



**INTEGRATION PLATFORM
TECHNOLOGIES:
SIEBEL eBUSINESS APPLICATION
INTEGRATION VOLUME II
MIDMARKET EDITION
VERSION 7.5**

12-BCID6W

SEPTEMBER 2002

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

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

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

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

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

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

Proprietary Information

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

Contents

Introduction

How This Guide Is Organized	10
Additional Resources	11
Revision History	11

Chapter 1. About Integration Objects

Integration Objects Terminology	14
Siebel Integration Objects	15
Integration Object and Integration Object Instance	16
Siebel Integration Object Wizards	17
Structure of Siebel Integration Objects	19
Associations	22
Multi-Value Groups	23
Pick Lists	29
Calculated Fields	29
Inner Joins	29
Field Dependencies	30
Primaries	30
Integration Component Keys	31
User Keys	31
Status Keys	37
Hierarchy Parent Key	37
Hierarchy Root Key	38

Chapter 2. Creating and Maintaining Integration Objects

Integration Object Builder Overview	40
Creating an Integration Object Using the EAI Siebel Wizard	41
Siebel Integration Object Fine-Tuning	44
Integration Object Validation	44
Integration Objects Synchronization	45
Synchronization Considerations	45
Synchronization Rules	49
The EAI Siebel Wizard	54
Siebel Integration Objects Maintenance and Upgrade	56
Permission Rules for Integration Components	56
EAI Siebel Adapter Access Control	58
Integration Object User Properties	59
Generating Schemas	63
Performance Considerations	64
Size of Integration Object	64
Force-Active Fields	64
Picklist Validation	64
Business Component Restrictions	65
System Fields	65
Best Practices	66

Chapter 3. Business Services

Overview of Business Services	68
Creating Business Services	68
Business Service Structure	69
About Property Sets	70
Creating Business Services in Siebel Tools	72
Defining a Business Service in Siebel Tools	73
Defining Business Service Methods	74

Defining Business Service Method Arguments	74
Defining and Writing Business Service Scripts	75
Specifying Business Service Subsystems	76
Defining Business Service User Properties	77
Creating a Business Service in the Siebel Client	78
Business Service Export and Import	79
Testing Your Business Service	80
Example	81
Code Sample	81

Chapter 4. Web Services

Overview	84
How the Web Services Dispatch Works	84
Publishing Siebel Web Services	85
Administration	85
WSDL Document Generation	88
Consuming External Web Services	89
WSDL Import Wizard	89
Administration	90
Examples	93
Invoking an External Web Service Using Workflow or Scripting	94
Invoking a Siebel Web Service Using an External Application	96

Chapter 5. The EAI Siebel Adapter

EAI Siebel Adapter Overview	100
EAI Siebel Adapter Methods	101
QueryPage Method	104
Query Method	104
Synchronize Method	105
Upsert Method	106

Update Method	106
Insert Method	106
Delete Method	107
Execute Method	107
XML Examples	110
MVGs in the EAI Siebel Adapter	112
Search Specification	114
EAI Siebel Adapter Concurrency Control	115
Modification Key	115
Modification IDs	116
Language-Independent Code	117
Siebel eAI and Run Time Events	118

Chapter 6. Siebel eAI and File Attachments

Overview	120
Exchange of Attachments with External Applications	121
Using MIME Messages to Exchange Attachments	121
Creating the Integration Object	122
Creating Workflow Processes Examples	123
The EAI MIME Hierarchy Converter	128
Outbound Integration	128
Inbound Integration	129
The EAI MIME Doc Converter	130
EAI MIME Doc Converter Properties	131

Chapter 7. Siebel Virtual Business Components

Overview of Virtual Business Components	134
Enhancements to VBCs for This Version	135
Usage and Restrictions	136
Virtual Business Components	137
Creating a New Virtual Business Component	137
Setting User Properties for the Virtual Business Component	138

XML Gateway Service	140
XML Gateway Methods	142
XML Gateway Method Arguments	142
Examples of Outgoing XML Format	143
Examples of Incoming XML Format	147
External Application Setup	149
Custom Business Service Methods	150
Common Method Parameters	151
Business Services Methods and Their Property Sets	152
Custom Business Service Example	168

Appendix A. Predefined EAI Business Services

Predefined EAI Business Services	172
--	-----

Appendix B. Property Set Representation of Integration Objects

Property Sets and Integration Objects	176
Property Set Node Types	177
Example of a Sample Account	179

Appendix C. DTDs for XML Gateway Business Service

Outbound DTDs	184
Delete	184
Init	184
Insert	184
PreInsert	185
Query	185
Update	186
Inbound DTDs	187
Delete Response	187
Init Response	187

Insert Response	187
PreInsert Response	187
Query Response	188
Update Response	188

Index

Introduction

This guide explains (the details of) Siebel eAI’s integration platform technologies, including integration objects, business services, the EAI Siebel Adapter, Virtual Business Components, and so on.

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

The audience for this guide consists primarily of employees in these categories:

Business Analysts	Persons responsible for analyzing application integration challenges and planning integration solutions at an enterprise.
Database Administrators	Persons who administer the database system, including data loading, system monitoring, backup and recovery, space allocation and sizing, and user account management.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel Integration Developers	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution for Siebel applications at an enterprise.

- | | |
|-------------------------------------|---|
| Siebel System Administrators | Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications. |
| System Integrators | Persons responsible for analyzing a business situation or using an analysis to build integration solutions or to develop custom solutions for specific applications at an enterprise. |

The audience for this book also needs to have experience in data integration, data transformation (data mapping), scripting or programming, and XML.

How This Guide Is Organized

This book is organized in a way that presents the most important information up front—specifically, how to create and maintain Integration Objects—followed by chapters describing individual components of the eAI integration platform, such as Virtual Business Components.

This book is Volume 2 of a six-volume set. The full set includes:

- *Overview: Siebel eBusiness Application Integration Volume I, MidMarket Edition*
- *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition*
- *Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition*
- *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV, MidMarket Edition*
- *XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition*
- *Application Services Interface Reference: Siebel eBusiness Application Integration Volume VI, MidMarket Edition*

Additional Resources

The product documentation set for Siebel eBusiness Applications is provided on the *Siebel Bookshelf* or in *Siebel Online Help*. The following integration related books and online help describe all the tools required to implement integration:

Siebel Tools Online Help, MidMarket Edition

Siebel Tools Reference, MidMarket Edition

Siebel Business Process Designer Administration Guide, MidMarket Edition

Siebel Enterprise Integration Manager Administration Guide, MidMarket Edition if you perform bulk loading or unloading of data.

Revision History

Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition, Version 7.5

About Integration Objects

1

This chapter describes the structure of Siebel integration objects. It describes the Integration Object Builder which assists you in building your own integration objects based on Siebel objects.

Integration Objects Terminology

This chapter describes the concepts that are often referred to using different terminology from one system to another. This section has been included to clarify the information in this chapter by providing a standard terminology for these concepts.

Table 1. Terminology

Term	Description
Metadata	Data that describes data. For example, the term datatype describes data elements such as char, int, Boolean, time, date, and float.
Siebel business object	A Siebel object type that creates a logical business model using links to tie together a set of interrelated business components. The links provide the one-to-many relationships that govern how the business components interrelate in this business object.
Component	A constituent part of any generic object.
Siebel business component	A Siebel object type that defines a logical representation of columns in one or more database tables. A business component collects columns from the business component's base table, its extension tables, and its joined tables into a single structure. Business components provide a layer of abstraction over tables. Applets in Siebel applications reference business components; they do not directly reference the underlying tables.
Field	A generic reference to a data structure that can contain one data element.
Siebel integration component field	A data structure that can contain one data element in a Siebel integration component.
Siebel integration component	A constituent part of a Siebel integration object.
Integration object	An integration object of any type, including the Siebel integration object, the SAP BAPI integration object, and the SAP IDOC integration objects.
Integration object instance	Actual data, usually the result of a query or other operation, which is passed from one business service to another, that is structurally modeled on a Siebel integration object.
Siebel integration object	An object stored in the Siebel repository that represents some Siebel business object.
Integration message	A bundle of data consisting of two major parts: header information that describes what should be done with or to the message itself, and instances of integration objects, that is, data in the structure of the integration object.

