create New Data Source dialog appears, select the Oracle driver manufactured by MERANT, Inc. This driver was installed during the installation of Siebel software, if the appropriate options were selected. Then, click Finish.

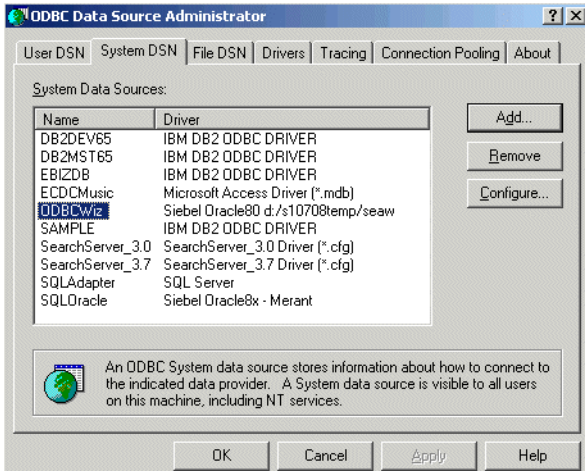


4 When the Oracle Driver Setup dialog appears, fill in the ODBC Data Source Name, Description, and Server Name for the server that hosts the Oracle database.

5 Click Test Connect to test the database connectivity (you must supply the login user ID and password) or click OK to close the window.



6 When the ODBC Data Source Administrator appears, the data source that you created appears under the System DSN tab. If you want to change the data source, click Configure. Otherwise, click OK.


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
System Data Sources:
Name
Driver
DB2DEV65 IBM DB2 ODBC DRIVER
DB2MST65 IBM DB2 ODBC DRIVER
EBIZDB IBM DB2 ODBC DRIVER
ECDCMusic Microsoft Access Driver (*.mdb)
ODBCWiz Siebel Oracle80 d:/s10708temp/seaw
SAMPLE IBM DB2 ODBC DRIVER
SearchServer_3.0 SearchServer_3.0 Driver (*.cfg)
SearchServer_3.7 SearchServer_3.7 Driver (*.cfg)
SQLAdapter SQL Server
SQLOracle Siebel Oracle8x - Merant
Add...
Remove
Configure...
An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.
OK
Cancel
Apply
Help

Installing ODBC Driver for MS SQL Server

This section explains how to install the ODBC driver external Microsoft SQL Server database.

Installing the Database Client

Be sure that the Microsoft SQL Server client (and connectivity) is installed on your system. For details, please obtain the services of a qualified Microsoft SQL Server database administrator and refer to the Microsoft SQL Server documentation.

Installing the ODBC Driver

If your external database (for example, PeopleSoft database) is on an MS SQL Server RDBMS platform, there are no special installation requirements for the ODBC Driver for MS SQL Server. The appropriate driver (SQL Server, shown in Figure 3) is installed on your system as the SQL Server client is being installed. For the supported versions of this driver, please refer to the Supported Platforms guide in the Siebel Bookshelf.

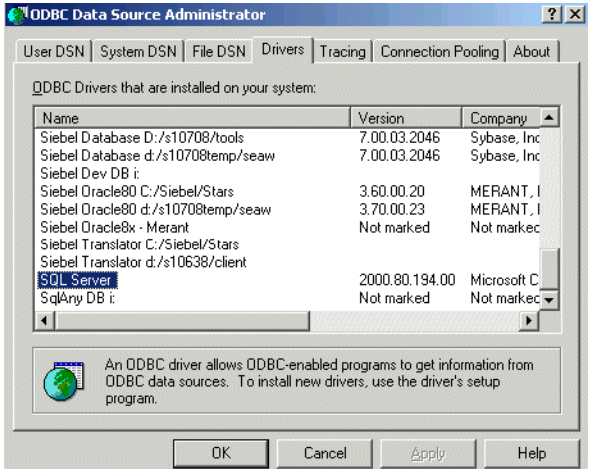


Figure 3. SQL Server Driver

Creating the ODBC Data Source for SQL Server Database

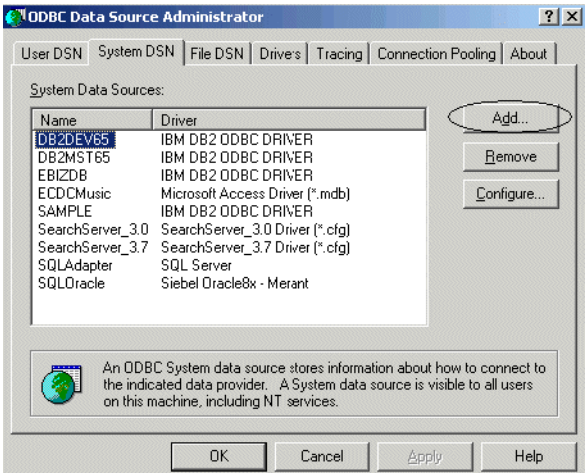
This section explains how to create the ODBC data source for an external Microsoft SQL Server database.

Creating an ODBC Data Source

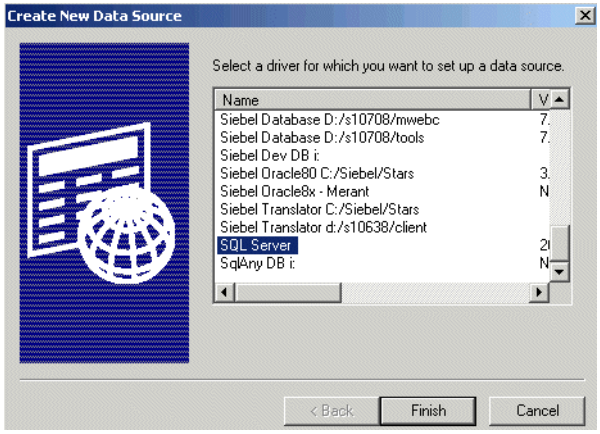
Next, you must create an ODBC data source.

To create the ODBC data source

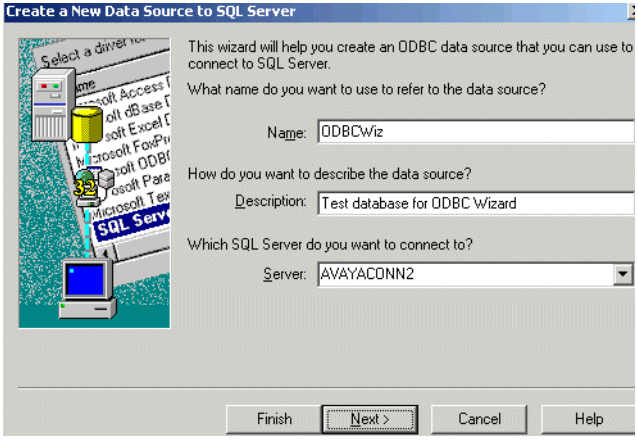
1. Select Data Sources (ODBC) menu item from the Windows Start button (Start > Settings > Control Panel > Administrative Tools > Data Sources (ODBC)) on Windows 2000 operating system. On Windows NT, launch the ODBC Administrator from the Control Panel by selecting ODBC Data Source.
2. When the ODBC Data Source Administrator dialog appears, under the System DSN tab, select the driver and click Add.



3 When the Create New Data Source dialog appears, select SQL Server as the driver for the data source and click Finish.

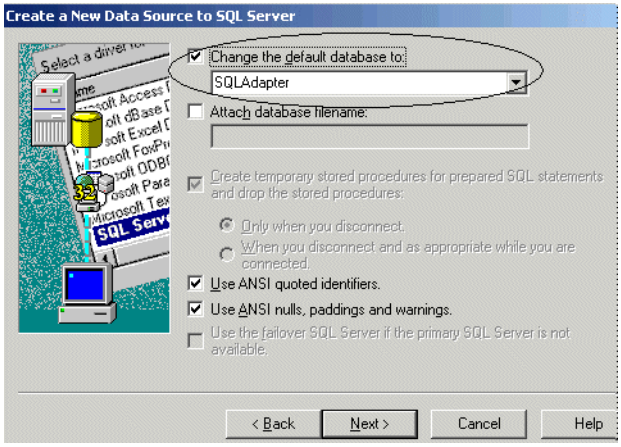


4 When the Create a New Data Source to SQL Server dialog appears, supply the data source Name, Description, and the Server that hosts the MS SQL Server database. Then, click Next.



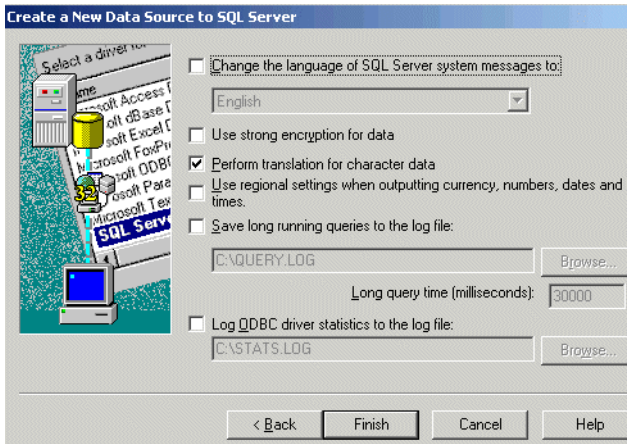
5 When the dialog refreshes, fill in the Login ID and Password to connect to the external database. These values will be used later on in the tools.cfg and Siebel Web Client configuration (siebel.cfg and uagent.cfg).

6 When the dialog refreshes, select the appropriate database on the external system.

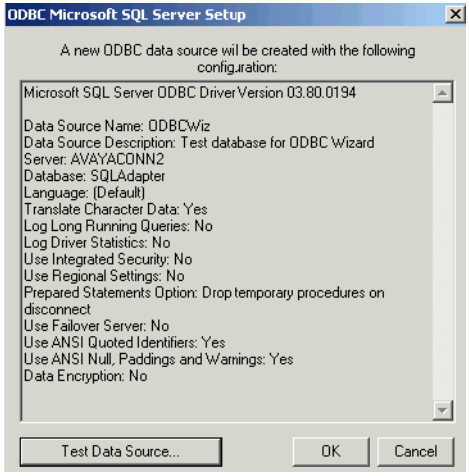


7 Accept the defaults and click Next.

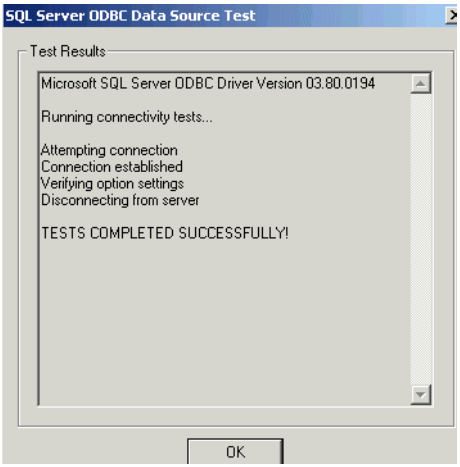
8 When the dialog refreshes again, click Finish.



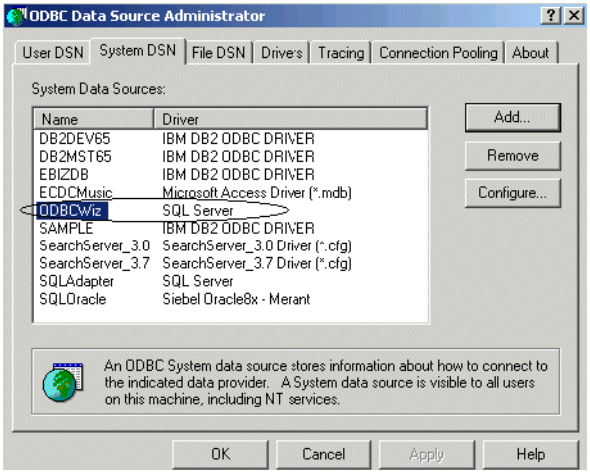
9 When the ODBC Microsoft SQL Sever Setup dialog appears, verify the parameters and click Test Data Source to test the connectivity to the data source.



10 If the test is successful, a message appears. Click OK.



11 Return to the main ODBC Data Source Administrator window, your new data source is included in the list.



Database Adapter 3

The Database Adapter is a Siebel business service that can be used to exchange information with PeopleSoft through its database interface. It accesses PeopleSoft's database interface by dynamically generating an SQL statement (or set of SQL statements), based upon data in an Integration Object and the invoked method which indicates whether the SQL statement is an INSERT, UPDATE, DELETE, or SELECT. The Adapter executes the SQL statements against the external PeopleSoft data store. For INSERT, UPDATE, DELETE, a count or error is returned. In the case of SELECT, the retrieved data values are returned and inserted into a second Integration Object.

Like other Siebel business services, the Database Adapter simplifies moving and converting data between Siebel eBusiness Applications and external applications.

Similar to the Siebel Integration Manager, the Database Adapter is used to move data to or from an external database. The Adapter dynamically creates an SQL statement (or set of SQL statements), based upon data in an Integration Object and the invoked method, which indicates whether the SQL statement is an INSERT, UPDATE, DELETE, or SELECT. The Database Adapter executes the SQL statements against the external PeopleSoft data store. For INSERT, UPDATE, DELETE, a count or error is returned. In the case of SELECT, the retrieved data values are returned and inserted into a second Integration Object.

Configuration Parameters

The parameters listed below are used by the Database Adapter to connect to the PeopleSoft database. You may define them when you invoke the adapter as a step in the workflow. You may also set them in the configuration file of Siebel Web Client (such as siebel.cfg or uagent.cfg) for use with the Siebel Workflow Simulator. Finally, you may define them in the configuration of Business Integration Manager or Workflow Process Manager server components. The settings that you make in workflow will override the other configurations.

Table 3. Configuration Parameters

Parameter	Display Name	Reference
ExtDBODBCDataSource	External DB Data Source	An external data source.
ExtDBPassword	External DB Password	Password used to access the external data source.
ExtDBTableOwner	External DB TableOwner	Table owner for the external data source.
ExtDBUserName	External DB UserName	Username for the external data source.

Methods and Arguments

The Database Adapter business service incorporates methods that are used to perform various tasks (queries, data inserts, updates, and so on). Each method can have one or more arguments that are used to further define the method's action. An argument typically consists of data or an object that the method processes.

Five methods may be used with the DB Adapter:

- Query
- Delete
- Upsert
- Synchronize
- Execute

These are described in the next sections. In each case, an introductory paragraph is followed by a table showing the display name, optional status, input or output type, data type, and sometimes a comment for each method parameter.

Data Type

The three types of databases that are used with the Connector support different sets of data types. These differences are indicated in Table 4.

Table 4. Databases and Data Types

Database	Supported Data Types
IBM DB2	varchar, char, bigint, integer, smallint, decimal, real, timestamp, data, and time
Microsoft SQL Server	varchar, char, bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, and datetime
Oracle	varchar2, char, number, and date

Query

The Database Adapter uses the Query method to query data from an external database based on an SQL integration object and returns the corresponding integration object instances.

Query takes a QBE (Query By Example) instance as input and returns one or more output objects. If the component field values are set in the input, the values will be used in the where clause of the generated SQL. A blank search specification at the root level will query for all the rows from the table corresponding to the root component.

Parent-child relationships are determined by foreign key definitions in the integration object. The Adapter expects the foreign key of a child component to refer to a target key in the parent component. For more information about target keys, see "Background on Database Adapter Integration Objects" on page 59 in the Database Wizard chapter.

Arguments

Query arguments are summarized in Table 5.

Table 5. Query Arguments

Parameter Name	Display Name	Optional	Type	Data Type	Comment
NumOutputObjects	Number of Output Integration Objects	No	Output	Number	Number of output integration objects.
ExtDBODBCDataSource	External DB Data Source	Yes	Input	String	
ExtDBPassword	External DB Password	Yes	Input	String	
ExtDBTableOwner	External DB TableOwner	Yes	Input	String	
ExtDBUserName	External DB UserName	Yes	Input	String	
SiebelMessage	Siebel Message	No	Input/Output	Hierarchy	Input/Output property set should have a Siebel Message as its child.

Delete

The Delete method is used to delete a hierarchy on the external database that is based upon an integration object. Delete takes a QBE instance as input and deletes the entire hierarchy rooted at the specified root component instance. The search specification is only allowed at the root level. If no search specification is provided, Delete removes all rows from the table corresponding to the root component. The CascadeDelete property is specified in the integration object definition at each component level. If this property is set, that component is also deleted when its parent is deleted. Here too, parent-child relationships are determined by the foreign key definition in each component.

Arguments

The arguments for the Delete method are defined in Table 6.

Table 6. Delete Argument

Parameter Name	Display Name	Optional	Type	Data Type	Description
ExtDBODBCDataSource	External DB Data Source	Yes	Input	String	
ExtDBPassword	External DB Password	Yes	Input	String	
ExtDBTableOwner	External DB Table Owner	Yes	Input	String	
ExtDBUserName	External DB UserName	Yes	Input	String	
SiebelMessage	Siebel Message	No	Input	Hierarchy	Child property set

Upsert

The Upsert method is used to insert or update data into the external database based on the input integration object instances. Upsert performs an UPDATE or an INSERT at each component level, depending upon whether the row already exists in the database. The input to the upsert method is the actual integration object instance data.

The Database Adapter uses a combination of two algorithms to upsert data, depending upon the ratio of the number of database rows to the number of component instances in the input instance. This optimization is turned on by default. If the number of database rows is small, it is efficient to query for all rows (of a given parent) and try to match them in memory. If there are a large number of database rows, it is more efficient to query the database for each input component instance, to determine whether the corresponding rowset exists in the database.

Upsert supports multiple user key specification to find the matching row in the database. Each user key is tied in sequence to determine whether the rowset exists in the database. If none of the specified user key fields have their values set, an error is returned. A null value for any of the user key fields is valid.