Siebel Integration Objects

Siebel integration objects allow you to represent integration metadata for Siebel business objects, XML, SAP IDOCs, and SAP BAPIs as common structures that the eAI infrastructure can understand. Because these integration objects adhere to a set of structural conventions, they can be traversed and transformed programmatically, using Siebel eScript objects, methods, and functions or transformed declaratively using Siebel Data Mapper.

NOTE: For more information, see *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV, MidMarket Edition*.

The typical integration project involves transporting data from one application to another. For example, you may want to synchronize data from a back-office system with the data in your Siebel application. You may want to generate a quote in the Siebel application and perform a query against your Enterprise Resource Planning (ERP) system transparently. In the context of Siebel eAI, data is transported in the form of an *integration message*. A message, in this context, typically consists of header data that identifies the message type and structure, and a body that contains one or more instances of data—for example, orders, accounts, or employee records.

When planning your integration project, you should consider several issues:

- How much data transformation will your message require?
- At what point in the process will you perform the data transformation?
- Will you require a confirmation message response to the sender?
- Are there data items in the originating data source that should not be replicated in the receiving data source, or that should replace existing data in the receiving data source?

This guide can help you understand how Siebel eAI represents the Siebel business object structure. It also provides descriptions of how Siebel eAI represents external SAP R/3 structures.

Integration Object and Integration Object Instance

Understanding the difference between integration objects and integration object instances is important, especially in regard to the way they are discussed in this chapter.

An integration object, in the context of Siebel eAI, is metadata; that is, it is a generalized representation or model of a particular set of data. It is a schema of a particular thing.

An Integration Object Instance is also referred to as a Siebel Message object.

An integration object instance is actual data organized in the format or structure of the integration object. [Figure 1](#) illustrates a simple example of an integration object and an integration object instance, using partial data.

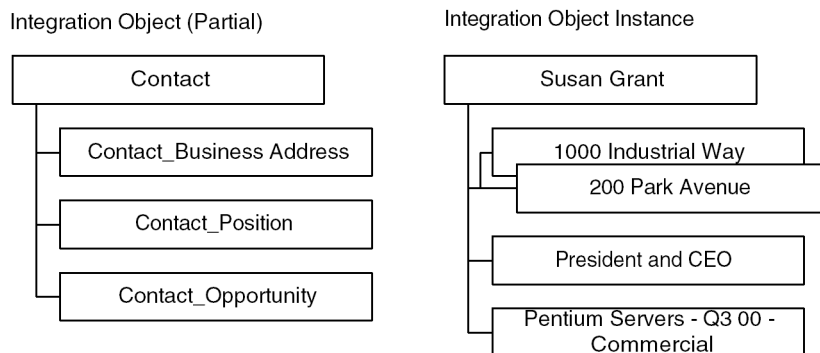


Figure 1. Integration Object and Integration Object Instance

Any discussion of integration objects in this book will include clarifying terms—for example, metadata or Siebel instance—to help make the distinction.

Siebel Integration Object Wizards

Within Siebel Tools, there are multiple wizards associated with integration objects: one that creates integration objects for internal use by the Siebel application, and others that create integration objects for external systems based on Siebel objects. [Figure 2](#) shows the logic of two of these wizards. The Generate Code wizard (not shown) works in the same manner as the Generate Schema wizard, but it generates Java classes.

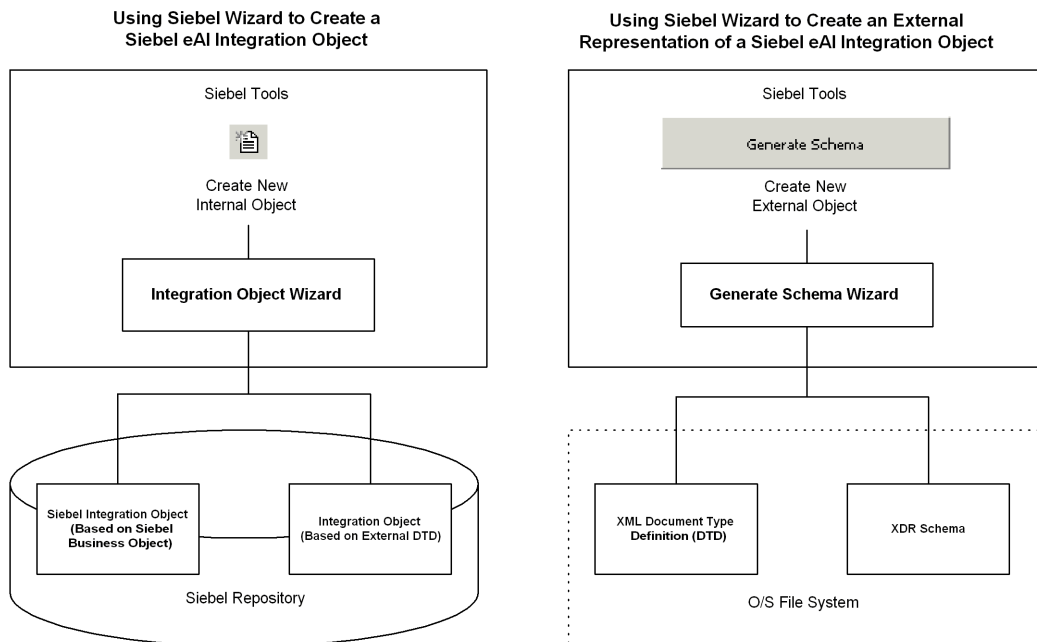


Figure 2. Integration Object Wizards

- **Integration Object Builder wizard.** This wizard lets you create a new object. It supplies the functionality for creating integration objects from Siebel business objects or integration objects based on representations of external business objects using XML Schema Definition (XSD) or Document Type Definition (DTD). To access this wizard, navigate to the New Object dialog box in Siebel Tools and after selecting the EAI tab, double-click the Integration Object icon to start the Integration Object Builder wizard.
- **Generate XML Schema wizard.** This wizard lets you choose an integration object and output XML schema—in XML Schema Definition (XSD) standard, Document Type Definition (DTD) or Microsoft's XDR (XML Data Reduced) format. To access this wizard, navigate to the Integration Objects list in Siebel Tools and select an integration object. Then click Generate Schema button to start the Generate XML Schema wizard.
- **Code Generator wizard.** The third wizard lets you create a set of Java class files based on any available integration object or Siebel business service. To access this wizard, navigate to the Integration Objects list in Siebel Tools and select an integration object. Then click the Generate Code button to start the Code Generator wizard.

NOTE: Specific instructions on how to use these wizards appear throughout the Siebel eBusiness Application Integration documentation set where appropriate.

Structure of Siebel Integration Objects

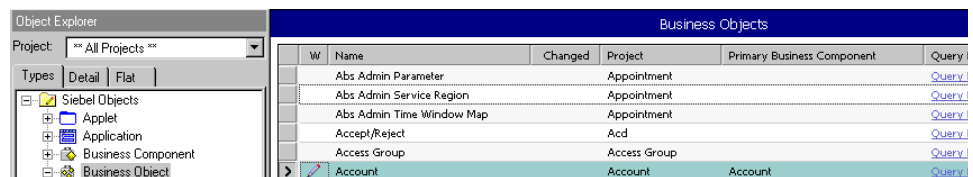
The Siebel integration object provides a hierarchical structure that represents a complex data type. Most specifically, prebuilt eAI integration objects describe the structure of Siebel business objects, SAP IDOCs, SAP BAPIs, XML, and external data. Most integration projects require the use of an integration object that describes Siebel business objects, either in an outbound direction—a *query* operation against a Siebel integration object—or an inbound direction—a *synchronize* operation against a Siebel integration object.

This chapter continues with descriptions of how to create integration objects. The initial process—using the Integration Object Builder wizard—is essentially the same for every integration object type currently supported.

Caution: You should avoid using or modifying integration objects in the EAI Design project. Using or modifying any objects in the EAI Design project can cause unpredictable results.

Siebel business objects conform to a particular structure in memory. Although it is generally not necessary to consider this structure when working with Siebel applications, when you are planning and designing an integration project it is helpful to understand how a Siebel eAI integration object represents that internal structure.

An integration object consists of one Parent Integration Component—sometimes referred to as the root component or the primary integration component. The Parent Integration Component corresponds to the primary business component of the business object you chose as the model for your integration object. [Figure 3](#) shows the Account business object in Siebel Tools.



W	Name	Changed	Project	Primary Business Component	Query
	Abs Admin Parameter		Appointment		Query
	Abs Admin Service Region		Appointment		Query
	Abs Admin Time Window Map		Appointment		Query
	Accept/Reject		Acid		Query
	Access Group		Access Group		Query
	Account		Account	Account	Query

Figure 3. Account Parent Business Component

For example, assume you chose the Account business object (on the first panel of the Integration Object Builder wizard) to base your integration object *myAccount_01* on. The Account business object in Siebel Tools has an Account business component as its primary business component. In the *myAccount_01* integration object, every child component will be represented as either a direct or indirect child of the primary business component named Account.

Each child component can have one or more child components. In Siebel Tools, if you look at the integration components for an integration object you have created, you will see that each component can have one or more fields. [Figure 4 on page 21](#) illustrates a partial view of a Siebel integration object based on the Account business object, with the Business Address component and the Contact component activated.

The structure, shown in [Figure 4](#), represents part of an integration object. The Account parent integration component can have both fields and child integration components. Each integration component can also have child integration components and fields. A structure of this sort represents the *metadata* of an Account integration object. You may choose to inactivate components and fields. By inactivating components and fields, you can define the structure of the integration object instances entering or leaving the system.

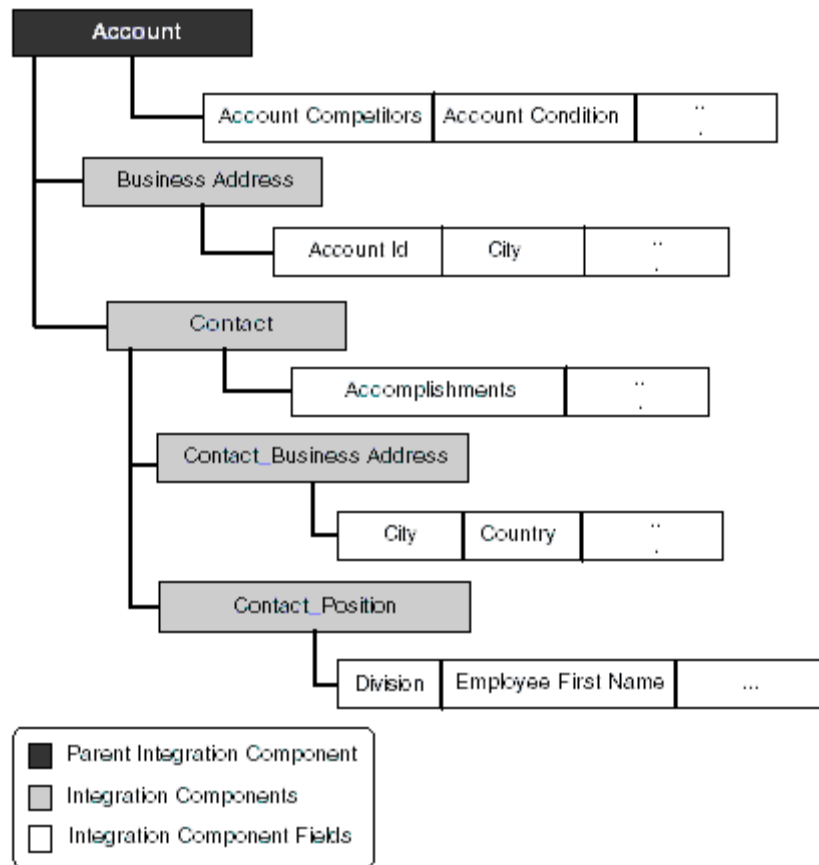


Figure 4. Representation of Partial Account Integration Object

Associations

Siebel business objects are made up of business components that are connected by a *link*. An *association* is a business component that represents the intersection table that contains these links. The integration component definition of associations is similar to that of multi-value groups (MVGs). User properties *Association* and *MVGAssociation* on the integration component denote that the corresponding business component is an associated business component or an associated MVG, respectively. For fields that are defined on MVG associations, *External Name* denotes the name of the business component field as it appears on the parent business component, and the user property *AssocFieldName* denotes the name of the business component field as it appears on the MVG business component.

For example, the Contact business object is partly made up of the Contact and Opportunity business components. The association between these two business components is represented by the Contact/Opportunity link with a value—that is, a table name—in the Inter Table column. The Integration Object Builder wizard creates a new integration component for the integration object based on the Contact business object that represents the association. This integration component—shown in [Figure 5](#) as the Opportunity integration component—has one user property defined: the *Association* user property, set to a value of Y.

Integration Components						
	W	External Name Context	Name	Changed	Parent Integration Component	External Name
		Contact	Contact	✓		Contact
>		Opportunity	Opportunity	✓	Contact	Opportunity
<div><div></div><div></div></div>						
Integration Component User Props						
	W	Name	Changed	Value	Inactive	Comments
>		Association	✓	Y		

Figure 5. Integration Component Representation of Association

NOTE: If an integration component is an association based on an intersection table, the user key for this integration component cannot contain fields, based directly or indirectly, on the same association-intersection table.

Multi-Value Groups

Multi-value groups (MVGs) are used within Siebel business components to represent database multivalued attributes. MVGs can be one of two types: regular MVGs or MVG Associations.

An integration object instance most often has multiple integration component instances. For example, an Account can have multiple Business Addresses but only one of these addresses is marked as the primary address. A business requirement may require that only the integration component instance that corresponds to the primary MVG be part of the integration object instance. In relation to Account and Business Addresses this means that only the primary address should be part of the Account integration object instance. The primary address can be obtained by one of the following steps:

- Creating a new MVG on the Account business component that uses a link with a search specification only returning the primary address record.
- Exposing the primary address information on the Account business component level using a join that has the primary Id as source field. Note that in this case the primary Address information corresponds to fields on the Account integration component instance and not the fields on a separate Address component instance.

In Siebel Tools, if a Siebel business component contains an MVG, the MVG will be represented in several screens as illustrated in the following sections.

About Integration Objects

Structure of Siebel Integration Objects

Screen 1: Fields View

For example, the Account business component contains a multi-value group field, the Address Id, as illustrated in [Figure 6](#).

Business Components					
W	Name	Changed	Project	Cache Data	Class
	Access Control Party Reporting		System		CSSBusComp
	Access Control Test		System		CSSBusComp
	Access Group		Access Group		CSSBCGroup
	Access Group Member		Access Group		CSSBCBase
>	Account		Account		CSSBCBase
<					
Fields					
Required	W	Name	Changed	Dest Field	Multi Value Link
		Address Active Status		Active Status	Business Address
>		Address Id		Id	Business Address
<					

Figure 6. Address Id MVG Field in the Account Business Component

Screen 2: Multi-Value Links

As shown in [Figure 7](#), the multi-value link property has the value Business Address. If you navigate to the Multi Value Link screen, you see that the Business Address multi-value link has the value Business Address as its Destination Business Component.

Business Components							
W	Name	Changed	Project	Cache Data	Class	Data Source	Dirty Reads
	Access Control Party Reporting Relation		System		CSSBusComp		✓
	Access Control Test		System		CSSBusComp		✓
	Access Group		Access Group		CSSBCGroup		✓
	Access Group Member		Access Group		CSSBCBase		✓
>	Account		Account		CSSBCBase		✓
<							
Multi Value Links							
W	Name	Auto Primary	Primary Id Field	Destination Business Component	Destination Link		
	Account Category	Default	Primary Category Id	Account Category	Account/Account Category		
	Account Credit Profile	Default		Account Credit Profile	Account/Account Credit Profile		
	Account Synonym	Default	Primary Synonym Id	Account Synonym	Account/Account Synonym		
	Bill To Business Address	Selected	Primary Bill To Address Id	Business Address	Account/Business Address		
	Bill To Contact	Selected	Primary Bill To Person Id	Contact	Account/Contact		
>	Business Address	Default	Primary Address Id	Business Address	Account/Business Address		
<							

Figure 7. Destination Business Component

Screen 3: Fields View

As shown in [Figure 8](#), the Business Address multi-value link has Business Address as its Destination Business Component. This means that there is another business component named Business Address that contains the fields that are collectively represented by Address Id in the Account business component.