Arguments

The arguments for the Upsert method are defined in Table 7.

Table 7. Upsert Arguments

Parameter Name	Display Name	Optional	Type	Data Type	Description
ExtDBODBCDataSource	External DB Data Source	Yes	Input	String	
ExtDBPassword	External DB Password	Yes	Input	String	
ExtDBTableOwner	External DB Table Owner	Yes	Input	String	
ExtDBUserName	External DB UserName	Yes	Input	String	
SiebelMessage	Siebel Message	No	Input	Hierarchy	Child property set

Synchronize

Synchronize makes the values of an external database match the values of an integration object instance by performing an Update, Insert, or Delete on the external tables. The Synchronize method is similar to Upsert except that deletes are performed on database rows where corresponding component instances are not present in the input integration objects.

Arguments

The arguments for the Synchronize method are defined in Table 8.

Table 8. Synchronize Arguments

Parameter Name	Display Name	Optional	Type	Data Type	Description
ExtDBODBCDataSource	External DB Data Source	Yes	Input	String	
ExtDBPassword	External DB Password	Yes	Input	String	
ExtDBTableOwner	External DB Table Owner	Yes	Input	String	
ExtDBUserName	External DB UserName	Yes	Input	String	
SiebelMessage	Siebel Message	No	Input	Hierarchy	Input/output property set should have a Siebel Message as its child

Execute

The Database Adapter Execute method is used to perform the operations listed below on an integration object. Any operation can be specified at the component level by using the Op Code.

- Delete
- Upsert
- Synchronize

When Execute performs a Delete or Synchronize on a component, all operations below that component are invalid and are ignored.

Arguments

The arguments for the Execute method are described in Table 9.

Table 9. Execute Arguments

Parameter Name	Display Name	Optional	Type	Data Type	Description
ExtDBODBCDataSource	External DB Data Source	Yes	Input	String	
ExtDBPassword	External DB Password	Yes	Input	String	
ExtDBTableOwner	External DB Table Owner	Yes	Input	String	
ExtDBUserName	External DB UserName	Yes	Input	String	
SiebelMessage	Siebel Message	No	Input	Hierarchy	Input/output property set should have a Siebel Message as its child

Op Codes

Operation codes indicate the type of operation to be performed on an integration component. These codes are specified in the component instance; otherwise, they are inherited from the parent's component instance. Processing an integration component, the Database Adapter detects the operation code and performs the action indicated by the code. For example, when the upsert code is detected, the Database Adapter performs an UPSERT operation, starting at that component level in the hierarchy.

- upsert
- delete
- sync
- none

Operation codes are used with the Execute method to specify the operation at the component level.

Database Adapter Example

This is an example of using the Database Adapter.

```
<?xml version="1.0" encoding="UTF-8"?>

<?Siebel-Property-Set EscapeNames="false"?>

<SiebelMessage MessageId="Insert" MessageType="Integration
Object" IntObjectName="ODBCIntObj" IntObjectFormat="Siebel
Hierarchical">

   <ListOfODBCIntObj>

      <Department>

         <Dept_Id>1</Dept_Id>

         <Name>Engineering</Name>

         <Location>San Mateo</Location>

         <ListOfEmployee>

            <Employee>

               <Dept_Id>1</Dept_Id>

               <Emp_Id>1</Emp_Id>

               <First_Name>John</First_Name>

               <Last_Name>Smith</Last_Name>

               <Salary>100</Salary>

            </Employee>

         </ListOfEmployee>

      </Department>

   </ListOfODBCIntObj>

</SiebelMessage>
```

As you can see in the example XML file above, the IntObjectName field contains the integration name that we have just created:

```
IntObjectName="ODBCIntObj"
```

Additional Information

This section provides additional information about the DB Adapter.

Generating SQL

The Database Adapter generates SQL statements based upon the integration object definitions in tools and data in the input object. The process generates multiple SQL statements, executes them, and joins the result set.

Starting at the root of the tree, the Adapter generates SQL for all children of a component type. For better performance, the Adapter may use an SQL OR clause to group these children together. (Because the maximum length of an SQL statement is limited by ODBC, the length of the SQL statements can be controlled by changing a parameter.)

The Database Adapter business service uses the ODBC API for all database access. With the algorithm outlined above, Database Adapter needs to be processing only one component at a time. All the joins will occur in memory.

The Database Adapter relies upon the ODBC API to cache the underlying ODBC cursors. The ODBC API will not perform connection pooling because it can only handle one open ODBC connection at a time. Database Adapter will cache ODBC connections, if necessary.

Handling Transactions

The Database Adapter does not perform any BeginTransaction or EndTransaction on the Siebel side of data flow transactions. It does perform BEGIN, COMMIT, and ROLLBACK transactions on PeopleSoft database through the ODBC API. The ODBC API provides interfaces to support manual transaction control.

Translating Data Types

The ODBC API, used by the DB Adapter, converts the generic data types specified in the tools to the corresponding ODBC data types. The SQL integration objects are expected to encode these ODBC data types as their External Data Type. The SQL wizard or PeopleSoft wizard (which generates these integration objects) is used to set this external data type. The wizard may query PeopleSoft metadata to obtain database data types such as NUMBER, VARCHAR2, and so on. Then, the ODBC API translates these application-specific data types into ODBC data types.

Passing Connection Parameters

The Database Adapter passes the following connection parameters to the ODBC API for its Connect method:

- ODBC DSN
- Database username
- Database password
- Table owner

The Database Adapter looks for these parameters in the following locations:

- Method arguments
- Server parameters
- Service user properties

The Adapter assumes that these parameters are passed to it by the caller. It will not explicitly retrieve these parameters.

Error Handling

The Database Adapter handles errors like other eAI external adapters. If an error occurs in the PeopleSoft database system, the error is reported by the underlying low-level ODBC layer. Regardless of the error type (database connectivity problem, invalid table/view/column names, and so on), the error from the ODBC layer is passed up the stack to the caller of the DB Adapter. You must define error handling logic using workflow exception steps in order to respond to an error condition.

Passing the PeopleSoft Operations Field to the DTE

Because PeopleSoft must be able to specify operations that will be carried out on Siebel business objects, each Integration Object must have a meta-operator field that can be filled in to specify the operation. The Field Type for the field is set to System (External). The Database Adapter does not use this field; it just passes the field to the Data Transformation Engine (DTE).

4 Database Wizard

This chapter describes how to use the Database Wizard to generate integration objects.

Database Wizard

The Database Wizard is a companion to the Database Adapter. It generates External Integration Objects that represent PeopleSoft data as they are specified in PeopleSoft's database interface. These Integration Objects are then used by the Database Adapter in communicating with PeopleSoft.

After you use the Database Wizard to generate External Integration Objects, you must use the Integration Object Editor to define user keys and define parent-child relationships for the components within these objects. This has to be done because the data relationship and key information (primary and foreign) are not maintained in the database tables that the Wizard uses to create the Integration Object.

Integration Process

To define a new integration point to exchange data with PeopleSoft through its database interface, you must:

1. Select a compatible ODBC driver to provide the connectivity that the Database Adapter and Database Wizard need to access the PeopleSoft database.
2. Identify tables and create any necessary views on the external database.
3. Run the Database Wizard to generate the integration object.
4. Set up keys and parent-child relationships, then compile the integration object.
5. In order to carry out the data exchange, set up the Database Adapter as a step in the Integration Workflow.

Background on Database Adapter Integration Objects

The Database Adapter uses Integration Objects to represent external database tables. Tables appear as integration components within an integration object. The tables need to identify a parent-child relationship, which is usually a subset of the existing relationships.

Model the database table relations, as shown in Figure 4.

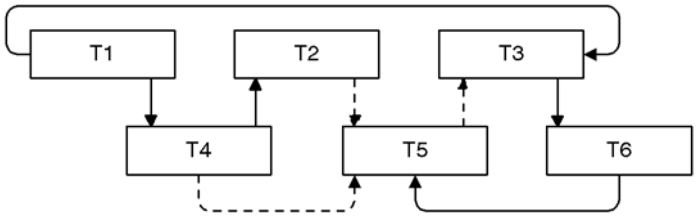


Figure 4. Database Table Relations

Ignore relations that do not contribute to the parent-child structure (represented by the dotted line connections in Figure 4).

Figure 5 shows the final tree to be represented by Siebel Integration Object.

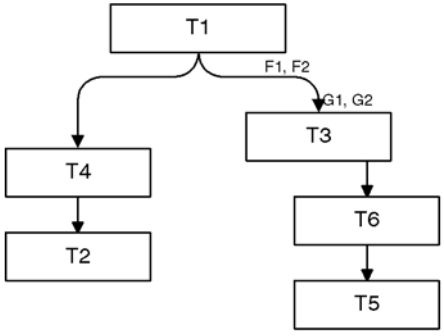


Figure 5. Final Hierarchy

Once the hierarchy is finalized, it is useful to note the keys involved. Each component (table) can have two types of keys—*foreign keys* and *target keys*. Every component that has a parent will have foreign keys. Every component that has a child will have a target key.

In the example, if the fields *F1* and *F2* in the parent table *T1* were mapped to fields *G1* and *G2* in the child table *T3*, then *F1, F2* would be the *target keys* and *G1, G2* would be the foreign keys.

In addition, if the component can be uniquely identified (that is, the table it represents has a logical primary key), it can be represented as a *user key*. User keys function similarly in the Database Adapter as the rest of EAI framework.

Every foreign key has an associated target key in the parent table.

Running the Wizard

This section explains how to run the wizard.

ODBC Sources

The Database Wizard and Database Adapter support standard ODBC drivers. The drivers listed below are validated for Siebel 7.0.

NOTE: See *Release Notes* for specific version numbers.

Database	ODBC Driver
DB2	IBM
Oracle	Merant
SQL Server	Microsoft

Siebel Tools Configuration

In tools.cfg locate the [DBWizSubSys] section and update the following parameters

DBWIZUserName	The Wizard uses this Username to access table definitions.
DBWIZPassword	Password for the Username.
DBWIZODBCDataSource	ODBC data source name.
DBWIZTableOwner	Table owner (prefix).

Wizard Initial Selections

Before proceeding, make sure you have a project locked to create the new Integration Object into. If the starting connection is successful, the wizard prompts you to select one of the following:

- Tables, Views, Synonyms, Aliases
- Tables
- System Tables
- Views
- Temporary Tables
- All

Choose the selection that contains the minimum number of tables and still contains all tables and views of interest. The “All” option will cause the wizard to return a selection containing all tables visible to the user specified in the [DBWizSubSys] section. The second textbox on this dialog prompts for the name of the integration object to be created. Choose a unique name.

Wizard Component Selections

After you click NEXT, the wizard will proceed to query the external database. Depending on various factors (especially the number of tables in the database), this operation may take several minutes. The wizard then brings up the component explorer. You will see all tables listed under one root component. This is only for visual representation—you can ignore the hierarchy and select the components (tables) of interest.

NOTE: You can jump to a component of interest by typing (without pauses) the first few character of the component name.

Completing the Integration Object

Before the generated integration object is usable, the following items need to be configured manually to establish the hierarchy.

Parent Integration Component

The generated object has no parent child relationship. But before the object can be used, exactly one component should be designated as the root, and every other component should be a child component.

Navigate to *Integration Object* > (select the generated integration object) > *Integration Component*. Notice that the *Parent Integration Component* column is empty for all components. For each component, you need to fill this column with its parent integration component's name. At the end of this step, every component except the root integration component should have a value in the *Parent Integration Component* column.

Target Key

A child integration component's foreign keys cannot point to just any field in the parent integration component. The parent integration component needs to be aware of which fields are being utilized by its children as foreign keys and needs to mark these fields by creating an *Integration Component Key*. A key can contain several fields. The relation between the parent's target key fields and the child's foreign key fields is established using the *Key Sequence Number*. The *Key Type* column should be set to *Target Key*.

Foreign Key

Once a parent has grouped some of its fields and created a target key, the children can then create a foreign key. The child component should create a key with *Key Type* column set to *Foreign Key* and the *Target Key* column point to the parent's target key. The number and data type of the constituent fields of the foreign key and the target key should match and the fields should always have a sequence number.

User Keys

For operations that need components to be identified uniquely (example: upsert), you need to setup user keys. Navigate to Integration Component Key and create User Keys and User Key Fields in the usual manner. The *Key Type* should be set to *User Key*. Typically, the logical primary key of the table becomes the User Key of the corresponding integration component.

Common Problems and Solutions

This section describes common DB Wizard-related problems and recommended solutions.

Table 10. Database Wizard Problems and Solutions

Problem	Solution
The wizard is unable to connect to the external database.	**1** Make sure that the ODBC sources exist and all parameters in the [DBWIZSubSys] have valid values. **2** Verify that the CFG file that you are modifying is the one that is actually being used by Siebel Tools. **3** If your underlying database is case sensitive, make sure the username and password specified have the right case.
The wizard connects but returns an error saying no tables were found.	**1** Make sure that the username you provided in [DBWizSubSys] has sufficient privileges to describe the tables of interest. **2** Verify that the table owner specified in the wizard configuration really owns the tables of interest. **3** Some database table owner names are case sensitive—make sure that the case specified in wizard configuration matches the database table owner case. **4** Not all table types that the wizard offers to list are supported by all databases. Try selecting the “All” type if this is an issue.
The wizard starts but the explorer view does not come up.	If the external database has a large number of tables, it may take an unusually long time to query the database and build the explorer view. In this case, wait several minutes for the operation to complete. Also, select a very specific list that contains all objects that you want to view on the first screen instead of the default All selection.
The wizard lists tables, but some tables or views of interest are missing.	**1** Make sure you choose a type that covers all tables of interest. For example, choosing the type “Tables” will eliminate all views and system tables from the generated selections. If this is the issue, choose the “All” type. **2** If the database allows a single external DB ODBC data source to point to multiple databases or defaults database by username, make sure that the data source and username combination used by the wizard selects the database of interest as the default database. **3** Check to be certain that the username the wizard is using has sufficient privileges to describe all tables and views of interest.

Table 10. Database Wizard Problems and Solutions

Problem	Solution
The wizard marks an integration component field of interest as disabled.	This occurs because the Database Adapter does not support the data type of this column. If it is possible to build a view that represents this column as a datatype supported by the DB Adapter, you might be able gain some or all access to this column by using such a view to build your integration component in place of the underlying table.
Can the wizard build objects against a database not validated by Siebel?	Because of the possible number of database and ODBC driver combinations, Siebel supports only the validated configurations. To successfully complete integration with the external database, the Database Adapter must support your configuration. The wizard is only an accessory and integration objects can always be built manually in Siebel Tools.
When should I use Siebel Adapter/Wizard versus Database Adapter and DB Wizard?	When you build and operate integration objects against Siebel tables, you should *always* use Siebel Adapter/Wizard. When you build and operate objects against a *non-Siebel* database, you can use Database Adapter and DB Wizard.
Is it possible to change the XML that is generated by the Integration Object? Is it possible to build an Integration Object instance from existing XML?	Very simple one-one mapping differences between XML tag names can be resolved by changing the *XML Tag* element on various component fields. If there are significant differences or there are structural differences, you will need a Data Transformation step. Siebel EAI Infrastructure supports complex data transformations using powerful programmatic and declarative tools. Refer to *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV* and *XML Integration Reference*.

Example

This example explains step by step the process of creating an integration object using the Database Wizard. This integration object in Siebel will represent the structure of the tables in an external database, and hence the integration between Siebel and the external database becomes possible.

Relevant Tables

The first step in integration using the EAI Database Wizard is to identify the relevant tables in the external database that will be integrated with the Siebel application. This example involves a sample database with a couple of tables. Figure 6 shows the relationship between these tables.

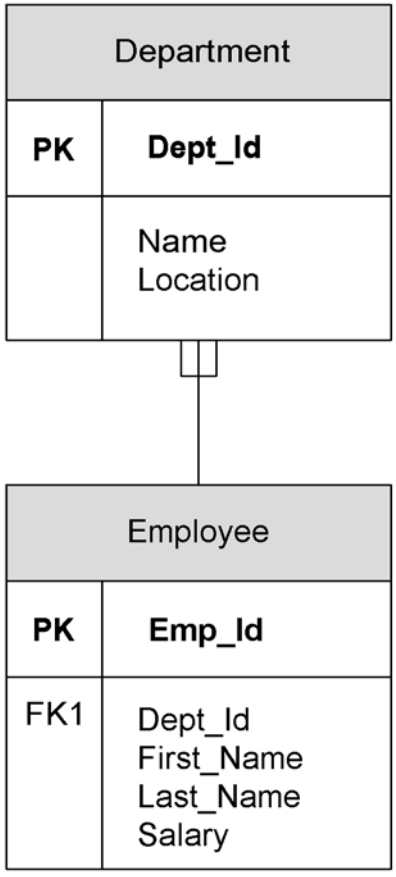


Figure 6. Table Relationships

As the figure above indicates, the external database has two tables—Department and Employee. The department table has three fields—Dept_Id, Name, and Location. Of these fields, Dept_Id is the primary key. The Employee table is a child of the Department table, and this relationship is established through the foreign key Dept_Id in the Department and Employee tables. Other than this field, Employee table also contains Emp_Id, which is the Primary Key for that table and First_Name, Last_Name, and Salary fields. The following sections explain how these tables and fields become integration components and integration component fields, respectively. The accompanying example shows how this integration object facilitates various database operations on these tables in the external database. If you create these tables in an external database and follow along, you can use the following SQL script:

```
drop table Department;
create table Department
(
   Dept_Id      int,
   Name       varchar(5),
   Location   varchar(50)
);


drop table Employee;
create table Employee
(
   Dept_Id    int,
   Emp_Id     int,
   First_Name varchar(25),
   Last_Name  varchar(25),
   salary     int
);
```

The “drop table” lines are added in order to be able to run the above query over and over again. Please make appropriate changes to the above script to make it run on the database of your choice. For example, the datatype varchar used above needs to be changed to varchar2 on an Oracle database. Similarly, you may want to change the datatype for the field salary to something that supports fractional currency.