Business Components				
	W	Name	Project	Table
>		Business Address	Contact	S_ADDR_ORG
		Business Service	Business Service	S_RT_SVC
		Business Service Input Argument Properties	Business Service	
		Business Service Method	Business Service	S_RT_SVC METH
		Business Service Method Arg	Business Service	S_RT_SVC M_ARG
<				

Fields				
Required	W	Name	Changed	Dest Field
		Account Id		OU_ID
		Active Status		ACTIVE_FLG
>		Address Name		ADDR_NAME
		Address Name Locked		NAME_LOCK_FLG

Figure 8. Business Address Business Component

Graphical Representation

Figure 9 shows a graphical way to represent this relationship.

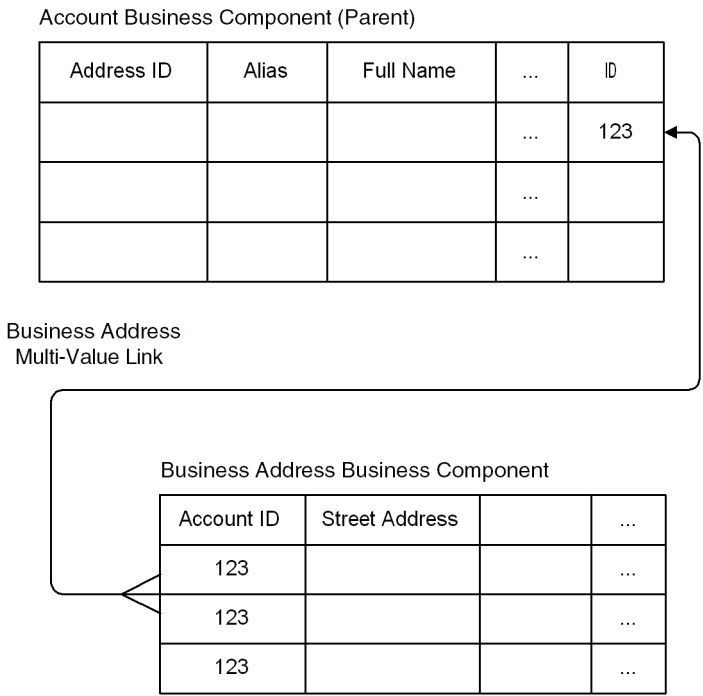


Figure 9. Address ID Field and Business Address MVG

The more table-like representation above shows how the Business Address multi-value link connects the two business components. The child points to the Business Address business component, which contains the multiple fields that make up the MVG.

NOTE: Two business components are used to represent an MVG.

Creating an Integration Component

To create a Siebel integration component to represent an MVG, it is necessary also to create two integration components:

- The first integration component represents the parent business component. In the example, this is the Account business component. This integration component contains only the fields that are defined in the parent business component, but which are not based on MVGs. The Multi-value Link property and the Multi-value property are empty for these fields.
- The second integration component represents the MVG business component. In the example, this is the Business Address business component. The second integration component has one integration field for each field based on the given MVG in the parent business component. An integration component user property will be set on this integration component to tell the EAI Siebel Adapter that it is based on an MVG business component. If the MVG is a regular MVG, the user property is named *MVG*. If the MVG is an Association MVG, then the user property is named *MVGAssociation*. In both cases, the value of the user property is *Y*.

Figure 10 shows an integration component based on an MVG and its user property value in Siebel Tools.

Integration Components			
W	External Name Context	Name	Parent Integration
	Account_Bill To Contact	Account_Bill To Contact	Account
	Account_Business Address	Account_Business Address	Account
	Account_Industry	Account_Industry	Account
	Account_Organization	Account_Organization	Account
	Account_Organization Unit Type	Account_Organization Unit Type	Account
Integration Component User Props			
W	Name	Changed	Value
	MVG	✓	Y

Figure 10. Integration Component Based on MVG Business Component

The EAI Siebel Adapter needs to know the names of the MVG fields as they are defined in the parent business component—in this example, Account—and also the names of the MVG fields as they are known in the business component that represents the MVG—in this example, Account Business Address. As shown in Figure 11, the integration component fields represent the MVG.

Integration Components					
	W	External Name Context	Name	Parent Integration	
		Account_Bill To Contact	Account_Bill To Contact	Account	
		Account_Business Address	Account_Business Address	Account	
		Account_Industry	Account_Industry	Account	
		Account_Organization	Account_Organization	Account	
		Account_Organization Unit Type	Account_Organization Unit Type	Account	

Integration Component Fields					
Inactive	W	Name	Changed	Data Type	Length
		Address Active Status	✓	DTYPE_TEXT	
		Address Id	✓	DTYPE_ID	30
		Address Integration Id	✓	DTYPE_TEXT	30
		Address Name	✓	DTYPE_TEXT	100
		Bill Address Flag	✓	DTYPE_TEXT	

Figure 11. Integration Component Fields Representing MVG

To represent both names, each field is assigned an integration component field user property that contains the entry *MVGFieldName*—or *AssocFieldName* if the user property is *MVGAssoc*. Its value is the name of the field shown in the parent business component—in this example, *Business Address*

Pick Lists

If an integration component field is created for a Siebel business component field and the business component field is based on a picklist, validation of the field can be done either in the EAI Siebel Adapter, or Object Manager. To have the validation done in the EAI Siebel Adapter, the integration component field should have a user property with the name PICKLIST and a value of Y. In all other cases, validation will be done by Object Manager.

When the EAI Siebel Adapter is performing the validation, it can be explicitly instructed which fields to use as a search criteria, using the 'PicklistUserKey' user property. The value of the 'PicklistUserKey' user property is a comma separated list of integration component fields that are used to find the matching record in the picklist (for example, 'Account, Account Location').

NOTE: Picklist validation in the EAI Siebel Adapter is required for dynamic picklists.

Calculated Fields

Calculated fields are inactive in the Integration Object when they are created. If your business needs require it, you need to activate the calculated fields in the integration object.

NOTE: Calculated Fields are those Integration Component Fields that have the Calculated flag checked on the corresponding Business Component Field.

Inner Joins

When inner joins are used, records for which the inner joined field is not set are not returned in any query. By default the wizard inactivates such fields. If your business needs require it, you need to activate them.

NOTE: Activating inner join may cause a query on that integration component not to find existing rows.

Field Dependencies

Some fields in a business component may depend on other fields. In most cases, such dependencies are due to Pick Map constraints and the EAI Siebel Adapter can automatically handle them. However, in some cases there may be other dependencies. Set the user property `FieldDependency <suffix>` on fields to specify fields that these fields depend on. Use different suffixes to specify multiple dependencies.

Primaries

Primaries are set through multi-value links. However, you should not use multi-value links for modifying the linked component. To modify the linked component you should use links. If you need to set primaries in addition to modifying the linked component, use both links and multi-value links in your integration object. The EAI Siebel Adapter should use the multi-value link only after it processes the component through the link; therefore, the link or the Association component should have a smaller external sequence number than the related MVG or MVGAssociation component. See [“Structure of Siebel Integration Objects” on page 19](#) for an example.

Integration Component Keys

There are multiple types of integration component keys.

User Key. See [“User Keys.”](#)

Status Key. See [“Status Keys” on page 37.](#)

Hierarchy Parent Key. See [“Hierarchy Parent Key” on page 37.](#)

Hierarchy Root Key. See [“Hierarchy Root Key” on page 38.](#)

Modification Key. See [“EAI Siebel Adapter Concurrency Control” on page 115.](#)

NOTE: There should be just one integration component key for every types of keys except the user key. For example, if there are two Hierarchy Parent Keys defined for an integration component, the EAI Siebel Adapter picks the first one and ignores the second one.

User Keys

User Key is a group of fields whose values must uniquely identify a Siebel business component record. During inbound integration, User Keys are used to determine whether the incoming data updates an existing record or inserts a new one. The Integration Object wizard automatically creates some User Keys based on characteristics discussed below. You should make sure that the generated User Keys match your business requirements; otherwise, inactivate generated keys or add new User Keys as appropriate.