Creating External DB ODBC Data Source

Once you have identified the database and the tables that you want to integrate Siebel with, you need to create an External DB ODBC data source on your machine (that is, the machine that has Siebel Tools installed) for that external database.

In one of the following steps, information about these data sources will need to be placed in the tools.cfg and uagent.cfg for consumption by Siebel Tools and Siebel Client. Please refer to Chapter 2, “Installing the Siebel Connector for PeopleSoft,” for the detailed information on how to create an external DB ODBC data sources for various RDBMS’s (DB/2, Oracle, and MS SQL server).

Setup Tools Configuration File

Once you have identified the external database and the tables, and created the appropriate external DB ODBC data source to connect to that database, you are almost ready to start creating the integration object. The integration object is created from Siebel Tools, and it needs to know where these external tables are.

These access parameters are set up in the Siebel Tools configuration file (tools.cfg). By default, you should see an empty [DBWizSubSys] section in the tools.cfg file.

So, the following section in the tools.cfg file should be changed to configure the connection to the target database:

```
[DBWizSubSys]
DBWizUserName          = <database login>
DBWizPassword          = <database password>
DBWizODBCWizDataSource = <data source name (as created above)>
DBWIZTableOwner        = <database table owner>
```

Here are sample sections for the three supported platforms.

- **DB2**

```
[ODBCWIZSubSys]
ExtDBUserName         = db2admin
ExtDBPassword         = db2admin
ExtDBODBCDataSource   = SAMPLE
ExtDBTableOwner       = db2admin
```

- **Oracle**

```
[ODBCWIZSubSys]
ExtDBUserName         = fsadmin
ExtDBPassword         = fsadmin
ExtDBODBCDataSource   = SQLOracle
ExtDBTableOwner       = fsadmin
```

- **MS SQL Server**

```
[ODBCWIZSubSys]
ExtDBUserName         = SQLAdapter
ExtDBPassword         = sqladapter
ExtDBODBCDataSource   = SQLAdapter
ExtDBTableOwner       = SQLAdapter
```

Just to show the variations in the values, the ODBC data sources we have created in the previous section have been named SAMPLE for DB/2, SQLOracle for Oracle, and SQLAdapter for MS SQL Server. You may choose a name that is appropriate for your integration project.

Creating the Integration Object

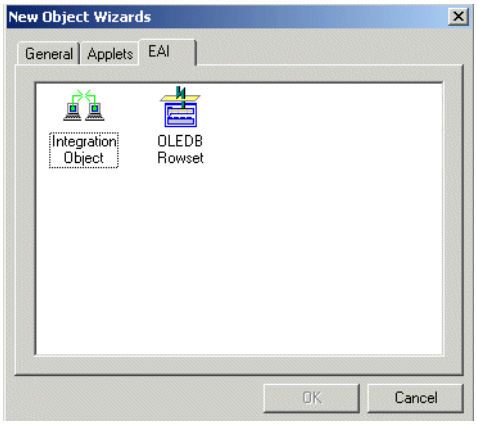
This section explains how to create an Integration Object using the DB Wizard.

Creating a New Integration Object

Use the following steps for creating a new integration object using Database Wizard from Siebel Tools.

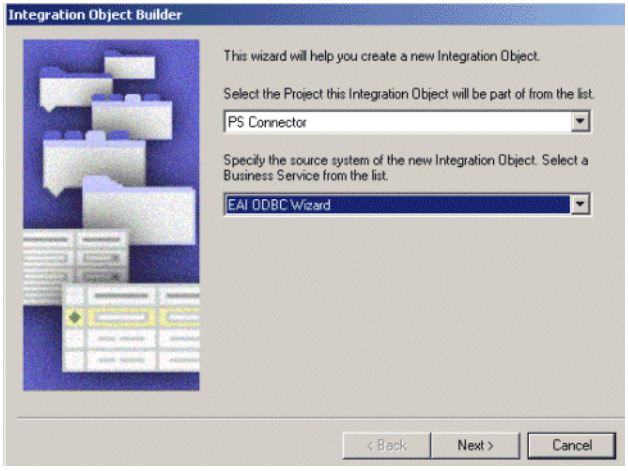
To create a new integration object

1. Select File > New Object … menu item to open the New Object Wizards dialog box.
2. Select the EAI tab in the New Object Wizards dialog box.

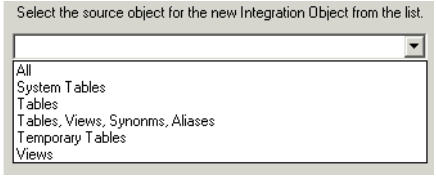


3 Double-click on the Integration Object icon. First choose a project to which you want this integration object to belong to. If you do not see any projects listed in the Projects drop-down list box, you need to press the Cancel button on this window and lock a project from the Siebel Tools main window.

The bottom drop-down list box contains all the EAI based integration object builder wizards. Choose EAI Database Wizard.

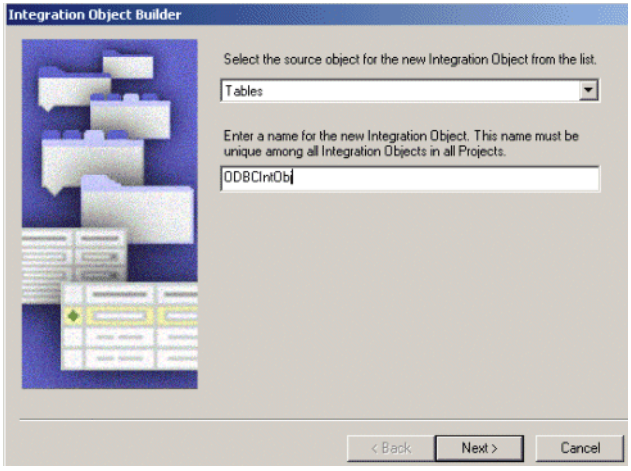


4 Select the components that will be included in the new integration object from the drop-down list.

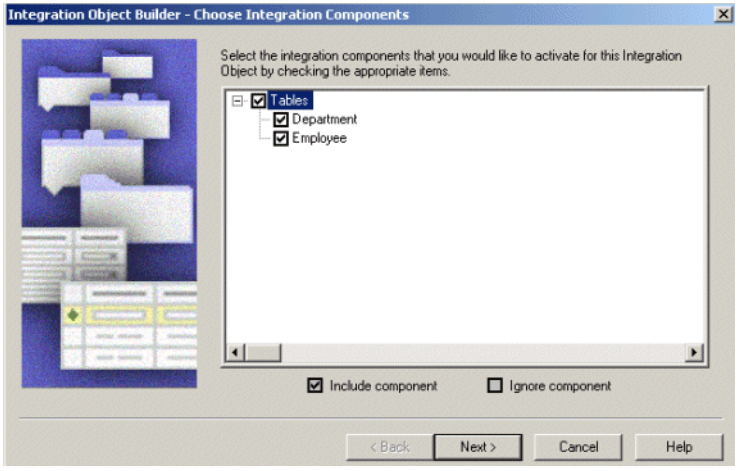


Selecting All will lead to a comprehensive integration object. But it will take a significant amount of time to gather all this information about those objects in the external database.

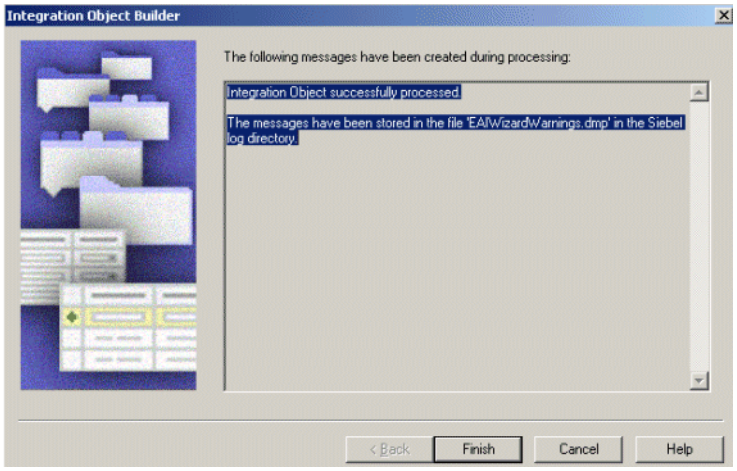
5 Select the source object, enter an unique name for the new integration object, and click Next.



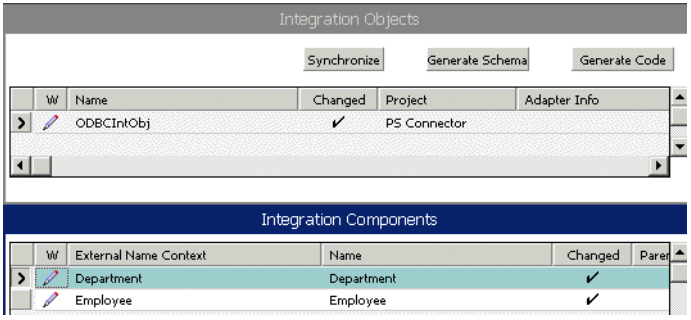
6 In the next screen, select the tables (or views, synonyms, and so forth, depending upon your choices in previous steps) for which the integration components will be built.



7 Once the integration object is created, you will see this message in the following box. If it is not successful, make the appropriate changes specified in the log file and rerun the SQL Database Wizard.



After the object is created, it will be listed, with all its components, in Siebel Tools.



Relationships

Now you need to set up the relationships between these components. Table 11 summarizes various fields that you need to set up for this given example. Individual steps are described in detail in the following sections.

Table 11. Fields

Name	Target Key	User Key	Foreign Key	Parent Integration Object
Department	Dept_Id	Dept_Id	None	None
Employee	None	Emp_Id	Foreign key field: Dept_Id Target Key: Dept_Id	Department

Setting up the Parent Integration Component

Hierarchy among the integration components is defined by using the Parent Integration Component field. When the SQL Database Wizard has finished with making the integration object and its integration components, this field will be empty for all the integration components. In the given example, the Department is parent to an Employee. Also, in this case, Department has no parent; therefore, the Parent Integration Component field will be empty.

In Figure 7, Department has no parent, and the parent of Employee is Department.

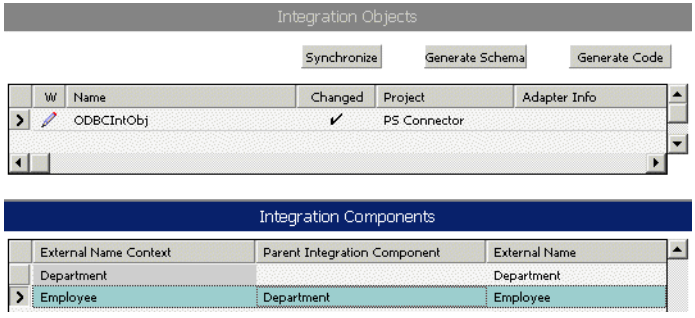


Figure 7. Parent Relationships

Defining Relationships for the Parent

For the parent integration component, you need to define a Target Key and a User Key. Target Key is what the child integration component refers to (through its foreign key) and a User Key uniquely identifies a row in a component. So, both parent and child integration components will have User Keys.

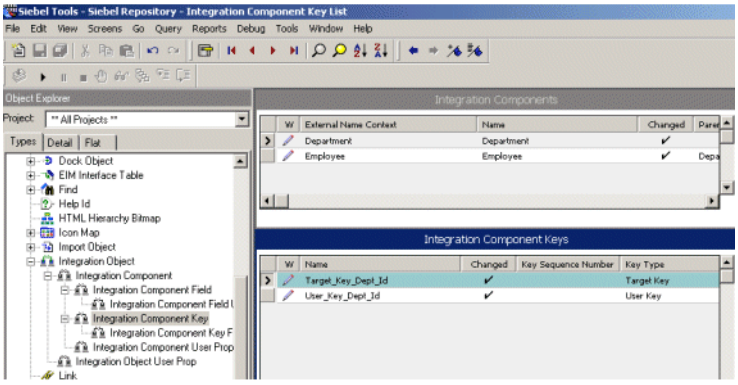


Figure 8. Key Definitions

Define two keys (using Integration Component Key) for the Department integration component. Key Type for one of these keys is Target Key and for the other, it is User Key.

For this example, Dept_Id is used as both Target Key and User Key, since it forms the basis for the parent-child relationship with Employee and also uniquely identifies the records in the Department table.

In Figure 9, Dept_Id is the Target Key. And, Dept_Id is also the User Key for the Department integration component.

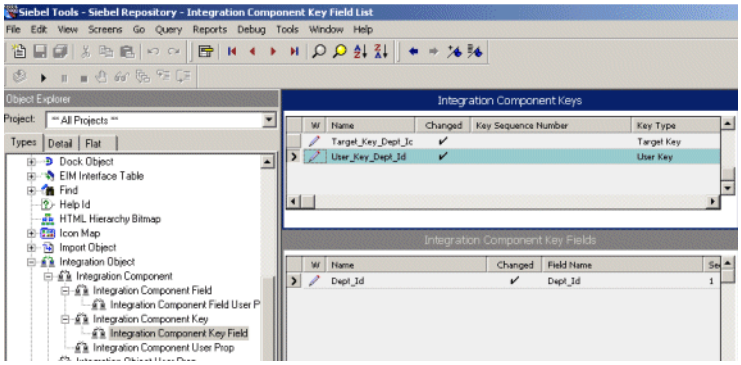


Figure 9. Dept_Id Target Key

Setting up Foreign Key and User Key for the Child

In setting up the hierarchical relationship between the parent (Department) and child (Employee), you need to define which keys in these two tables are related. When you define a foreign key for a child, you need to perform two steps – setting up the target key (that is, the appropriate key in the parent to which this foreign key points to), and the actual field in the foreign key.

In Figure 10, the key type is Foreign Key and the target key name is Target_Key_Dept_Id, which was set up in the parent key structure.

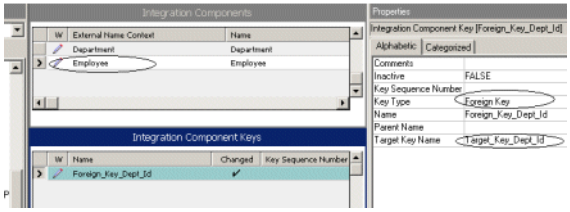


Figure 10. Key Type of Foreign Key and Target Key Name of Target_key_dept_id

For the foreign key, setup the appropriate field from the child. In this case, you select the Dept_Id field from the Employee table.

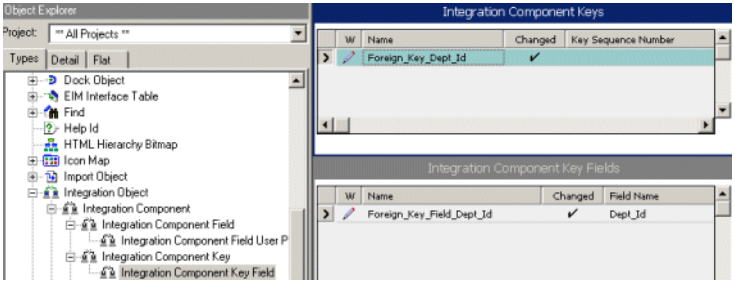


Figure 11. Target and Foreign Keys as Parent and the Child

Now that the relationship between the parent and the child is fully set up (through the Primary Integration Component field and the Target and Foreign Keys as parent and the child, respectively), the last step is to set up keys including a User Key for the child.

In Figure 12, the Employee integration component has its own user key. The field here is Emp_Id from the underlying Employee table. This user key could have contained as many fields as necessary. For example, if the Emp_Id is not unique across the company, but only within a department, then use Emp_Id and Dept_Id as the fields in this user key.

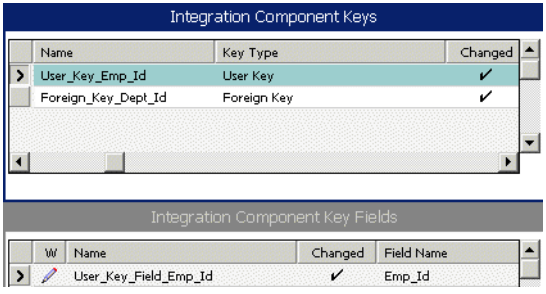


Figure 12. Employee User Key

Compiling

In order for this new integration object to be used and recognized from the Siebel Client, you will need to compile a new.srf. Before proceeding, make sure that the integration object is named appropriately, because you will be using this in the XML datafiles you use to manipulate the data in the external database.

To compile a new SRF

- Select Tools > Compile.

Example

5 Prebuilt Integration Objects

This chapter explains the concepts behind the integration objects and workflows that are provided with the Siebel Connector for PeopleSoft.

Overview

The integration objects and workflows that are provided with the Siebel Connector for PeopleSoft can be used for receiving and sending accounts, employees, and positions from Siebel applications.

Siebel eBusiness Applications includes three prebuilt integration objects to support integration with PeopleSoft: EAI Account, EAI Employee, and EAI Position. The most frequently integrated components of each object are included in the corresponding integration object.

Every integrated component has multiple fields. For example, the Account integration component of the EAI Account integration object has 182 fields. In theory, all of these fields can be used for the outbound integration. This means that values for each of these fields may be written to an XML file for transportation to a PeopleSoft. However, several fields in the integration components are inactive because they are either unlikely to be required or useful in integrating with a PeopleSoft ERP system or not required by Siebel applications. You may activate these fields using Siebel Tools.

User keys uniquely identify the rows in an integration object. A user key may be a single field or a combination of fields. For example, one Account integration component user key combines two fields, Location and Name. If the user key is selected, all account records inserted or updated through this integration component are identified by these two fields. Each integration component can have multiple user keys. And, as the following discussion indicates, user keys are important in sequencing.

In Figure 13, a sample Account integration component has four user keys. When a new record based on this integration component is inserted or updated, the uniqueness of that record is checked first against User Key:1 (Key Sequence Number 1). If the fields that make up this user key are not present in the record under consideration, the next user key is considered (Key Sequence Number 2). If those fields are absent too, then the uniqueness is checked using User Key:3, and so on, until fields for a particular user key are present.

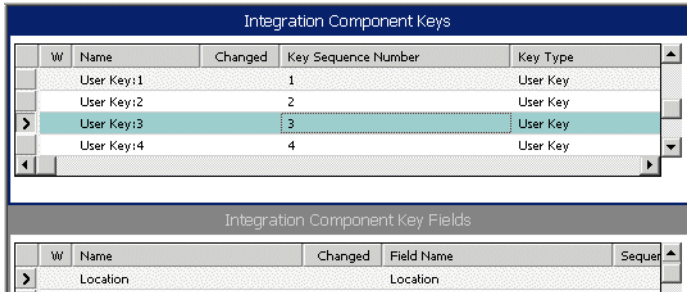


Figure 13. Integration Component Keys and Fields (1)

As Figure 13 indicates, the second user key has an Integration Id field. This field is typically used to store a unique external reference, often from the external system. Using this field, you can enforce the uniqueness of records in two different databases and systems.

If you decide not to input values into the Integration Id, User Key:3 is considered to check the record's uniqueness. This user key contains the fields that you will normally use to uniquely identify an account record. In fact, Name is the required field in the Account Business Component. You can see this in Figure 14.

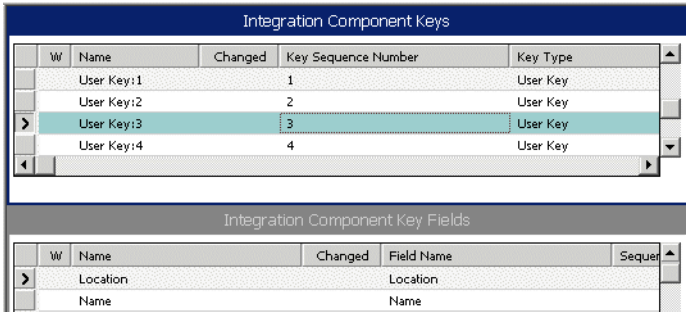


Figure 14. Integration Components and Fields (2)

If you pass an XML file that looks like the example in Figure 15 for an upsert method of Siebel Adapter, User Key 3 will be considered. Because this file contains neither Id nor Integration Id fields, the first two user keys are not used. If you are using the upsert method with the above record, Siebel Adapter will look into the Siebel database and first find out if an account record with name A.K. Parker and Location San Mateo exists. If such a record exists, then that record in Siebel database will be updated. If no such record (Name: A.K. Parker Location: San Mateo) exists, then a new account record will be inserted into the Siebel database.

```
<Account>
  <AccountStatus>Active</AccountStatus>
  <Alias>Parker</Alias>
  <BOExportStatus>1</BOExportStatus>
  <CSN>PARK123</CSN>
  <Competitor>Y</Competitor>
  <CurrencyCode>USD</CurrencyCode>
  <DUNSNumber>123456789</DUNSNumber>
  <Description>This is test description</Description>
  <Division>Div</Division>
  <FreightTerms>FOB</FreightTerms>
  <FreightTermsInfo>Information</FreightTermsInfo>
  <GSAFlag>N</GSAFlag>
  <HomePage>www.akparker.com</HomePage>
  <LanguageCode>ENU</LanguageCode>
  <LastManagerReviewDate>07/20/2001 15:50:11</LastManagerReviewDate>
  <Location>San Mateo</Location>
  <MainFaxNumber>6502341234</MainFaxNumber>
  <MainPhoneNumber>6504531245</MainPhoneNumber>
  <ManagersReview>Test</ManagersReview>
  <Name>A.K. Parker</Name>
  <PartnerFlag>Y</PartnerFlag>
  <PriceList>Test Price List</PriceList>
```

Figure 15. Passing an XML file for an Adapter Upsert Method

List of Values (LOV) Bounded Fields

Some fields in the Siebel business components take values from a predefined List of Values (LOV). The fields in the integration components that are based on such fields can take values from only that list. So, it is important that these lists of values be appropriately mapped.

For example, the field Account Status in the Account integration component of EAI Account integration object depends on a predefined list of values. This can be seen in Figure 16.

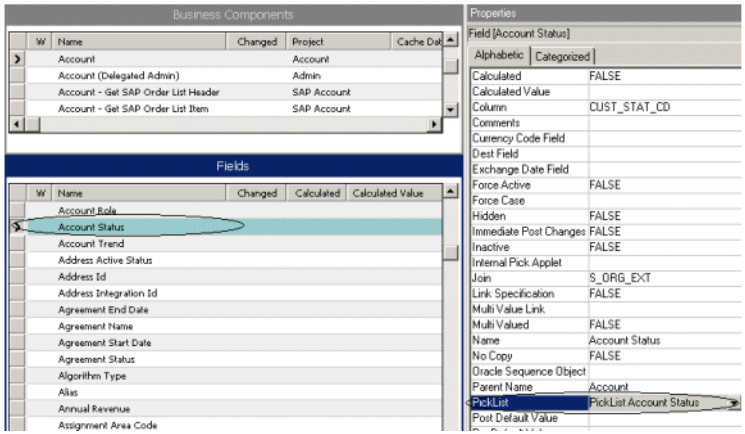


Figure 16. LOV Bounded Fields

As you can see from the Account business component, shown in Figure 17, the field Account Status gets its values from a picklist named PickList Account Status.

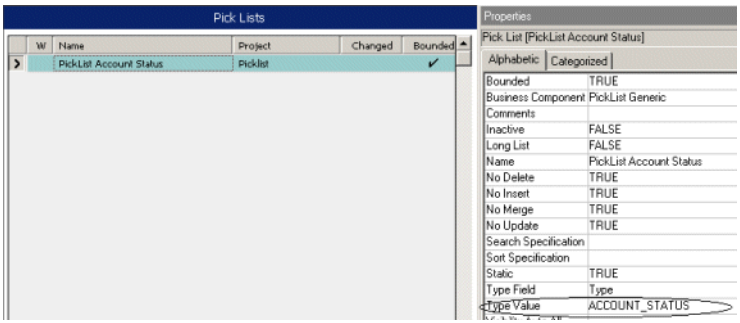


Figure 17. Account Status Values

If you look into the properties of PickList Account Status, you will notice that it has LOV type ACCOUNT_STATUS. You can see this in Figure 18.

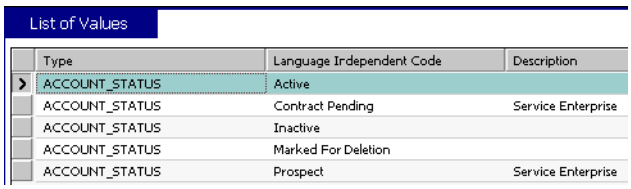


Figure 18. List of Values

And finally, if you query the list of values for that particular type, all the possible values that can go into the Account Status field as the Language Independent Code are displayed. Here, you can input only the above five values (Active, Contract Pending, Inactive, Marked For Deletion, and Prospect) in the Account Status field.

Note that the prebuilt Integration Objects utilize the AllLangIndependentVals user property. The Siebel Adapter expects the language independent code to be specified in LOV bounded fields. If additional values are needed, they should be added to the List of Values. If the names in the external system are different but the meaning is the same, appropriate mapping should be done possibly using the EAI Value Mapping feature.

Picking Related Components

When you insert or update rows into the Siebel database it is often necessary to create relationships with other business components through a foreign key. For example, an account has a M:1 relationship with Price Lists. This means that an account will have one price list associated with it. The same price list could be used for several accounts.

For this type of integration, you store a pointer to the appropriate price list row in the account row. In most cases, there are three ways to identify the foreign row by name, ID, or integration ID. In the Account—Price List example, you can use any one of the following three fields to link these two components: PriceList (name of the price list), Price List Id, or Price List Integration Id.

The related row should already exist in Siebel. In the example above, if you put a Price List name in the Account record, that particular Price List must already exist in Siebel. A new Price List row will not be created if it does not exist.

EAI Account Integration Object

This section describes the EAI Account object, focusing on integration components, related business components, integration component fields, prebuilt workflows, and an example.

Integration Components

The EAI Account integration object incorporates five integration components: Account, Account_Business Address, Account_Organization, Account_Position, and Contact. A Price List can be associated with an Account, and an Organization can be associated with a Contact.

In Figure 19, Account is the central element in the Account integration object. Using this integration object, you can insert, update, or delete information from the account integration component. Account is the primary integration component for this integration object.

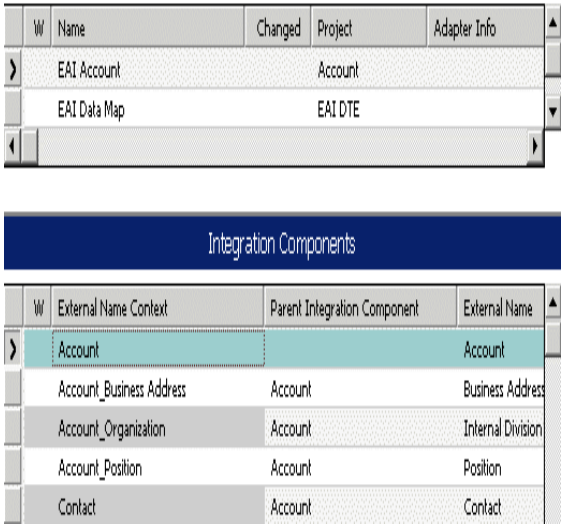


Figure 19. Account Integration Object

Similarly, new rows can be inserted or existing rows updated in the Business Address and Contact entities at the same time that rows are inserted or updated in the Account component. When an account is deleted, all addresses associated with that account are also deleted. However, when an account is deleted, the contact associated with that account is not deleted. The association between the account and contact is deleted. As Figure 19 on page 89 indicates, the relationship between account and contact is many-to-many. Thus, when an account is deleted, the contact associated with this account might be associated with other accounts that are still there in the system.

The relationship between Account/Position and Account/Organization is made through an association. This means that you can not create, update, or delete positions or organizations using this integration object. In other words, when an account is associated with a position or organization, that particular position or organization must already exist in the Siebel database.

Account also has a one-to-many relationship with Price List. This means that a foreign key to Price List (named Price List Id) exists in the Account component. This relationship is implemented through the picklist. An existing Price List can be identified by any one of the following three fields: Price List Id (the foreign key), Price List (the name), or Price List Int Id (the integration id).

Similarly, the relationship between contact and organization is implemented through a picklist. As in Price List, three organization related fields are added to the contact component: Organization Id, Organization, and Contact Organization Integration Id. You can see this in Table 12.

Table 12. Integration Components

Integration Component	Business Component	XML Tag	User Keys	Comments
Account		Account	1. Id 2. Integration Id 3. Location, Name 4. CSN	Account is the Primary Integration Component
Account_ Business Address	Business Address	Account_ BusinessAddres	1. Address Id 2. Address Integration Id	
Account _Organization	Internal Division	Account_ Organization	1. Organization Id 2. Organization Integration Id 3. Organization	
Account_ Position	Position	Account_ Position	1. Position Id 2. Position Integration Id 3. Position	
Contact	Contact	Contact	1. Id 2. Integration Id	

Related Business Components

Related business components are listed in Table 13, below.

Table 13. Related Business Components

Business Component	XML Tag	Foreign Key	Integration Component Fields for Picking
Price List	PriceListId (Account)	Price List Id	Price List Id Price List Integration Id Price List
Organization	OrganizationId (Contact)	Organization Id	Organization Id Contact Organization Integration Id Organization

Integration Component Fields

Table 14 through Table 18 on page 96 identify the integration components for Account, Account_Business_Address, Account_Organization, Account_Position, and Contract.

Account Integration Component

The following table lists the components of Account integration.

Table 14. Account Integration Components

Name	XML Tag	Description
Account Status	AccountStatus	Status of the account. Bounded by a predefined list of values of type ACCOUNT_STATUS.
Alias	Alias	Alternative name for the account.
CSN	CSN	An account number associated with external system that may be the source for accounts.
Competitor	Competitor	Whether this account is a competitor.
Currency Code	CurrencyCode	Base functional currency ID associated with this account.

Table 14. Account Integration Components

Name	XML Tag	Description
DUNS Number	DUNSNumber	DUNS Number (defined by Dun and Bradstreet corporation).
Description	Description	Overall description / comments about the account.
Division	Division	Division name.
EAI Sync Date	EAISyncDate	Date and time on which the last EAI synchronization was performed for this account.
EAI Sync Error Text	EAISyncError Text	The error, if any, from the last EAI synchronization.
EAI Sync Status Code	EAISyncStatusCode	Status of the last EAI synchronization. Bounded by a predefined list of values of type EAI_ACCOUNT_SYNC_STATUS.
Freight Terms	FreightTerms	Freight terms code. Bounded by a predefined list of values of type FREIGHT_TERMS.
Freight Terms Info	FreightTerms Info	Freight terms information.
GSA Flag	GSAFlag	Whether this account is government related and therefore qualifies for GSA pricing.
Home Page	HomePage	URL.
Id	Id	Siebel ID
Integration Id	IntegrationId	ID of this company in an external system.
Language Code	LanguageCode	Preferred language of communication with this account. The language code is bounded by PickList Language.
Last Manager Review Date	LastManagerReviewDate	Last Manager Review Date.
Location	Location	Location of the account—optional part of user key.
Main Fax Number	MainFax Number	Main Fax Number.
Main Phone Number	MainPhone Number	Main Phone Number.
Managers Review	Managers Review	Manager's Review.
Name	Name	Name of the account; non-optional part of the user primary key. *This is a required field.*

Table 14. Account Integration Components

Name	XML Tag	Description
Partner Flag	PartnerFlag	Partner indicator.
Price List	PriceList	Price list unique name. Used to pick price list.
Price List Id	PriceListId	Price list Siebel ID. Used to pick price list.
Price List Integration Id	PriceList IntegrationId	Price list Integration ID. Used to pick price list.
Prospect Flag	ProspectFlag	Prospect flag.
Region	Region	Region Name.
Type	Type	Account type. Bounded by a predefined list of values of type ACCOUNT_TYPE.

Account_Business Address Integration Component

Table 15 lists the Account_Business Address integration components.

Table 15. Account_Business Address Integration Components

Name	XML Tag	Description
Address Active Status	AddressActiveStatus	Status of the address.
Address Id	AddressId	One of the user keys (out-of-the-box) for this integration component. For a given record either this field or Address Integration Id should be specified.
Address Integration Id	AddressIntegrationId	One of the user keys (out-of-the-box) for this integration component. For a given address record, this field or Address Id should be specified.
Bill Address Flag	BillAddressFlag	Whether this is a billing address.
City	City	City designated on address.
Country	Country	Country. Bounded by a predefined list of values of type COUNTRY.
County	County	County.
Email Address	EmailAddress	Electronic mail address.

Table 15. Account_Business Address Integration Components

Name	XML Tag	Description
Fax Number	FaxNumber	Site specific fax number.
Main Address Flag	MainAddressFlag	Whether this is the primary address.
Phone Number	PhoneNumber	Site specific phone number.
Postal Code	PostalCode	Postal Code / ZIP Code.
Province	Province	Province designated.
Ship Address Flag	ShipAddressFlag	Whether this is an address to ship to.
State	State	State abbreviation. Bounded by a predefined list of values of type STATE_ABBREV.
Street Address	StreetAddress	Street address which may be comprised of multiple address lines.
Street Address 2	StreetAddress2	This field can be used in conjunction with Street Address to store multiple lines.

Account_Organization Integration Component

Table 16 lists the Account_Organization integration components.

Table 16. Account_Organization Integration Components

Name	XML Tag	Description
Organization	Organization	Any one of these three fields can be used to associate an account with an organization. This organization to be associated with should already exist in Siebel database.
Organization Id	OrganizationId	
Organization Integration Id	OrganizationIntegrationId	

Account_Position Integration Component

Table 17 lists the Account_Position integration components.

Table 17. Account_Position Integration Components

Name	XML Tag	Description
Position	Position	Any one of these three fields can be used to associate an account with a position. This position to be associated with should already exist in Siebel database.
Position ID	PositionId	
Position Integration ID	PositionIntegrationId	

Contact Integration Component

Table 18 lists the Contract integration components.

Table 18. Contact Integration Components

Name	XML Tag	Description
Active Status	ActiveStatus	Status of the contact.
Alternate Phone #	AlternatePhone	Alternate phone number.
Assistant Phone #	AssistantPhone	Personal Assistant's phone number.
Cellular Phone #	CellularPhone	Cellular phone number.
Contact Organization Integration Id	ContactOrganizationIntegrationId	Contact's organization (potentially different from Account's organization) can be integrated using any of the following three fields: Contact Organization Integration Id, Organization Id, or Organization.
Email Address	EmailAddress	Contact Email address.
Fax Phone #	FaxPhone	Fax phone number.
First Name	FirstName	Contact first name.