NOTE: Evaluation of which User Key will be used is done only once, when the first integration component instance is used. For subsequent instances of the integration component within a message, the same User Key is used.

Integration component keys are built by the Integration Object Builder wizard based on values in the underlying table of the business component upon which the integration component is based.

NOTE: Integration objects that represent Siebel business objects, and that will be used in insert, update, synchronize, or execute operations, must have at least one user key defined for each integration component.

In Siebel Tools, the Integration Component Key specifies which of the integration component fields should be used for a user key.

The Integration Object Builder wizard computes the user keys by traversing several other Siebel types, including the business object, business component, table, and link types. Not every user key will meet the requirements to be used as the basis for integration object user keys.

To understand how the Integration Object Builder wizard determines valid integration component keys, you can simulate the process of validating the user keys.

For example, determine the table on which your business component is based. In Siebel Tools, you can look up this information yourself. Navigate to the Business Components screen and select a business component and check the Table column.

You can then navigate to the Tables screen, locate the table you want—in this example, S_CONTACT—and open the User Keys applet to see the user keys defined for that table.

Tables					
		Extend	Apply	Activate	
W	Name	Changed	Project	User Name	
>	S_CONTACT		Newtable	Person	
<					

User Keys						
W	Name	Changed	User Key Type	Source Interface Table	Inactive	Index
>	S_CONTACT: ERP Interface		ERP Interface		✓	S_CONTACT_EI
	S_CONTACT: New_S2K_SO		New_S2K_SO			S_CONTACT_UI
	S_CONTACT: Scopus Migration		Scopus Migration			S_CONTACT_MI
	S_CONTACT_EI		EI Index			S_CONTACT_EI
	S_CONTACT_I1		Integration Id		✓	S_CONTACT_I1
	S_CONTACT_U1		Traditional U1 Index			S_CONTACT_U1
	S_CONTACT_U1 - Std Replaced		Replaced			S_CONTACT_U1

Not every user key will necessarily be valid for a given business component. Multiple business components can map to the same underlying table; therefore, it is possible that a table's user key is not valid for a particular business component but is specific to another business component.

User Keys				
	W	Name	Changed	User Key Type
		S_CONTACT1: Scopus Migration		Scopus Migration
		S_CONTACT_EI		EI Index
		S_CONTACT_II		Integration Id
		S_CONTACT_U1		Traditional U1 Index
		S_CONTACT_U1 - Std Replaced		Replaced

User Key Columns				
	W	Name	Changed	Column
		BU_ID		BU_ID
		PERSON_UID		PERSON_UID
		PRIV_FLG		PRIV_FLG

Integration Platform Technologies MidMarket Edition 33

About Integration Objects

Structure of Siebel Integration Objects

If the columns of the user key are exposed in the business component and those columns are not foreign keys, the Integration Object Builder wizard creates an integration component key based on the table's user key. The Integration Object Builder wizard also defines one integration component key field corresponding to each of the table's user key columns. For example, in [Figure 14](#), the user key columns are exposed in the Integration Component Fields applet for the Contact integration component.

Business Components									
W	Name	Changed	Project	Cache Data	Class	Data Source	Dirty Reads	Distinct	
➤	Contact	✓	Contact		CSSEQuery		✓		
Fields									
Required	W	Name	Changed	Join	Column	PickList			
➤		Accomplishments		S_CONTACT_T	ACCOMPLISH				
		Account							
		Account Currency Code		Contact - S_ORG	BASE_CURCY_CI				
		Account Id		S_CONTACT	PR_DEPT_OU_ID				
		Account Integration Id		Contact - S_ORG	INTEGRATION_I	PickList Account			

Figure 14. Integration Component Field List

The Integration Object Builder wizard, for the preceding example, builds the integration component keys based on these table user keys. As illustrated in [Figure 15](#), the wizard defines one integration component key field for each table user key column.

Integration Components						
	W	External Name Context	Name	Changed	Parent Integration Component	External Name
>		Contact	Contact	✓		Contact
		Contact Note	Contact Note	✓	Contact	Contact Note
		Contact_Account	Contact_Account	✓	Contact	Account
		Contact_Business Address	Contact_Business Address	✓	Contact	Business Address
		Contact_Position	Contact_Position	✓	Contact	Position
<						

Integration Component Keys								
	W	Name	Changed	Key Sequence	Target Key Name	Key Type	Inactive	Comments
		V70 Wizard-Generated User Key:1	✓	1		User Key		
		V70 Wizard-Generated User Key:10	✓	10		User Key		
		V70 Wizard-Generated User Key:11	✓	11		User Key		
		V70 Wizard-Generated User Key:2	✓	2		User Key		
		V70 Wizard-Generated User Key:3	✓	3		User Key		
		V70 Wizard-Generated User Key:4	✓	4		User Key		
		V70 Wizard-Generated User Key:5	✓	5		User Key		
		V70 Wizard-Generated User Key:6	✓	6		User Key		
		V70 Wizard-Generated User Key:7	✓	7		User Key		
		V70 Wizard-Generated User Key:8	✓	8		User Key		
>		V70 Wizard-Generated User Key:9	✓	9		User Key		

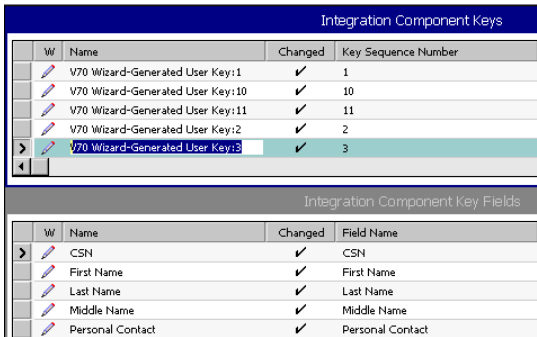
Figure 15. Integration Component Keys for Each Table User Key Column

About Integration Objects

Structure of Siebel Integration Objects

Each valid integration component key will contain fields. For example, as shown in [Figure 16](#), for the Contact integration component, User Key 3 is made up of five fields: CSN, First Name, Last Name, Middle Name, and Personal Contact.

NOTE: You should only modify user keys if you have a good understanding of the business component and integration logic.



The screenshot displays two tables from the Siebel Integration Object Builder. The top table, 'Integration Component Keys', lists several user keys. The bottom table, 'Integration Component Key Fields', shows the fields associated with the selected user key, 'V70 Wizard-Generated User Key:3'.

W	Name	Changed	Key Sequence Number
	V70 Wizard-Generated User Key:1	✓	1
	V70 Wizard-Generated User Key:10	✓	10
	V70 Wizard-Generated User Key:11	✓	11
	V70 Wizard-Generated User Key:2	✓	2
>	V70 Wizard-Generated User Key:3	✓	3

W	Name	Changed	Field Name
>	CSN	✓	CSN
	First Name	✓	First Name
	Last Name	✓	Last Name
	Middle Name	✓	Middle Name
	Personal Contact	✓	Personal Contact

Figure 16. Contact Integration Component Key Fields

When the Integration Object Builder wizard creates these integration component keys, it attempts to use the appropriate table user keys, that is, the user keys that will help uniquely identify a given record. In some cases, you may find that certain integration component keys created by the Integration Object Builder wizard are not useful for your particular needs. In that case, you can manually inactivate the keys you do not want to use by checking the Inactive flag on that particular user key in Siebel Tools. You can also inactivate user key fields within a given user key.

NOTE: For ease of maintenance and upgrade, inactivate unnecessary generated User Keys and User Key Fields instead of deleting them.

Status Keys

In the context of Siebel business objects, user keys are a group of fields whose values must uniquely identify only one Siebel business component record. Integration components within a corresponding integration object also contain user keys.

For many integrations, you want to know the status. For example, if you are sending an order request you want to know the ID of the Order created so that you can query on the order in the future. You can set the Status Object of the EAI Siebel Adapter to `True` to return an integration object instance as a status object.

The status returned is defined in the Integration Component using Status Keys. A Status Key is an Integration Component key of the type Status Key. Fields defined as part of the Status Key are included in the returned Status Object. If a Status Key is not defined for the Integration Component then neither the component nor any of its children are included in the returned object.

- To include descendants of an Integration Component without including any of its fields in the returned status object, specify an empty Status Key.
- To include information about which operation—update, insert, or delete—was performed during an upsert or synchronize request, include a field named *Operation* in the Status Key.

Hierarchy Parent Key

The Hierarchy Parent Key is used for integration objects that have a homogeneous hierarchy. This key should only have the parent id. The Hierarchy Parent Key is used for maintaining the hierarchy and keeping the data normalized.

For example, when you insert quotes, each quote item in turn can have more quote items. In this case the very first quote item inserted by the EAI Siebel Adapter has the hierarchy parent id set to blank, but for each child quote item, the EAI Siebel Adapter checks the keys to figure out which fields are to be set. If Hierarchy Parent Key is not defined, then the child quote item will be inserted as a new quote item without a link to its parent (denormalized).

Hierarchy Root Key

The Hierarchy Root Key is an optional key that is useful only when integration objects have a homogeneous hierarchy. You can use this key to improve performance. The Hierarchy Root Key should have only one field which the EAI Siebel Adapter populates with the value of the Id field in the component instance that is in the root of the homogenous hierarchy. For example, assume quote Q1 has quote items A, B, and C where each of the quote items has child quote items (A1, A2, B1, B2,...). If you want to update the quantity requested for all quote items starting with the root quote item B, it will be faster if the data is de-normalized. Using the Hierarchy Root Key, you can search for all records with root id equal to the row id of B and set the QuantityRequested field for each item.

NOTE: When the business component is hierarchy enabled, then the wizard automatically sets the 'Hierarchy Parent Key' for the complex integration component. To have a business component hierarchy enabled you need to set the property 'Hierarchy Parent Field'.

Creating and Maintaining Integration Objects

2

This chapter describes how to use the Integration Object Builder in Siebel Tools to create new Siebel integration objects. This wizard guides you through the process of selecting objects (either from the Siebel repository or from an external system) on which you can base your new Siebel integration object. This chapter also describes how to fine-tune and refine the integration object you have created.

Integration Object Builder Overview

The Integration Object Builder builds a list of valid components from which you can choose the components to include in your Siebel integration object.

NOTE: The Integration Object Builder provides a partial rendering of your data in the integration object format. You must review the integration object definition and complete the definition of your requirements. In particular, you should confirm that user key definitions are defined properly. You may need to enter keys and user properties manually or inactivate unused keys and fields in Siebel Tools. You should not expect to use the integration object without modification.

The following checklist gives the high-level procedure for creating an integration object.

Checklist

- ☐ Create an integration object using the EAI Siebel Wizard.
For details, see [“Creating an Integration Object Using the EAI Siebel Wizard” on page 41.](#)
 - ☐ Fine-tune your integration object.
For details, see [“Siebel Integration Object Fine-Tuning” on page 44.](#)
 - ☐ Validate your integration object.
For details, see [“Integration Object Validation” on page 44.](#)
-

Creating an Integration Object Using the EAI Siebel Wizard

Siebel Tools provides a wizard to walk you through creating an integration object. You should use this wizard to create your integration object.

To create a new Siebel integration object

- 1** Start Siebel Tools.
- 2** Create a new project and lock it, or lock an existing project in which you want to create your integration object.
- 3** Choose File > New Object... to display the New Object Wizards dialog.
- 4** Select the EAI tab and double-click the Integration Object icon.
- 5** In the Integration Object Builder wizard:
 - a** Select the project you locked in [Step 2](#).
 - b** Select the EAI Siebel Wizard.
- 6** Click Next and in the second page of the Integration Object Builder wizard:
 - a** Select the source object. This is the object that will be the model for the new Siebel integration object.

NOTE: Only business objects with Primary Business Components appear on this picklist.

- b** Type a unique name in the field for the new Siebel integration object and click Next.

NOTE: The name of an integration object must be unique among other integration objects.

The next page of the wizard, the Integration Object Builder - Choose Integration Components page, displays the available components of the object you chose.

- 7** Deselect the components you would like the wizard to ignore. This means you will not be able to integrate data for that component between the Siebel application and another system.

NOTE: Any component that has a plus sign (+) next to it is a parent in a parent-child relationship with one or more child components. If you deselect the parent component, the children below that component are deselected as well. You cannot include a child component without also including the parent. The Integration Object Builder enforces this rule by automatically selecting the parent of any child you choose to include.

For example, assume you have chosen to build your Siebel integration object on the Siebel Account business object and you want to create an integration component based on the Account and Contact business components.

- a** Deselect the Account integration component at the top of the scrolling list.

This action deselects the entire tree below Account.

- b** Select the Contact component.

NOTE: When selecting a child component, its parent component will also be selected, but none of the components below the child component are selected. You must select the ones you want individually.

- 8** Click Next.

The next page displays error or warning messages generated during the process. Review the messages and take the appropriate actions to address them.

- 9** Click Finish to complete the process of creating a new Siebel integration object.

NOTE: After creating Integration Objects in Siebel Tools, you must compile a new .srf file and copy the .srf file to the Siebel Server root/objects directory.

Your new Siebel integration object appears in the list of integration objects in Siebel Tools.

On the Integration Components screen, the Account integration component is the only component that has a blank field in the Parent Integration Component column. The blank field identifies Account as the root component. The Siebel integration object also contains the other components selected, such as Contact and its child components.

NOTE: Once you created your integration object based on a Siebel Business Object, you should not change its integration component's External Name Context; otherwise, the synchronization process will not be able to recognize the integration component and will remove it from the integration object.

- 10** To view the fields that make up each integration component, select a component from the integration component list in Siebel Tools.

The list of fields for that component displays in the Integration Component Fields applet.

NOTE: The XML Sequence property in this applet defines the order in which the XML tags appear when an integration object is created.

Siebel Integration Object Fine-Tuning

After you create your integration object you need to fine-tune and customize your integration object based on your business requirements. Following is a list of the most common practices in fine-tuning an integration object.

- Deactivate the fields that do not apply to your business requirements.
- If necessary, activate the fields that have been deactivated by the Siebel wizard. For details, see [Chapter 1, “About Integration Objects.”](#)
- Add the fields that have not been included by the Siebel wizard. For details on the implications of activating such fields, see [“Calculated Fields” on page 29](#) and [“Inner Joins” on page 29](#).
- Validate the User Keys. For details, see [Chapter 1, “About Integration Objects.”](#)
- Update the User Properties for your Integration Object to reflect your business requirements. For details, see [“Integration Object User Properties” on page 59](#).

Integration Object Validation

Once you have created your integration object and made the necessary modifications to meet your business requirements, you need to validate your integration object.

To validate your integration object

- 1** Open Siebel Tools.
- 2** Select your integration object.
- 3** Click Validate.
- 4** Review the report and modify your integration object as needed.

NOTE: Integration objects you create in Siebel Tools must be compiled into the Siebel.srf file. Once you test the integration object, you must copy the compiled .srf to your server\objects directory.

Integration Objects Synchronization

Business objects often require updates to their definitions to account for changes in data type, length, edit format, or other properties. It is common to want to alter database metadata, but if you do so you have to also update your integration objects to account for these updates. Otherwise, you can cause undesirable effects on your integration projects.

Some examples of these changes are:

- A field removed
- A new required field
- A new picklist for a field
- A change of relationship from one-to-many to many-to-many
- An upgrade to a new version of Siebel applications

Synchronization Considerations

To help simplify the synchronization task, Siebel eAI provides an integration object synchronization utility. Although the process of synchronizing your integration object with its underlying business object is straightforward, you should review the integration objects you have modified to make sure that you have not inadvertently altered them by performing a synchronization. After synchronization, you should validate your integration object.

The following checklist gives the high-level procedure for updating an integration object.

Checklist

-
- ☐ Run the Synchronization wizard.
For details, see [“Updating the Entire Integration Object” on page 50](#).

 - ☐ Modify the newly updated integration object as needed, using the DIFF tool and the copy of the integration object as reference.
For details, see *Siebel Tools Reference, MidMarket Edition*.