Table 18. Contact Integration Components

Name	XML Tag	Description
Home Phone #	HomePhone	Home phone number.
Id	Id	Siebel ID.
Integration ID	IntegrationId	ID of this contact person in an external system.
Job Title	JobTitle	Job title.
Last Name	LastName	Contact last name.
M/F	MF	Sex—Male or Female.
M/M	MM	Personal Title. Bounded by a predefined list of values of type MR_MS.
Middle Name	MiddleName	Middle name.
Organization	Organization	Contact's organization (potentially different from Account's organization) can be integrated using any of the following three fields: Contact Organization Integration Id, Organization Id, or Organization.
Organization ID	OrganizationId	Contact's organization (potentially different from Account's organization) can be integrated using any of the following three fields: Contact Organization Integration Id, Organization Id, or Organization.
Preferred Communications	PreferredCommunications	Preferred communications medium code. Bounded by a predefined list of values of type OFFER_MEDIA.
Preferred Language Code	PreferredLanguageCode	Preferred language ID to communicate with the contact.
Work Phone #	WorkPhone	Work phone number.

Prebuilt Account Workflows

Siebel Connector for PeopleSoft includes two prebuilt workflows for Account integration with PeopleSoft: Account – Send PeopleSoft Customer and Account - Receive PeopleSoft Customer.

Account – Send PeopleSoft Customer

The Account – Send PeopleSoft Customer workflow is used to manage queries from Siebel and writes to an XML file.

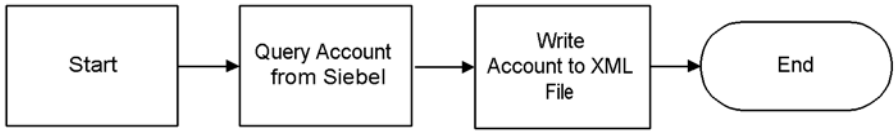


Figure 20. Account – Send PeopleSoft Customer Workflow

Using the two process steps in the above workflow, you can write one or more accounts in the Siebel database to an XML file. The accounts you choose to write to the XML file can be determined by using the Object Id process property of this workflow (Process Properties are discussed below). For example, you can pass the Row Id of the account you wish to obtain into this Object Id parameter. You can do this either programmatically (via Siebel eScript and Data Transformation Engine maps) or via Process Simulator. If you pass a specific value, the account row that matches this row will be returned. If you want all the accounts written to an XML file, leave this Object Id property empty.

Using the XML File Process Property, you can input name and directory location for the XML file. The active fields in the underlying integration object (which is named EAI Account for this workflow) determine the format of the XML file.

Process Properties

The Account Outbound workflow has several process properties. Except for the process property XML File, all others come by default. The XML File process property was created. Of all these process properties, three are a bit more important than others—Object Id, Siebel Message, and XML File.

You can use Object Id to restrict the number of rows returned; leaving it empty will return all the rows. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. And finally, XML File can be used to store the name and location of the XML file to write the results to.

The significance of using Process Properties is that they can be accessed by any workflow step in that given workflow process. In other words, they work as global variables for that entire workflow.

Workflow Steps

This example uses appropriate input arguments for the Write Siebel Message method. Notice that both are process properties. The first argument is File Name. This argument could have been a Literal type instead of being a process property because this variable is not required by other workflow steps.

But this argument makes a process property so that it can be accessed from outside; for example, through another workflow (where this workflow is a subprocess) or through code. Here, you are placing the name of the XML file in this property (for example, C:\AccountOutbound.xml).

The second argument is Siebel Message. The previous workflow step puts the account data into this process property. This workflow step accesses the exact same information (in a hierarchy/property set format), converts it into XML format, and writes it to a file.

This workflow step has no Output Arguments because, once the data is written to an XML file, the job of this workflow process is completed.

This integration involves the following processes:

1 Query Account from Siebel

In the first step, the Query method of the EAI Siebel Adapter business service is used to get data from the Siebel database. This is a standard business service shipped with the Siebel software. Appropriate values for various parameters must be passed to the Query method.

As Figure 20 on page 98 shows, values are passed to two important parameters: Integration Object Name and Object Id. First, a value is passed for the Integration Object Name. This integration object is used to determine which fields need to be queried from the Siebel database. The EAI Siebel Adapter determines this by looking at the various active Integration Components and the active fields within each integration component. As the figure shows, in this case, the underlying integration object is named EAI Account. The contents of this integration object can be seen from Siebel Tools.

Next, the Object Id value is passed to the method. This property can be used to restrict the account rows written to the XML file. The EAI Siebel Adapter queries using the input parameters and places the output into an output argument named Siebel Message (this is a Process Property), which would then be used in the next workflow step. Siebel Message is a hierarchical data structure that is stored in the memory.

2 Write Account to XML File.

After data is obtained about a particular account (in the above step), that data is written to an XML file. In the second step, business service named EAI XML Write to file is used. The business service's Write Siebel Message method is used to write the Siebel Message output from the previous workflow step.

Account – Receive PeopleSoft Customer

The prebuilt Account – Receive PeopleSoft Customer workflow involves reading accounts from an XML file and making upserts into the Siebel application. This workflow is shown in Figure 21.

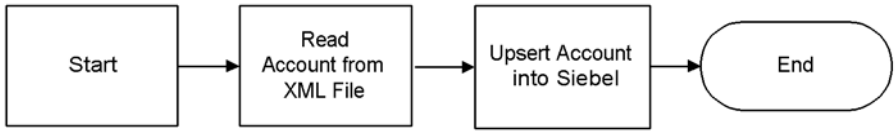


Figure 21. Account – Receive PeopleSoft Customer Workflow

This workflow is the functional reverse of the workflow discussed in the previous section (Account – Send PeopleSoft Customer). Here, the goal is to read an XML file filled with account information and upsert the data into the Siebel database. Upsert involves both updating and inserting data.

If the record already exists in Siebel database, it is updated with any changes supplied in the XML file. If the record does not exist, it is inserted as a new record. This determination is made by using the user keys in the integration object.

Process Properties

The Account Inbound workflow has exactly the same set of process properties as Account Outbound. Except for the process property XML File, all others come by default.

The process property XML File was created. Of all these process properties, here, two have a higher significance – Siebel Message and XML File. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. Also, XML File is the name and directory location of the XML file that contains account records to be loaded.

Workflow Steps

This integration workflow involves the following sequential processes:

1 Read Account from XML File.

In the first step in the workflow in Figure 21 on page 101, the Read Siebel Message method of the EAI XML Read from File business service is used to read data from an external XML file. This is a standard business service shipped with the Siebel software. A value for only one parameter is passed to this method, the XML file location.

XML data from the XML file is converted into Siebel Message, and is kept in the output argument of the same name, as shown below.

This data is read in the next workflow step.

2 Upsert Account into Siebel

In the next step, an Insert or Update method is used to write data to the Siebel database. This needs the Siebel Message output from the previous workflow step. As before, this step decides whether to insert or update a record based on the user keys.

Finally, how would this workflow know which Integration Object it has to go against? This requirement is not listed in the process properties or in the input arguments. The information comes from the input XML file itself.

Looking at the header of the input XML file, notice the following:

At the top of the Account_Inbound.xml file, there is an envelope for Siebel Message. That envelope includes IntObjectName = EAI Account. This is the same integration object that was used for the outbound workflow.

This workflow step has no Output Arguments because the workflow is complete once data is written to Siebel database.

Example

The following example shows how to insert a single Account record into Siebel database. The following discussions analyze the sample XML file used to load the account record and examine the process for inserting the record.

Sample XML File

The following sample XML file is loaded using the prebuilt Account - Send PeopleSoft Customer workflow.

```
<?xml version="1.0" encoding="UTF-8"?>

<?Siebel-Property-Set EscapeNames="false"?>

<SiebelMessage MessageId="9WL-B" MessageType="Integration
Object" IntObjectName="EAI Account" IntObjectFormat="Siebel
Hierarchical">

   <ListOfAccount>

      <Account>

         <AccountStatus>Active</AccountStatus>

         <Alias>Parker</Alias>

         <BOExportStatus>1</BOExportStatus>

         <CSN>PARKER123</CSN>

         <Competitor>Y</Competitor>

         <CurrencyCode>USD</CurrencyCode>

         <DUNSNumber>123456789</DUNSNumber>

         <Description>This is test description</Description>

         <Division>Div</Division>

         <FreightTerms>FOB</FreightTerms>

         <FreightTermsInfo>Information</FreightTermsInfo>

         <GSAFlag>N</GSAFlag>
```

```
            <HomePage>www.akparker.com</HomePage>

            <LanguageCode>ENU</LanguageCode>

            <LastManagerReviewDate>07/20/2001 15:50:11</
LastManagerReviewDate>

            <Location>San Mateo</Location>

            <MainFaxNumber>6502341234</MainFaxNumber>

            <MainPhoneNumber>6504531245</MainPhoneNumber>

            <ManagersReview>Test</ManagersReview>

            <Name>A.K. Parker Incorporated</Name>

            <PartnerFlag>Y</PartnerFlag>

            <PriceList>Test Price List</PriceList>

            <ProspectFlag>N</ProspectFlag>

            <Region>Other</Region>

            <Type>Customer</Type>

            <ListOfAccount_BusinessAddress>

               <Account_BusinessAddress IsPrimaryMVG="Y">

                  <AddressActiveStatus>Y</AddressActiveStatus>

               <AddressIntegrationId>PARK100</AddressIntegrationId>

                  <BillAddressFlag>Y</BillAddressFlag>

                  <City>San Mateo</City>

                  <Country>USA</Country>

                  <County>San Mateo</County>

                  <EmailAddress>admin@akparker.com</EmailAddress>

                  <FaxNumber>6502343456</FaxNumber>

                  <MainAddressFlag>Y</MainAddressFlag>
```

```
            <PhoneNumber>6502343457</PhoneNumber>
            <PostalCode>94402</PostalCode>
            <Province>San Mateo</Province>
            <ShipAddressFlag>Y</ShipAddressFlag>
            <State>CA</State>
            <StreetAddress>123 Main Street</StreetAddress>
            <StreetAddress2>Suite 100</StreetAddress2>
         </Account_BusinessAddress>
      </ListOfAccount_BusinessAddress>
      <ListOfContact>
         <Contact>
            <ActiveStatus>Y</ActiveStatus>
            <AlternatePhone>6501234567</AlternatePhone>
            <AssistantPhone>6501234567</AssistantPhone>
            <CellularPhone>6501235678</CellularPhone>
            <EmailAddress>contact@akparker.com</EmailAddress>
            <FaxPhone>6503452345</FaxPhone>
            <FirstName>Tom</FirstName>
            <HomePhone>6504562345</HomePhone>
            <IntegrationId>PARK100</IntegrationId>
            <JobTitle>Manager</JobTitle>
            <LastName>Jones</LastName>
            <MF>F</MF>
            <MM>Ms.</MM>
            <MiddleName>A</MiddleName>
```

```
                <Organization>Default Organization</Organization>
                <PreferredCommunications>Phone</
PreferredCommunications>
              <PreferredLanguageCode>ENU</PreferredLanguageCode>
                <WorkPhone>6508794567</WorkPhone>
            </Contact>
         </ListOfContact>
         <ListOfAccount_Organization>
            <Account_Organization IsPrimaryMVG="Y">
               <Organization>Default Organization</Organization>
            </Account_Organization>
         </ListOfAccount_Organization>
         <ListOfAccount_Position>
            <Account_Position IsPrimaryMVG="Y">
               <Position>Siebel Administrator</Position>
            </Account_Position>
         </ListOfAccount_Position>
      </Account>
   </ListOfAccount>
</SiebelMessage>
```

DBeclaration

```
<?xml version="1.0" encoding="UTF-8"?>

<?Siebel-Property-Set EscapeNames="false"?>
```

The top line in the XML file identifies the XML version that the underlying parser supports. It also includes information about encoding and Siebel specific values. For more information, see *XML Integration Reference* on the *Siebel Bookshelf*.

Root Element

```
<SiebelMessage MessageId="9WL-B" MessageType="Integration
Object" IntObjectName="EAI Account" IntObjectFormat="Siebel
Hierarchical">

<!--- Other elements --->

</SiebelMessage>
```

The root element of this XML file is SiebelMessage. SiebelMessage has a hierarchical structure. The most relevant attribute for the SiebelMessage element is the name of the integration object. This is specified in the IntObjName attribute, which is the EAI Account in the example above. In our example, the integration object that was created for the account integration is named EAI Account. Using this attribute, the Siebel Adapter looks to the definition of EAI Account integration object to interpret the fields that follow in the following XML file.

```
   <ListOfAccount>

      <Account>

<!--- Other elements --->

      </Account>

   </ListOfAccount>
```

The XMLTag for our integration object (EAI Account) is ListOfAccount. This was already defined in Siebel Tools. This tag makes the upper most element besides the SiebelMessage. Under this record, you can view one or more instances of Account record.

Example ListOfEmployee Integration Component

Figure 22 shows Employee Integration components and the ListOfEmployee XML Container Element.

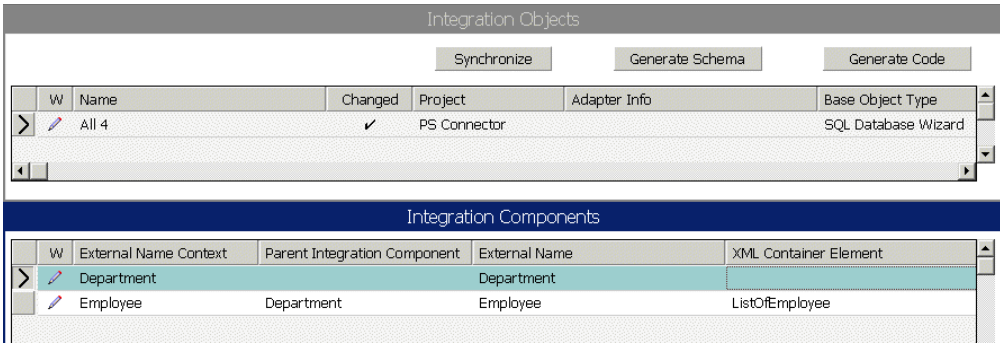


Figure 22. ListOfEmployee Integration Component

Primary and Child Integration Components (Relationship)

The primary and child integration components (relationship) are included in the following XML sample file.

```
<Account>
   <AccountStatus>Active</AccountStatus>
   <Alias>Parker</Alias>
   <BOExportStatus>1</BOExportStatus>

   <ListOfAccount_BusinessAddress>
   </ListOfAccount_BusinessAddress>

   <ListOfContact>
   </ListOfContact>
```

```
<ListOfAccount_Organization>
</ListOfAccount_Organization>

<ListOfAccount_Position>
</ListOfAccount_Position>
```

Moving further down in the hierarchy, you will notice that Account is the parent of ListOfAccount_BusinessAddress, ListOfContact, and so on. This structure exists because Account is the Primary Integration Component in the EAI Account integration object.

As Figure 23 indicates, the Account integration component does not have a Parent Integration Component, where as all the other integration components have Account as the Parent Integration Component.

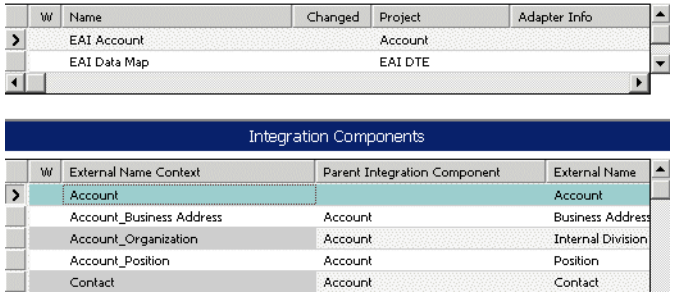


Figure 23. Primary and Child Integration Components (1 of 2)

All this means is that our primary focus for inserting or updating is an account record. And, it just so happens that this primary component (Account) has links other components like address, contact, and so on. You can set a limit on an account so that it has one or more addresses and one or more contacts by selecting the appropriate cardinality.

As Figure 24 indicates, one account can have zero or more addresses, zero or more contacts, and so on. Because of this cardinality, you will see ListOfAccount_BusinessAddress element in the XML file. And, within that parent element, multiple business addresses can be specified. And once the loading has been done (that is, when Siebel Adapter is successful), you will see that all these addresses are attached to the Account you are loading.

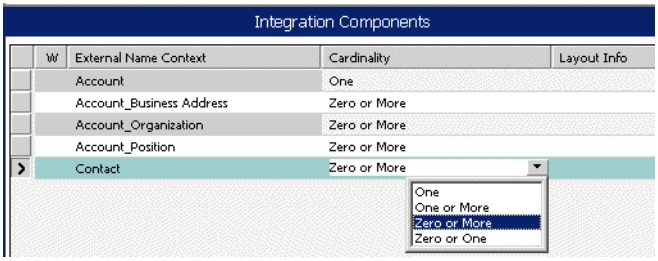


Figure 24. Primary and Child Integration Components (2 of 2)

Individual Integration Components

The individual integration components are included in the following XML file.

```
<ListOfContact>
            <Contact>
               <ActiveStatus>Y</ActiveStatus>
               <AlternatePhone>6501234567</AlternatePhone>
               <AssistantPhone>6501234567</AssistantPhone>
               <CellularPhone>6501235678</CellularPhone>
               <EmailAddress>contact@akparker.com</EmailAddress>
               <FaxPhone>6503452345</FaxPhone>
               <FirstName>Tom</FirstName>
               <HomePhone>6504562345</HomePhone>
               <IntegrationId>PARK100</IntegrationId>
               <JobTitle>Manager</JobTitle>
               <LastName>Jones</LastName>
               <MF>F</MF>
               <MM>Ms.</MM>
               <MiddleName>A</MiddleName>
               <Organization>Default Organization</Organization>
             <PreferredCommunications>Phone</PreferredCommunications>
              <PreferredLanguageCode>ENU</PreferredLanguageCode>
               <WorkPhone>6508794567</WorkPhone>
            </Contact>
          </ListOfContact>
```

In the XML file, these sections deal with the individual integration components. As you can see in Figure 25, the tags for the elements in the preceding XML file come from the XML Tag column of the Integration Component Field. These are just the Active fields in the component. Inactive fields will be omitted from consideration.