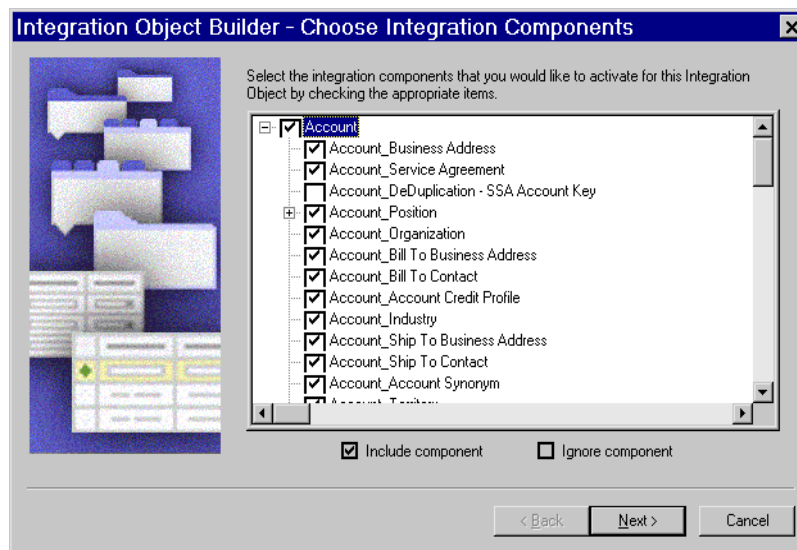
 - ☐ Run Validation.
For details, see [“Integration Object Validation” on page 44](#).
-

To update an integration object

- 1 Access the integration object you want to update in Siebel Tools.
- 2 Run the Synchronization wizard by double clicking on the Synchronization button.

NOTE: The update process overrides the integration object and deletes user keys, user properties, and so on. You can use the copy of the integration object made by the Synchronization wizard to see how you modified the object.

- a On the Integration Objects Builder, click on the plus sign to list all the related integration components, as shown in the following figure.



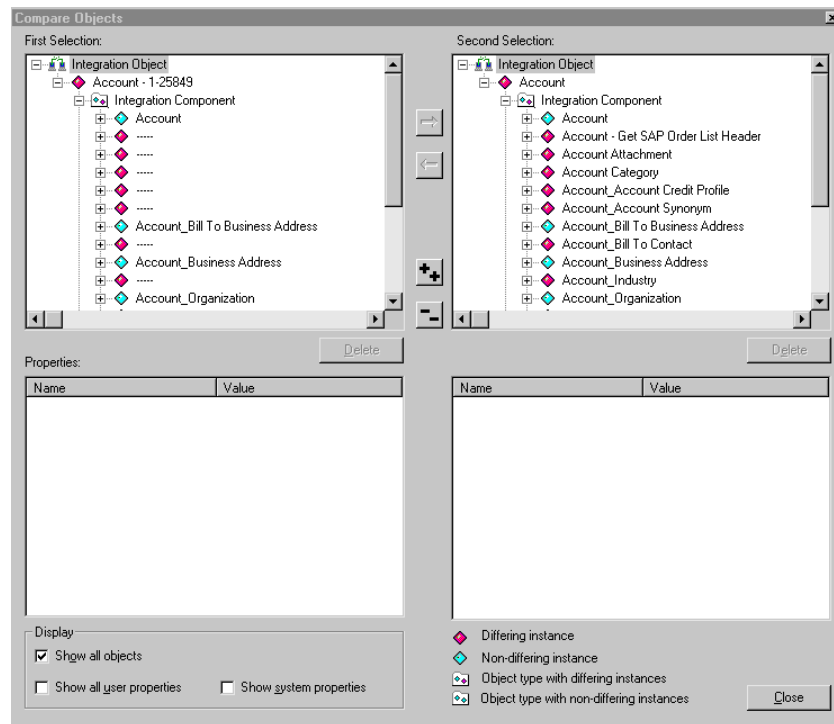
The process of retrieving Siebel integration object and Siebel business object definitions can take varying amounts of time depending on the size and detail of the selected objects.

- b** Uncheck the boxes beside the objects and components you do not want to include in the synchronization of your Siebel integration object. Note that only the objects that are included in the new integration object are marked.

NOTE: The process of performing the synchronization can take some time, depending on the complexity of the selected objects.

- c** Click Finish to synchronize the Siebel integration object and the Siebel business object.

The Compare Objects dialog box, shown below, appears.



This tool allows you to move properties and objects between versions using arrow buttons.

When you synchronize the Siebel integration object and the Siebel business object, the Synchronization wizard performs update, insert, and delete operations—by selecting or deselecting components—to make the Siebel integration object look like the representation of the Siebel business object you chose.

The wizard generally updates the Siebel integration object either by updating the object and its components or by updating some components and deleting others. For details, see [“Updating the Entire Integration Object” on page 50](#) and [“Deleting a Component from the Integration Object” on page 52](#).

3 Copy custom properties and custom user keys as needed.

The wizard includes any new fields—added to the business object for the new version of your Siebel application—in your integration object and sets all these fields to active.

4 Deactivate any new field that you do not need as a component of your updated integration object.

5 Right click on your integration object, and select the Validate option to validate your integration object.

NOTE: If you need to synchronize any of the external integration objects, you should also follow this general procedure to perform a synchronization operation.

Synchronization Rules

During the synchronization process, the Synchronization wizard follows particular update rules. Consider a simple example involving the Siebel Account integration object with only Contact and its child components marked as active in the object. [Figure 17](#) helps you to visualize this example.

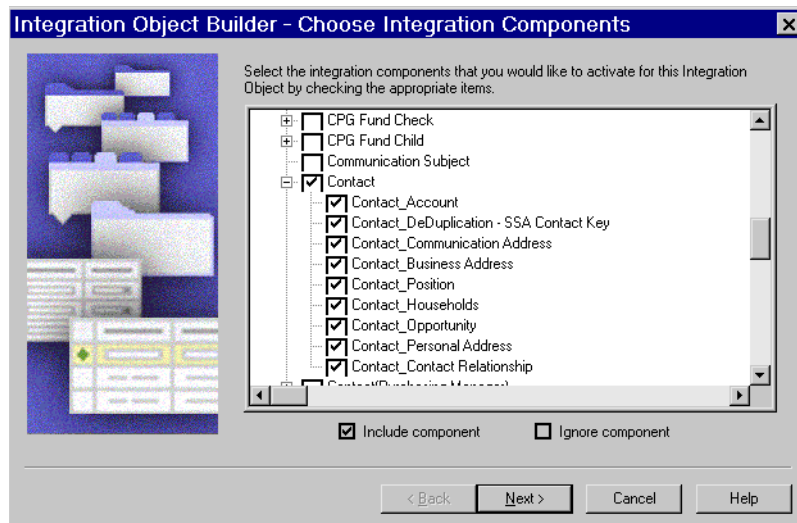


Figure 17. Example of Selected Integration Components

Since the Account component is the parent of Contact, it is also selected, even though you cannot see it in [Figure 17](#).

Updating the Entire Integration Object

After initiating the Synchronization wizard, if you check the boxes in the wizard, the wizard creates a new integration object in memory. If the underlying Siebel business object has been changed, then the new, in-memory integration object will be different from the integration object in the database. As a result, the wizard will synchronize the outdated integration object in the database with the new, in-memory integration object.

Figure 18 illustrates this concept.