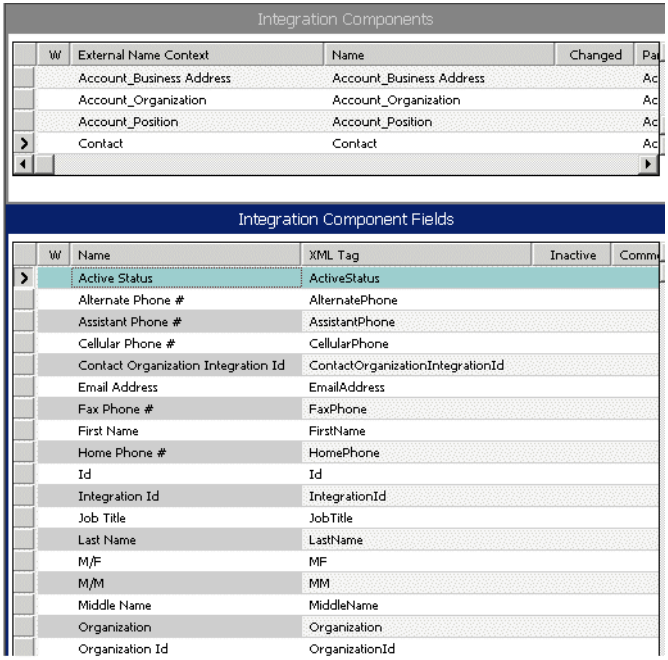


Figure 25. Individual Integration Components (1 of 2)

The Siebel Adapter uses User keys to detemine whether the contact in the XML file already exists in Siebel database or not.

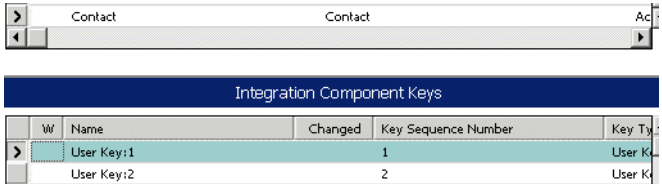


Figure 26. Individual Integration Components (2 of 2)

The Contact integration component has two active user keys. The first key is based on the Id field while the second is based on the Integration Id field. In our example XML file, Id field is not specified. Hence, Siebel Adapter uses Integration Id to decide the uniqueness of this Contact record. In order for the insert or update to take place, at least one user key should be successfully evaluated.

Upserting through a Workflow

The account record in the XML file can be inserted into Siebel database by using the Account – Receive PeopleSoft Customer workflow. See "Account – Receive PeopleSoft Customer" on page 101 for the discussion on this workflow.

EAI Position Integration Object

The EAI Position integration object involves only one integration component, Position. You can associate parent position, billing product, business address, division, and organization to Position.

As you can see in Figure 27, Position is the central entity. Using this integration object, you can insert, update, or delete information from the employee integration component. Position is also the primary integration component for this integration object.

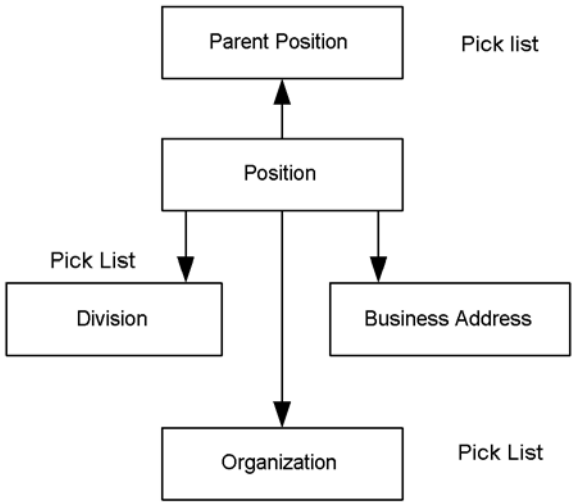


Figure 27. EAI Position Integration Associations (Relationships)

Position has a one-to-one relationship with Parent Position, which means a foreign key to Parent Position (named Parent Position Id) exists in the Position component. This relationship is implemented through a picklist. An existing organization can be identified by any one of the following three fields: Parent Position Id (the foreign key), Parent Position Name (the name), or Parent Position Integration Id (the integration ID).

During position inbound integration, the administrator can sort the positions records in order such that all the parent positions are inserted into Siebel database before their corresponding child positions. For example, the administrator can choose to define the Chief Executive Officer (CEO) of the company first, define its reporting Vice Presidents second, and define the corresponding managers and staff last. Alternatively, the administrator can choose to leave the positions records in random order (for example, the parent positions may be defined after the child positions) and run the whole position inbound process twice. By running the process twice, it makes sure that all child positions are able to associate their parent positions which are not defined by the time the child positions are created.

Similarly, as Figure 27 on page 114 indicates, the relationship between position and division, the relationship between position and organization, and the relationship between position and business are implemented through a picklist.

An existing division can be identified by any one of the following three fields: Division Id (the foreign key), Division (the name), or Division Integration Id (the integration id). Note that upon picking a division, if the division is already associated with an organization, the associating info of the organization with the position will be filled in automatically.

Alternatively, an existing organization can be identified by any one of the following three fields: Organization Id (the foreign key), Organization (the name), or Organization Integration Id (the integration id).

The business addresses of a position are automatically retrieved from the business addresses, if they exist, of the corresponding division. During a position integration, only primary business address of the position can be specified from the list and it can be identified by any one of the following fields: Primary Position Address Id (the foreign key) and Primary Position Address Integration Id (the integration ID).

Integration Components

Table 19 identifies the components associated with the Position integration object.

Table 19. Integration Components

Integration Component	Business Component	XML Tag	User Keys
Position	Position	Position	1. ID 2. Integration ID 3. Division, Name 4. Name

Related Business Components

Table 20 lists the related business components.

Table 20. Related Business Components

Business Component	XML Tag	Foreign Key	Integration Component Fields for Picking
PS Position Type	BillingProductID (Account)	Billing Product ID	Billing Product ID Billing Product Integration ID Billing Product
Internal Division	DivisionID (Account)	Division ID	Division ID Division Integration ID Division
Organization	OrganizationID (Contact)	Organization ID	Organization ID Organization Integration ID Organization

Table 20. Related Business Components

Business Component	XML Tag	Foreign Key	Integration Component Fields for Picking
Position	ParentPositionID	Parent Position ID	Parent Position ID Parent Position Integration ID Parent Position Name
Business Address	Primary Position Address ID	Primary Position Address ID	Primary Position Address ID Primary Position Address Integration ID

Integration Component Fields

Position Integration Component

The Position integration component includes fields from the base position business component. These components are listed in Table 21.

Table 21. Position Integration Components

Name	XML Tag	Description
Billing Product	BillingProduct	
Billing Product Id	BillingProductId	
Billing Product Integration Id	BillingProduct IntegrationId	
Compensatable	Compensatable	
Description	Description	Description of the position.
Division	Division	The name of the division that the position belongs to.
Division Id	DivisionId	The ID of the division to which the position belongs. *This is a Required Field.*

Table 21. Position Integration Components

Name	XML Tag	Description
Division Integration Id	DivisionIntegrationId	The integration ID of the division to which the position belongs.
ID	ID	The ID of the position.
Integration Id	IntegrationId	The integration ID of the position.
Name	Name	The name of the position.
Organization	Organization	The name of the organization to which the position belongs.
Organization Id	OrganizationId	The ID of the organization to which the position belongs.
Organization Integration Id	OrganizationIntegrationId	The integration ID of the organization to which the position belongs.
Parent Position Id	ParentPositionId	The ID of the parent position.
Parent Position Integration Id	ParentPositionIntegrationId	The integration Id of the parent position.
Parent Position Name	ParentPositionName	The name of the parent position.
Position Type	PositionType	The type of position.
Primary Position Address Id	PrimaryPositionAddressId	The ID of the primary business address of the position.
Primary Position Address Integration Id	PrimaryPositionAddressIntegrationId	The Integration ID of the primary business address of the position.

Prebuilt Position Workflows

Two workflows are associated with Position integration: Position – Send PeopleSoft Customer workflow and Position – Receive PeopleSoft Customer workflow. These workflows are described in the following sections.

Position – Send PeopleSoft Customer Workflow

The Position – Send PeopleSoft Customer workflow involves querying from the Siebel application for position data and writing the data to an XML file.

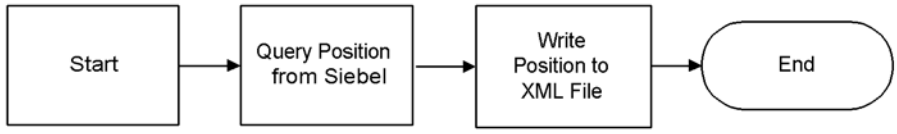


Figure 28. Position – Send PeopleSoft Customer Workflow

Using the two process steps in the above workflow, you can obtain one or more position records from the Siebel database to an XML file. The positions you choose to write to the XML file can be determined by using the Object Id process property of this workflow (Process Properties are discussed below). For example, you can pass the Row Id of the position you wish to obtain into this Object Id parameter. You can do this either programmatically (using Siebel eScript and Data Transformation Engine maps) or with Process Simulator.

If you pass a specific value, the position row that matches this row will be returned. If you want all the positions written to an XML file, leave this Object Id property empty.

Using the XML File Process Property, you can input name and directory location for the XML file. The active fields in the underlying integration object (which is named EAI Position for this workflow) determine the format of the XML file.

Process Properties

The Position Outbound workflow has several process properties. Except for the process property XML File, all others come by default. The XML File process property was created. Of all these process properties, three are a bit more important than others—Object Id, Siebel Message, and XML File. You can use Object Id to restrict the number of rows returned; leaving it empty will return all the rows. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. And finally, XML File can be used to store the name and location of the XML file to write the results to.

Workflow Steps

As shown in Figure 28 on page 119, this integration workflow involves the following sequential processes:

1 Query Position from Siebel.

The first step is to get data about the position(s) from the Siebel database. Here, the Query method of the EAI Siebel Adapter business service is used to get data from the Siebel database. This is a standard business service shipped with the Siebel software.

Two important parameters are passed to the Query method. First, the Integration Object Name is passed to the method. This integration object is used to determine which fields need to be queried from the Siebel database. The EAI Siebel Adapter determines this by looking at the various active Integration Components and the active fields within each integration component. In this case, the underlying integration object is named EAI Account. The contents of this integration object can be seen from Siebel Tools.

Next, the Object Id, a process property, is passed. This property can be used to restrict the account rows written to the XML file.

The EAI Siebel Adapter queries using the input parameters and places the output into an output argument named Siebel Message (this is a Process Property), which will be used in the next workflow step. Siebel Message is a hierarchical data structure that is stored in the memory.

2 Write Position to XML File.

After data is obtained about a particular account (in the Step 1), it is written to an XML file. Another business service named EAI XML Write to File is used. The Write Siebel Message method is used to write the Siebel Message output from the previous workflow step.

Now some discussion about the input arguments the above method takes. Notice that both are process properties. File Name could have been of Literal type (instead of being a process property), because this variable is not required by other workflow steps. But this argument has been made a process property so that it can be accessed from outside (for example, through another workflow—where this workflow is a subprocess; or through code). In essence, you will be placing the name of the XML file in this property (that is, C:\PositionOutbound.xml).

The second argument is Siebel Message. The previous workflow step would have placed the account data into this process property. This workflow step accesses the exact same information (which is in a hierarchy and property set format) and converts it into XML format and writes it to a file.

This workflow step has no Output Arguments because once the data is written to an XML file, the job of this workflow process is completed.

Position – Receive PeopleSoft Customer Workflow

The Position - Receive PeopleSoft Customer workflow involves reading position data from an XML file and upserting data into the Siebel application.

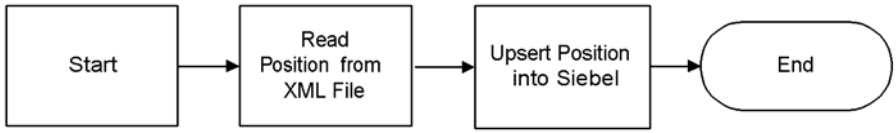


Figure 29. Position - Receive PeopleSoft Customer Workflow

This workflow does the reverse of the workflow discussed in the previous section (Position – Send PeopleSoft Position). Here the goal is to read an XML file filled with position(s) information, and upsert them into the Siebel database. Here the upsert stands for a combination of update and insert. Meaning, if the record already exists in Siebel database, then the record will be updated, using any changes supplied in the XML file. If no such record exists, then this record will be inserted as a brand new record. This determination is made by using the user keys in the integration object.

Process Properties

The Position Inbound workflow has exactly the same set of process properties as Position Outbound. Just like there, except for the process property XML File, all others come by default. The XML File process property was created. Of all these process properties, here, two have a higher significance—Siebel Message and XML File. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. And, XML File is the name and directory location of the XML file that contains account records to be loaded.

Workflow Steps

As shown in Figure 29 on page 122, this integration workflow involves the following sequential processes:

1 Read Position from XML File.

At first, the workflow uses Read Siebel Message method of the EAI XML Read from File business service to read data from an external XML file. This is a standard business service shipped with the Siebel software.

A value for one parameter, location of the XML file, is passed to this method. The XML data from the above file is converted into Siebel Message, and is kept in the output argument of the same name.

This data is read by the next workflow step.

2 Upsert Position into Siebel.

This step writes the data to Siebel database. The EAI Siebel Adapter business service is used. But now, the Insert or Update method is used. As the name suggests, this method inserts or updates the record according to need.

All this needs is the Siebel Message output from the Read Position from XML File workflow step. As discussed before, this step decides on whether to insert or update a record, based on the user keys.

Finally, how would this workflow know which Integration Object it has to go against? This is neither listed in the process properties nor in the input arguments. This information comes from the input XML file itself. See the Example on page 124 for details.

This workflow step has no Output Arguments because once the data is written to Siebel database, the job of this workflow process is complete.

Example

The following XML file is loaded using the prebuilt Employee - Receive PeopleSoft Employee workflow.

```
<?xml version="1.0" encoding="UTF-8"?>

<?Siebel-Property-Set EscapeNames="false"?>

<SiebelMessage MessageId="PositionInboundSample"
MessageType="Integration Object" IntObjectName="EAI Position"
IntObjectFormat="Siebel Hierarchical">

  <ListOfPosition>

     <Position>

         <BillingProduct>Field Engineer</BillingProduct>

         <IntegrationId>PosId123</IntegrationId>

         <Compensatable>Y</Compensatable>

       <DivisionIntegrationId>SiebelAdm1</DivisionIntegrationId>

         <Name>Engineering Manager</Name>

         <OrganizationIntegrationId>DefOrg1</
OrganizationIntegrationId>

         <PositionType>Manager</PositionType>

         <Description>Insert all fields using Integration Id</
Description>

         <ParentPositionId>0-5220</ParentPositionId>

         <PrimaryPositionAddressId>1-VGHC</
PrimaryPositionAddressId>

     </Position>

  </ListOfPosition>

</SiebelMessage>
```

The following XML example includes child positions that are defined before the parent positions using the prebuilt Employee - Receive PeopleSoft Employee workflow. The following XML are run twice to make sure the child position can associate its parent position the second time.

```
<?xml version="1.0" encoding="UTF-8" ?>

<?Siebel-Property-Set EscapeNames="false"?>

<SiebelMessage MessageId="PositionInboundSample2"
MessageType="Integration Object" IntObjectName="EAI Position"
IntObjectFormat="Siebel Hierarchical">

 <ListOfPosition>

     <Position>

     <BillingProduct>Field Engineer</BillingProduct>

   <Compensatable>Y</Compensatable>

   <Division>Siebel Engineering</Division>

   <Name>Software Engineer</Name>

   <Organization>Default Organization</Organization>

   <PositionType>Developer</PositionType>

   <ParentPositionName> Engineering Manager </ParentPositionName>

   <Description>This is a child position. Its parent position is
Engineering Manager</Description>

     </Position>

     <Position>

   <BillingProduct>Field Engineer</BillingProduct>

   <Compensatable>Y</Compensatable>

   <Division>Siebel Engineering</Division>

   <Name>Engineering Manager </Name>
```

```
   <Organization>Default Organization</Organization>

   <PositionType>Manager</PositionType>

   <ParentPositionName />

   <Description>This is the parent position of the child position,
Software Engineer</Description>

     </Position>

  </ListOfPosition>

</SiebelMessage>
```

EAI Employee Integration Object

The EAI Employee integration object includes five integration components: Employee, Employee_ERM Employee Period Summary, Employee_Position, Employee_Responsibility, and Employee_StartHomeAddress. You can associate an Organization with the Employee object.

As you can see in Figure 30, Employee is the central entity. Using this integration object, you can insert, update, or delete information from the employee integration component. Employee is the primary integration component for this integration object.

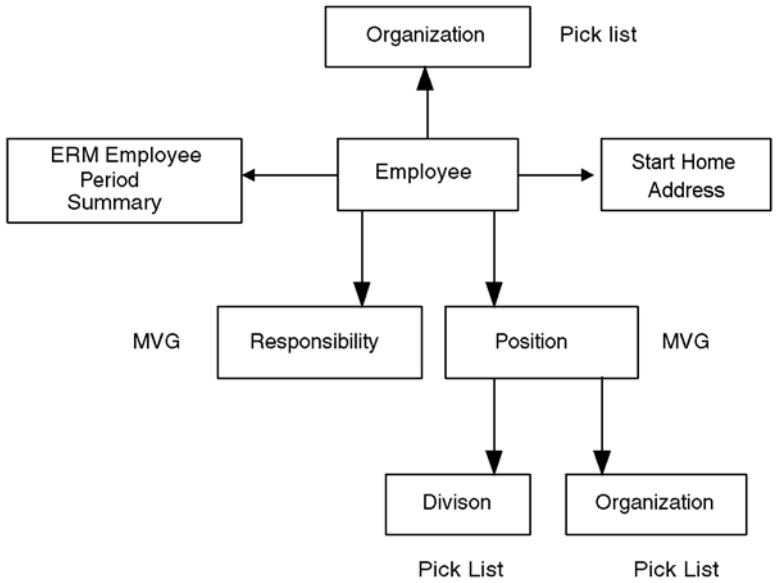


Figure 30. Employee as Central Entity

The relationship between Employee/Start Home Address and Employee/ERM Employee Period Summary come through an MVG. New rows can be inserted or existing rows updated in the Start Home Address entities during the same time rows are inserted or updated in the Employee component. When an account is deleted, all the addresses associated with that account are also deleted. Insert, update or delete of ERM Employee Period Summary is not supported in this release. The information of ERM Employee Summary is available for read-only.

The relationship between Employee/Position and Employee/Responsibility come through MVG associations. Therefore, this integration object cannot be used to create, update, or delete positions or responsibilities. In other words, when you associate an employee with a position or a responsibility, that particular position or responsibility should already exist in Siebel database. As you can see from the ERD, these components are picked through Multi-Value Groups (MVGs). Hence, each employee can be associated with multiple positions or responsibilities.

Employee has a one-to-one relationship with Organization, which means a foreign key to Organization (named Employee Organization Id) exists in the Employee component. This relationship is implemented through a picklist. An existing organization can be identified by one of the following fields: Employee Organization Id (the foreign key) or Employee Organization Integration Id (the integration ID).

Similarly, as Figure 30 on page 127 indicates, the relationship between position and division and the relationship between position and organization are implemented through a picklist.

An existing division can be identified by any one of the following three fields: Division Id (the foreign key), Division (the name), or Division Integration Id (the integration ID).

An existing organization can be identified by any one of the following three fields: Organization Id (the foreign key), Organization (the name), or Organization Integration Id (the integration ID).

Integration Components

The Employee integration objects and associated components are listed in Table 22.

Table 22. Employee Integration Objects and Components

Integration Component	Business Component	XML Tag	User Keys	Comments
Employee	Employee	Employee	1. ID 2. Integration ID 3. Login Domain, Login Name 4. Login Name 5. First Name, Last Name, Middle Name	Employee is the Primary Integration Component
Employee_ERM Employee Period Summary	ERM Employee Period Summary	Employee_ ERMEmployee PeriodSummary	1. Period Summary ID 2. Performance Measure, Period Summary End Date	Performance review of Employee
Employee_ Position	Position	Employee_ Position	1. Position ID 2. Position Integration ID 3. Division, Position 4. Division Integration ID, Position Integration ID	Positions held by the Employee
Employee_ Responsibility	Responsibility	Employee_ Responsibility	1. Responsibility ID 2. Responsibility	Responsibility held by the Employee
Employee_Start Home Address	Personal Address	Employee_Start HomeAddress	1. Start Home Address Address ID 2. Start Home Address Integration ID 3. Start Home Address Name	Employee's starting home address

Related Business Components

The related business components are listed in Table 23.

Table 23. Related Business Components

Business Component	XML Tag	Foreign Key	Integration Component Fields for Picking
Organization (Employee)	EmployeeOrganizationID	Employee Organization ID	Employee Organization ID Employee Organization
Division (Employee_Position)	DivisionID	Division ID	Division ID Division Integration ID Division
Organization (Employee_Position)	OrganizationID	Organization ID	Organization ID Organization

Integration Component Fields

Employee Integration Component fields are identified in Table 24 on page 131 through Table 28 on page 135.

Employee Integration Component

Employee integration component draws fields from the base account business component, ERM Employee Period Summary MVG, Position MVG, Responsibility MVG, and Start Home Address MVG.

Base Account Business Component Fields

Table 24 lists base account business component fields that are used by the Employee Integration Object.

Table 24. Base Account Business Component Fields

Name	XML Tag	Comments
Alias	Alias	Alternative name for the employee.
Cell Phone #	CellPhone	Cell phone number.
EMail Addr	EMailAddr	Email Address.
Emp #	Emp	Employee Number.
Employee Organization	EmployeeOrganization	Name of the organization to which the employee belongs.
Employee Organization Id	EmployeeOrganizationId	ID of the organization to which the employee belongs.
Fax #	Fax	Fax Number.
First Name	FirstName	This is a required field.
Hire Date	HireDate	
Home Phone #	HomePhone	Home phone number.
Id	Id	
Integration Id	IntegrationId	Integration ID of the employee. Must be unique in the database.
Job Title	JobTitle	
Last Name	LastName	This is a required field.
Login Domain	LoginDomain	HQ, Internal, External, and so on. This is a required field.
Login Name	LoginName	Must be unique in the database. This is a required field.
M/F	MF	Male or Female.
Maiden Name	MaidenName	

Table 24. Base Account Business Component Fields

Name	XML Tag	Comments
Middle Name	MiddleName	
Next Annual Review Date	NextAnnual ReviewDate	Next annual review date of the employee.
Nick Name	NickName	Nickname of the employee.
Pager PIN	PagerPIN	
Pager Phone #	PagerPhone	
Pager Type	PagerType	This is a picklist field. The List of value type (LOV) is PAGER_TYPE.
Party Id	PartyUId	Another Employee ID. Must be unique in the database. This is a required field.
Personal Title	PersonalTitle	This is a picklist field. The List of value type (LOV) is MR_MS.
Phone #	Phone	Main Work Phone number.
Share Address Flag	ShareAddressFlag	This is a Boolean field, Y/N. If share address flag is Y, the position business address is shared with the calculated address fields (that is, Address –Calc).
Share Home Phone Flag	ShareHomePhoneFlag	This is a Boolean field, Y/N. If share address flag is Y, the home address is shared to Home Phone # –Calc.
Start Home Address Id	StartHomeAddressId	ID of the Employee's starting home address.
Termination Date	TerminationDate	The termination date of the employment.

ERM Employee Period Summary MVG Component Fields

Table 25 lists ERM Employee Period Summary MVG component fields

NOTE: This is read-only information (outbound integration).

Table 25. ERM Employee Period Summary MVG Component Fields

Name	XML Tag	Comments
Performance Measure	PerformanceMeasure	
Period Summary End Date	PeriodSummaryEndDate	The end date of the performance review period.
Period Summary Id	PeriodSummaryId	ID of the review.
Period Summary Rollup Period Type	PeriodSummaryRollupPeriodType	The frequency of performance review (quarterly, annually).
Period Summary Score	PeriodSummaryScore	The score of the review.
Period Summary Submit To Mgr First Name	PeriodSummarySubmit ToMgrFirstName	The first name of the manger who performs the review of the employee.
Period Summary Submit To Mgr Id	PeriodSummarySubmit ToMgrId	The employee ID of the manger who performs the review of the employee.
Period Summary Submit To Mgr Last Name	PeriodSummarySubmitToMgrLastName	The last name of the manger who performs the review of the employee.
Period Summary Submit To Mgr Integration Id	PeriodSummarySumbitToMgr IntegrationId	The integration ID of the manger who performs the review of the employee.
Period Summary Submit To Mgr Login	PeriodSummarySumbitToMgrLogin	The login name of the manger who performs the review of the employee.

Position MVG Component Fields

Table 26 lists Position MVG component fields.

Table 26. Position MVG Component Fields

Name	XML Tag	Comment
Division	Division	The name of division to which the position belongs.
Division Id	DivisionId	The ID of the division to which the position belongs.
Division Integration Id	DivisionIntegrationId	The integration Id of the division to which the position belongs.
Employee End Date	EmployeeEndDate	The end date of the position being performed by the employee.
Employee Start Date	EmployeeStartDate	The start date of the position being performed by the employee.
Organization	Organization	The name of the organization to which the position belongs.
Organization Id	OrganizationId	The ID of the organization to which the position belongs.
Position	Position	The name of the position.
Position Id	PositionId	The ID of the position.
Position Integration Id	PositionIntegrationId	The integration ID of the position.

Responsibility MVG Component Fields

Table 27 lists Responsibility MVG component fields.

Table 27. Responsibility MVG Component Fields

Name	XML Tag	Comments
Responsibility	Responsibility	The name of the responsibility
Responsibility Id	ResponsibilityId	The ID of the responsibility

Start Home Address MVG Components Fields

Table 28 lists Responsibility MVG component fields.

Table 28. Start Home MVG Component Fields

Name	XML Tag	Comments
Start Home Address	StartHomeAddress	
Start Home Address Address Id	StartHomeAddressAddressId	Siebel ID
Start Home Address Integration Id	StartHomeAddressIntegrationId	Integration ID

Prebuilt Employee Workflows

Siebel 7 provides two prebuilt workflows for Employee integration with PeopleSoft: Employee – Send PeopleSoft Customer Workflow and Employee – Receive PeopleSoft Customer Workflow. These workflows are described in the following sections.

Employee – Send PeopleSoft Customer Workflow

The Employee – Send PeopleSoft Customer workflow, shown in Figure 31, involves querying employees from the Siebel application and writing the data to an XML file.

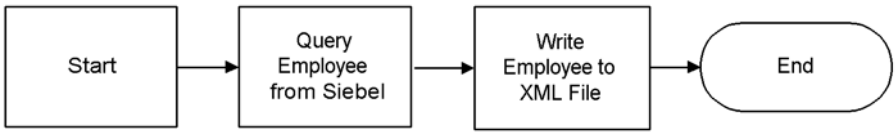


Figure 31. Employee – Send PeopleSoft Customer Workflow

Using the two process steps in the above workflow, you can obtain one or more employee records from the Siebel database to an XML file. The employees you choose to write to the XML file can be determined by using the Object Id process property of this workflow (Process Properties are discussed below).

For example, you can pass the Row Id of the employee you wish to obtain into this Object Id parameter. You can do this either programmatically (using Siebel eScript and Data Transformation Engine maps) or through Process Simulator. If you pass a specific value, the employee row that matches this row will be returned. If you want all the employees written to an XML file, leave this Object Id property empty.

Using the XML File Process Property, you can input name and directory location for the XML file. The active fields in the underlying integration object (which is named EAI Employee for this workflow) determine the format of the XML file.

Process Properties

The Employee Outbound workflow has several process properties. Except for the process property XML File, all others come by default. The XML File process property was created. Of all these process properties, three are a bit more important than others – Object Id, Siebel Message, and XML File. You can use Object Id to restrict the number of rows returned; leaving it empty will return all the rows. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. And finally, XML File can be used to store the name and location of the XML file to write the results to.

Workflow Steps

As shown in Figure 31 on page 136, this integration workflow involves the following sequential processes:

1 Query Employee from Siebel.

At first, the Query method of the EAI Siebel Adapter business service is used to extract data from Siebel database. This is a standard business service shipped with the Siebel software.

Two important parameters are passed to the Query method. First, the Integration Object Name is passed. This integration object is used to determine which fields need to be queried from the Siebel database. The EAI Siebel Adapter determines this by looking at the various active Integration Components and the active fields within each integration component. In this case, the underlying integration object is named EAI Account. The contents of this integration object can be seen from Siebel Tools.

Next, the Object Id parameter, a process property is passed. This property can be used to restrict the account rows written to the XML file.

The EAI Siebel Adapter queries using the input parameters and places the output into an output argument named Siebel Message (this is a Process Property), which will be used in the next workflow step. Siebel Message is a hierarchical data structure that is stored in the memory.

2 Write Employee to XML File.

After data about a particular account is obtained (in the Query Employee from Siebel step), it is written to an XML file. Another business service named EAI XML Write to File is used. The Write Siebel Message is used to write the Siebel Message output from the previous workflow step.

Now some discussion about the input arguments the above method takes. Notice that both are process properties. File Name could have been of Literal type (instead of being a process property), because this variable is not required by other workflow steps. But this argument has been made a process property so that it can be accessed from outside (for example, through another workflow—where this workflow is a subprocess; or by using code). In essence, you will be placing the name of the XML file in this property (that is, C:\EmployeeOutbound.xml).

The second argument is Siebel Message. The previous workflow step would have placed the account data into this process property. This workflow step accesses the exact same information (which is in a hierarchy/property set format) and converts it into XML format and writes it to a file.

This workflow step has no Output Arguments because once the data is written to an XML file, the job of this workflow process is complete.

Employee – Receive PeopleSoft Customer Workflow

The Position – Receive PeopleSoft Customer workflow, shown in Figure 32, involves reading employee information from an XML file and upserting the data into the Siebel application.

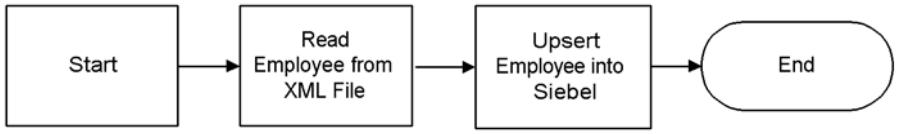


Figure 32. Employee – Receive PeopleSoft Customer Workflow

This workflow does the reverse of the workflow discussed in the previous section (Employee – Send PeopleSoft Employee). Here the goal is to read an XML file filled with employee(s) information, and upsert them into the Siebel database. Here the upsert stands for a combination of update and insert. Meaning, if the record already exists in Siebel database, then the record will be updated, using any changes supplied in the XML file. If no such record exists, then this record will be inserted as a brand new record. This determination is made by using the user keys in the integration object.

Process Properties

The Employee Inbound workflow has exactly the same set of process properties as Employee Outbound. Except for the process property XML File, all others come by default. The XML File process property is created. Of all these process properties, here, two have a higher significance—Siebel Message and XML File. Siebel Message is used to transfer the object related data between workflow steps, in a hierarchical format. And, XML File is the name and directory location of the XML file that contains account records to be loaded.

Workflow Steps

As shown in Figure 32 on page 139, this integration workflow involves the following sequential processes:

1 Read Employee from XML File.

At first, the Read Siebel Message method of the EAI XML Read from File business service is used to read data from an external XML file. This is a standard business service shipped with the Siebel software.

A value for a single parameter, XML location, is passed to this method. The XML data from the above file is converted into Siebel Message, and is kept in the output argument of the same name.

This data will then be read by the next workflow step.

2 Upsert Employee into Siebel.

Next, the Insert or Update method is used to write the extracted Employee data to the Siebel database. As the name suggests, this method inserts or updates the record according to the need.

All this needs is the Siebel Message output from the Read Employee from XML File workflow step. As discussed before, this step decides on whether to insert or update a record, based on the user keys.

Finally, how would this workflow know which Integration Object it has to go against? This is neither listed in the process properties nor in the input arguments. This information comes from the input XML file itself. See Example on page 141 below for details.

This workflow step has no Output Arguments because once the data is written to Siebel database, the job of this workflow process is complete.

Example

The following sample XML file will be used to load the prebuilt Employee – Receive PeopleSoft Employee workflow.

```
<?xml version="1.0" encoding="UTF-8" ?>
<?Siebel-Property-Set EscapeNames="false"?>
<SiebelMessage MessageId="EmployeeInboundSample"
MessageType="Integration Object"
IntObjectName="EAI Employee" IntObjectFormat="Siebel
Hierarchical">
<ListOfEmployee>
   <Employee>
   <Alias>JohnLee</Alias>
   <CellPhone>6504551234</CellPhone>
   <Emp>111</Emp>
   <EmployeeOrganization>Default Organization</
EmployeeOrganization>
   <Fax>6504551999</Fax>
   <FirstName>John</FirstName>
   <HireDate>1/7/2000</HireDate>
   <HomePhone>6504451111</HomePhone>
   <JobTitle>Administrator</JobTitle>
   <LastName>Lee</LastName>
   <LoginName>JLEE</LoginName>
   <LoginDomain>INTERNAL</LoginDomain>
   <MF>M</MF>
   <MaidenName>Chan</MaidenName>
   <MiddleName>Paul</MiddleName>
   <NextAnnualReviewDate>1/7/2002</NextAnnualReviewDate>
   <NickName>Johnny</NickName>
   <PagerPIN>PIN111</PagerPIN>
   <PagerPhone>6505559898</PagerPhone>
   <PagerType>Numeric</PagerType>
   <PartyUId>EMP1234</PartyUId>
   <PersonalTitle>Mr.</PersonalTitle>

   <Phone>6505531335</Phone>
   <ShareAddressFlag>Y</ShareAddressFlag>
   <ShareHomePhoneFlag>Y</ShareHomePhoneFlag>
   <TerminationDate>7/8/2001</TerminationDate>
   <ListOfEmployee_Position>
       <Employee_Position IsPrimaryMVG="Y">
       <EmployeeEndDate />
       <EmployeeStartDate>3/7/2001</EmployeeStartDate>
       <EmployeeStartDate>7/8/2001</EmployeeStartDate>
       <Position>Siebel Administrator</Position>
```

```
        <Division>Siebel Administration</Division>
      </Employee_Position>
        <Employee_Position IsPrimaryMVG="N">
        <EmployeeEndDate>3/6/2001</EmployeeEndDate>
        <EmployeeStartDate>1/7/2000</EmployeeStartDate>
        <Position>Compensation Administrator</Position>
        <Division>Siebel Sales</Division>
      </Employee_Position>
    </ListOfEmployee_Position>
    <ListOfEmployee_Responsibility>
       <Employee_Responsibility IsPrimaryMVG="Y">
       <Responsibility>ERM User</Responsibility>
      </Employee_Responsibility>
       <Employee_Responsibility IsPrimaryMVG="N">
       <Responsibility>ePortal User</Responsibility>
      </Employee_Responsibility>
    </ListOfEmployee_Responsibility>
    <ListOfEmployee_StartHomeAddress>
       <Employee_StartHomeAddress IsPrimaryMVG="Y">
       <StartHomeAddress>2000 College Ave,</StartHomeAddress>
       <StartHomeCity>San Mateo</StartHomeCity>
       <StartHomeCountry>USA</StartHomeCountry>
       <StartHomeState>CA</StartHomeState>
       <StartHomeZipcode>94401</StartHomeZipcode>
       <StartHomeAddressName>2000 College Ave, Los Angeles, CA
          94401</StartHomeAddressName>
      </Employee_StartHomeAddress>
       <Employee_StartHomeAddress IsPrimaryMVG="N">
       <StartHomeAddress>1000 Phoebe Ct.,</StartHomeAddress>
       <StartHomeCity>Foster City </StartHomeCity>
       <StartHomeCountry>USA</StartHomeCountry>
       <StartHomeState>CA</StartHomeState>
       <StartHomeZipcode>94404</StartHomeZipcode>
       <StartHomeAddressName>1000 Phoebe Ct., Foster City, CA
          94404</StartHomeAddressName>
      </Employee_StartHomeAddress>
    </ListOfEmployee_StartHomeAddress>
  </Employee>
  </ListOfEmployee>
  </SiebelMessage>
```

Prebuilt Integration Object DTDs A

This appendix presents the DTDs of the prebuilt integration objects.

The following sections present the DTDs for the EAI Employee, EAI Position, and EAI Account prebuilt integration objects.

EAI Employee

```
<!-- Copyright (C) 2001, Siebel Systems, L.P., All rights reserved.
-->

<!-- Siebel DTD Generation -->

<!ELEMENT SiebelMessage (ListOfEmployee+) >

   <!ATTLIST SiebelMessage MessageId CDATA #IMPLIED>

   <!ATTLIST SiebelMessage MessageType CDATA #IMPLIED>

   <!ATTLIST SiebelMessage IntObjectName CDATA #FIXED "EAI
Employee">

   <!ATTLIST SiebelMessage IntObjectFormat CDATA #IMPLIED>

<!ELEMENT ListOfEmployee (Employee+) >

<!ELEMENT Employee (Id?,

                    Alias?,

                    CellPhone?,

                    EMailAddr?,

                    Emp?,

                    EmployeeOrganization?,

                    EmployeeOrganizationId?,
```

```
Fax?,
FirstName?,
HireDate?,
HomePhone?,
IntegrationId?,
JobTitle?,
LastName?,
LoginDomain?,
LoginName?,
MF?,
MaidenName?,
MiddleName?,
NextAnnualReviewDate?,
NickName?,
PagerPIN?,
PagerPhone?,
PagerType?,
PartyUId?,
PersonalTitle?,
Phone?,
ShareAddressFlag?,
ShareHomePhoneFlag?,
StartHomeAddressId?,
TerminationDate?,
ListOfEmployee_Position?,
```

```
                    ListOfEmployee_ERMEmployeePeriodSummary?,
                    ListOfEmployee_Responsibility?,
                    ListOfEmployee_StartHomeAddress?)>
<!ELEMENT Id (#PCDATA)>
<!ELEMENT Alias (#PCDATA)>
<!ELEMENT CellPhone (#PCDATA)>
<!ELEMENT EMailAddr (#PCDATA)>
<!ELEMENT Emp (#PCDATA)>
<!ELEMENT EmployeeOrganization (#PCDATA)>
<!ELEMENT EmployeeOrganizationId (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT HireDate (#PCDATA)>
<!ELEMENT HomePhone (#PCDATA)>
<!ELEMENT IntegrationId (#PCDATA)>
<!ELEMENT JobTitle (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT LoginDomain (#PCDATA)>
<!ELEMENT LoginName (#PCDATA)>
<!ELEMENT MF (#PCDATA)>
<!ELEMENT MaidenName (#PCDATA)>
<!ELEMENT MiddleName (#PCDATA)>
<!ELEMENT NextAnnualReviewDate (#PCDATA)>
<!ELEMENT NickName (#PCDATA)>
<!ELEMENT PagerPIN (#PCDATA)>
```

```
<!ELEMENT PagerPhone (#PCDATA)>

<!ELEMENT PagerType (#PCDATA)>

<!ELEMENT PartyUId (#PCDATA)>

<!ELEMENT PersonalTitle (#PCDATA)>

<!ELEMENT Phone (#PCDATA)>

<!ELEMENT ShareAddressFlag (#PCDATA)>

<!ELEMENT ShareHomePhoneFlag (#PCDATA)>

<!ELEMENT StartHomeAddressId (#PCDATA)>

<!ELEMENT TerminationDate (#PCDATA)>

<!ELEMENT ListOfEmployee_Position (Employee_Position*)>

<!ELEMENT Employee_Position (Division?,

                    DivisionId?,

                    EmployeeEndDate?,

                    EmployeeStartDate?,

                    Organization?,

                    OrganizationId?,

                    Position?,

                    PositionId?,

                    PositionIntegrationId?,

                    DivisionIntegrationId?)>

<!ELEMENT ListOfEmployee_ERMEmployeePeriodSummary
(Employee_ERMEmployeePeriodSummary*)>

<!ELEMENT Employee_ERMEmployeePeriodSummary (PeriodSummaryEndDate?,

                    PeriodSummaryId?,

                    PeriodSummaryRollupPeriodType?,
```

```
                    PeriodSummaryScore?,
                    PeriodSummarySubmitToMgrFirstName?,
                    PeriodSummarySubmitToMgrId?,
                    PeriodSummarySubmitToMgrLastName?,
                    PeriodSummarySumbitToMgrIntegrationId?,
                    PeriodSummarySumbitToMgrLogin?,
                    PerformanceMeasure?)>
<!ELEMENT ListOfEmployee_Responsibility (Employee_Responsibility*)>
<!ELEMENT Employee_Responsibility (Responsibility?,
                    ResponsibilityId?)>
<!ELEMENT ListOfEmployee_StartHomeAddress
(Employee_StartHomeAddress*)>
<!ELEMENT Employee_StartHomeAddress (StartHomeAddress?,
                    StartHomeAddressAddressId?,
                    StartHomeAddressIntegrationId?,
                    StartHomeAddressName?,
                    StartHomeCity?,
                    StartHomeCountry?,
                    StartHomeState?,
                    StartHomeZipcode?)>
<!ELEMENT Division (#PCDATA)>
<!ELEMENT DivisionId (#PCDATA)>
<!ELEMENT EmployeeEndDate (#PCDATA)>
<!ELEMENT EmployeeStartDate (#PCDATA)>
<!ELEMENT Organization (#PCDATA)>
```

```
<!ELEMENT OrganizationId (#PCDATA)>
<!ELEMENT Position (#PCDATA)>
<!ELEMENT PositionId (#PCDATA)>
<!ELEMENT PositionIntegrationId (#PCDATA)>
<!ELEMENT DivisionIntegrationId (#PCDATA)>
<!ELEMENT PeriodSummaryEndDate (#PCDATA)>
<!ELEMENT PeriodSummaryId (#PCDATA)>
<!ELEMENT PeriodSummaryRollupPeriodType (#PCDATA)>
<!ELEMENT PeriodSummaryScore (#PCDATA)>
<!ELEMENT PeriodSummarySubmitToMgrFirstName (#PCDATA)>
<!ELEMENT PeriodSummarySubmitToMgrId (#PCDATA)>
<!ELEMENT PeriodSummarySubmitToMgrLastName (#PCDATA)>
<!ELEMENT PeriodSummarySumbitToMgrIntegrationId (#PCDATA)>
<!ELEMENT PeriodSummarySumbitToMgrLogin (#PCDATA)>
<!ELEMENT PerformanceMeasure (#PCDATA)>
<!ELEMENT Responsibility (#PCDATA)>
<!ELEMENT ResponsibilityId (#PCDATA)>
<!ELEMENT StartHomeAddress (#PCDATA)>
<!ELEMENT StartHomeAddressAddressId (#PCDATA)>
<!ELEMENT StartHomeAddressIntegrationId (#PCDATA)>
<!ELEMENT StartHomeAddressName (#PCDATA)>
<!ELEMENT StartHomeCity (#PCDATA)>
<!ELEMENT StartHomeCountry (#PCDATA)>
<!ELEMENT StartHomeState (#PCDATA)>
<!ELEMENT StartHomeZipcode (#PCDATA)>
```

EAI Position

```
<!-- Copyright (C) 2001, Siebel Systems, L.P., All rights reserved.
-->

<!-- Siebel DTD Generation -->

<!ELEMENT SiebelMessage (ListOfPosition+) >

   <!ATTLIST SiebelMessage MessageId CDATA #IMPLIED>

   <!ATTLIST SiebelMessage MessageType CDATA #IMPLIED>

   <!ATTLIST SiebelMessage IntObjectName CDATA #FIXED "EAI
Position">

   <!ATTLIST SiebelMessage IntObjectFormat CDATA #IMPLIED>

<!ELEMENT ListOfPosition (Position+) >

<!ELEMENT Position (Id?,
                    BillingProduct?,
                    BillingProductId?,
                    BillingProductIntegrationId?,
                    Compensatable?,
                    Description?,
                    Division?,
                    DivisionId?,
                    DivisionIntegrationId?,
                    IntegrationId?,
                    Name?,
                    Organization?,
                    OrganizationId?,
                    OrganizationIntegrationId?,
                    ParentPositionId?,
```

```
                    ParentPositionIntegrationId?,
                    ParentPositionName?,
                    PositionType?,
                    PrimaryPositionAddressId?,
                    PrimaryPositionAddressIntegrationId?)>
<!ELEMENT Id (#PCDATA)>
<!ELEMENT BillingProduct (#PCDATA)>
<!ELEMENT BillingProductId (#PCDATA)>
<!ELEMENT BillingProductIntegrationId (#PCDATA)>
<!ELEMENT Compensatable (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Division (#PCDATA)>
<!ELEMENT DivisionId (#PCDATA)>
<!ELEMENT DivisionIntegrationId (#PCDATA)>
<!ELEMENT IntegrationId (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Organization (#PCDATA)>
<!ELEMENT OrganizationId (#PCDATA)>
<!ELEMENT OrganizationIntegrationId (#PCDATA)>
<!ELEMENT ParentPositionId (#PCDATA)>
<!ELEMENT ParentPositionIntegrationId (#PCDATA)>
<!ELEMENT ParentPositionName (#PCDATA)>
<!ELEMENT PositionType (#PCDATA)>
<!ELEMENT PrimaryPositionAddressId (#PCDATA)>
<!ELEMENT PrimaryPositionAddressIntegrationId (#PCDATA)>
```

EAI Account

```
<!-- Copyright (C) 2001, Siebel Systems, L.P., All rights reserved.
-->

<!-- Siebel DTD Generation -->

<!ELEMENT SiebelMessage (ListOfAccount+) >

   <!ATTLIST SiebelMessage MessageId CDATA #IMPLIED>

   <!ATTLIST SiebelMessage MessageType CDATA #IMPLIED>

   <!ATTLIST SiebelMessage IntObjectName CDATA #FIXED "EAI Account">

   <!ATTLIST SiebelMessage IntObjectFormat CDATA #IMPLIED>

<!-- Shared Element List.  These elements are guarenteed -->

<!-- to have the same datatype, length, precision, and scale.-->

<!ELEMENT OrganizationId (#PCDATA) >

<!ELEMENT Organization (#PCDATA) >

<!ELEMENT IntegrationId (#PCDATA) >

<!ELEMENT Id (#PCDATA) >

<!ELEMENT EmailAddress (#PCDATA) >

<!ELEMENT ListOfAccount (Account+) >

<!ELEMENT Account (Id?,

                   AccountStatus?,

                   Alias?,

                   CSN?,

                   Competitor?,

                   CurrencyCode?,

                   DUNSNumber?,

                   Description?,
```

```
Division?,
EAISyncDate?,
EAISyncErrorText?,
EAISyncStatusCode?,
FreightTerms?,
FreightTermsInfo?,
GSAFlag?,
HomePage?,
IntegrationId?,
LanguageCode?,
LastManagerReviewDate?,
Location?,
MainFaxNumber?,
MainPhoneNumber?,
ManagersReview?,
Name?,
PartnerFlag?,
PriceList?,
PriceListId?,
PriceListIntegrationId?,
ProspectFlag?,
Region?,
Type?,
ListOfAccount_BusinessAddress?,
ListOfContact?,
```

```
                    ListOfAccount_Position?,
                    ListOfAccount_Organization?)>
<!ELEMENT AccountStatus (#PCDATA)>
<!ELEMENT Alias (#PCDATA)>
<!ELEMENT CSN (#PCDATA)>
<!ELEMENT Competitor (#PCDATA)>
<!ELEMENT CurrencyCode (#PCDATA)>
<!ELEMENT DUNSNumber (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Division (#PCDATA)>
<!ELEMENT EAISyncDate (#PCDATA)>
<!ELEMENT EAISyncErrorText (#PCDATA)>
<!ELEMENT EAISyncStatusCode (#PCDATA)>
<!ELEMENT FreightTerms (#PCDATA)>
<!ELEMENT FreightTermsInfo (#PCDATA)>
<!ELEMENT GSAFlag (#PCDATA)>
<!ELEMENT HomePage (#PCDATA)>
<!ELEMENT LanguageCode (#PCDATA)>
<!ELEMENT LastManagerReviewDate (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT MainFaxNumber (#PCDATA)>
<!ELEMENT MainPhoneNumber (#PCDATA)>
<!ELEMENT ManagersReview (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT PartnerFlag (#PCDATA)>
```

```
<!ELEMENT PriceList (#PCDATA)>
<!ELEMENT PriceListId (#PCDATA)>
<!ELEMENT PriceListIntegrationId (#PCDATA)>
<!ELEMENT ProspectFlag (#PCDATA)>
<!ELEMENT Region (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT ListOfAccount_BusinessAddress (Account_BusinessAddress*)>
<!ELEMENT Account_BusinessAddress (AddressActiveStatus?,
                    AddressId?,
                    AddressIntegrationId?,
                    BillAddressFlag?,
                    City?,
                    Country?,
                    County?,
                    EmailAddress?,
                    FaxNumber?,
                    MainAddressFlag?,
                    PhoneNumber?,
                    PostalCode?,
                    Province?,
                    ShipAddressFlag?,
                    State?,
                    StreetAddress?,
                    StreetAddress2?)>
<!ELEMENT ListOfContact (Contact*)>
```

```
<!ELEMENT Contact (Id?,
                   ActiveStatus?,
                   AlternatePhone?,
                   AssistantPhone?,
                   CellularPhone?,
                   ContactOrganizationIntegrationId?,
                   EmailAddress?,
                   FaxPhone?,
                   FirstName?,
                   HomePhone?,
                   IntegrationId?,
                   JobTitle?,
                   LastName?,
                   MF?,
                   MM?,
                   MiddleName?,
                   Organization?,
                   OrganizationId?,
                   PreferredCommunications?,
                   PreferredLanguageCode?,
                   WorkPhone?)>
<!ELEMENT ListOfAccount_Position (Account_Position*)>
<!ELEMENT Account_Position (Position?,
                   PositionId?,
                   PositionIntegrationId?)>
```

```
<!ELEMENT ListOfAccount_Organization (Account_Organization*)>
<!ELEMENT Account_Organization (Organization?,
                    OrganizationId?,
                    OrganizationIntegrationId?)>
<!ELEMENT AddressActiveStatus (#PCDATA)>
<!ELEMENT AddressId (#PCDATA)>
<!ELEMENT AddressIntegrationId (#PCDATA)>
<!ELEMENT BillAddressFlag (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT County (#PCDATA)>
<!ELEMENT FaxNumber (#PCDATA)>
<!ELEMENT MainAddressFlag (#PCDATA)>
<!ELEMENT PhoneNumber (#PCDATA)>
<!ELEMENT PostalCode (#PCDATA)>
<!ELEMENT Province (#PCDATA)>
<!ELEMENT ShipAddressFlag (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT StreetAddress (#PCDATA)>
<!ELEMENT StreetAddress2 (#PCDATA)>
<!ELEMENT ActiveStatus (#PCDATA)>
<!ELEMENT AlternatePhone (#PCDATA)>
<!ELEMENT AssistantPhone (#PCDATA)>
<!ELEMENT CellularPhone (#PCDATA)>
<!ELEMENT ContactOrganizationIntegrationId (#PCDATA)>
```

```
<!ELEMENT FaxPhone (#PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT HomePhone (#PCDATA)>
<!ELEMENT JobTitle (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT MF (#PCDATA)>
<!ELEMENT MM (#PCDATA)>
<!ELEMENT MiddleName (#PCDATA)>
<!ELEMENT PreferredCommunications (#PCDATA)>
<!ELEMENT PreferredLanguageCode (#PCDATA)>
<!ELEMENT WorkPhone (#PCDATA)>
<!ELEMENT Position (#PCDATA)>
<!ELEMENT PositionId (#PCDATA)>
<!ELEMENT PositionIntegrationId (#PCDATA)>
<!ELEMENT OrganizationIntegrationId (#PCDATA)>
```

Index