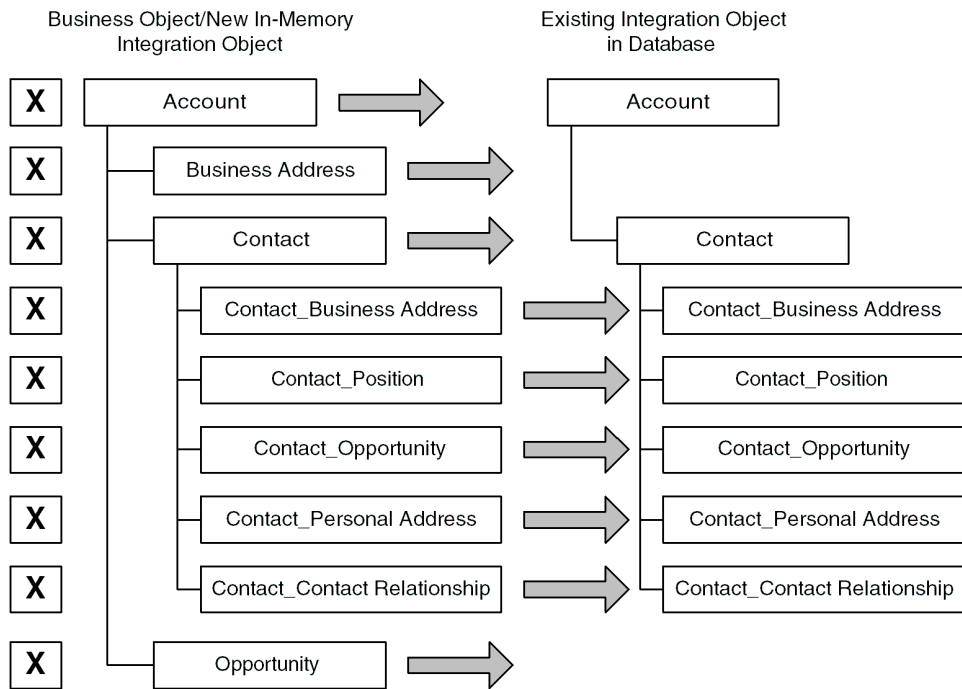


Figure 18. Synchronizing the Integration Object

Figure 19 shows how the resulting integration object will be structured after the synchronization.

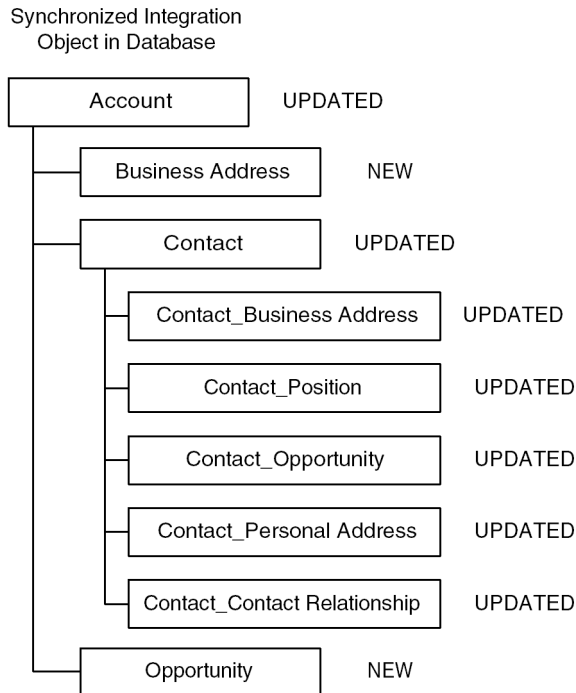


Figure 19. Completely Updated Integration Object

The integration object now contains two new components, *Business Address* and *Opportunity*. Other components have been updated with the definitions of the corresponding components in the business object.

Deleting a Component from the Integration Object

If you choose to deselect a component in the Synchronization wizard, you specify to the wizard that it should delete the component in the integration object with the matching External Name Context property. The integration object that exists in the database has a component with the same External Name, External Name Sequence, and External Name Context as the deselected component in the new, in-memory integration object.

Figure 20 illustrates this concept.

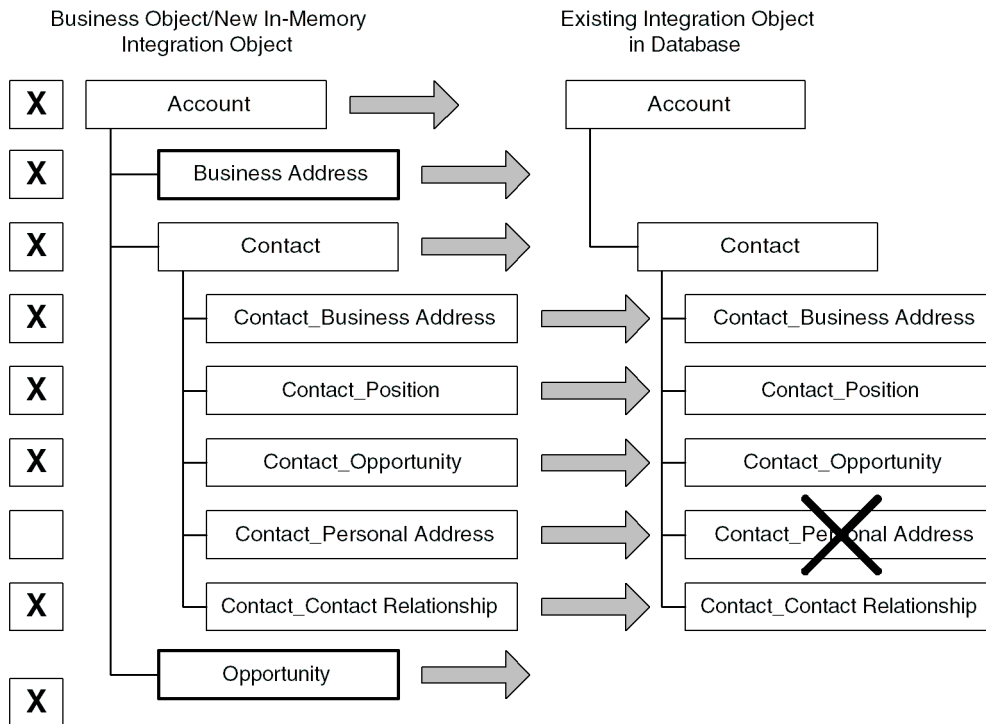


Figure 20. Deleting a Component from the Integration Object

Figure 21 shows the integration object after synchronization.

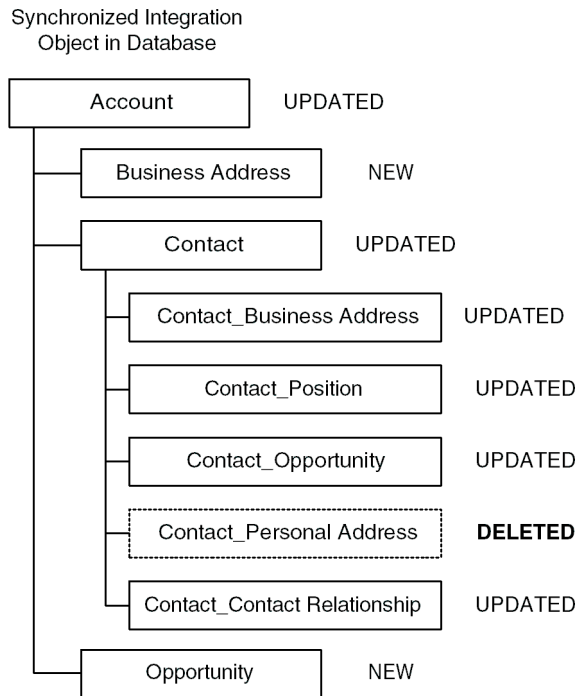


Figure 21. Synchronization Resulting in a Deleted Component

The component *Contact_Personal Address* has been deleted. When you use the updated integration object, you will not be able to pass data for that component between the Siebel application and an external application.

This example is intended to show how you might cause unexpected results by deselecting components. However, if you do want to delete a particular component from the integration object, deleting a component from the integration object method will accomplish your goal.

As the examples illustrate, you need to be aware of the possible changes that can occur when you are synchronizing business objects and integration objects. The Synchronization wizard can provide assistance in managing your integration objects, but you need to have a clear understanding of your requirements, your data model, and the Siebel business object structure before undertaking a task as important as synchronization.

The EAI Siebel Wizard

You can use the EAI Siebel Wizard to create integration objects that represent Siebel business objects. During the process of creating a new integration object, described in [“Integration Object Builder Overview” on page 40](#), you can choose the EAI Siebel Wizard as the business service to help create the object. This wizard understands the structure of Siebel business objects. As shown in [Figure 22](#), the wizard returns a list of the available business objects from which you can choose one to base your integration object on.

The wizard also returns a list of the available components contained within the object you have chosen. When you select certain components in the wizard, you are activating those components in your integration object. Your integration object actually contains the entire structural definition of the business object you selected in the first wizard dialog box. Only the components you checked, or left selected, are active within your integration object. That means that any instances you retrieve of that integration object will contain only data represented by the selected components.

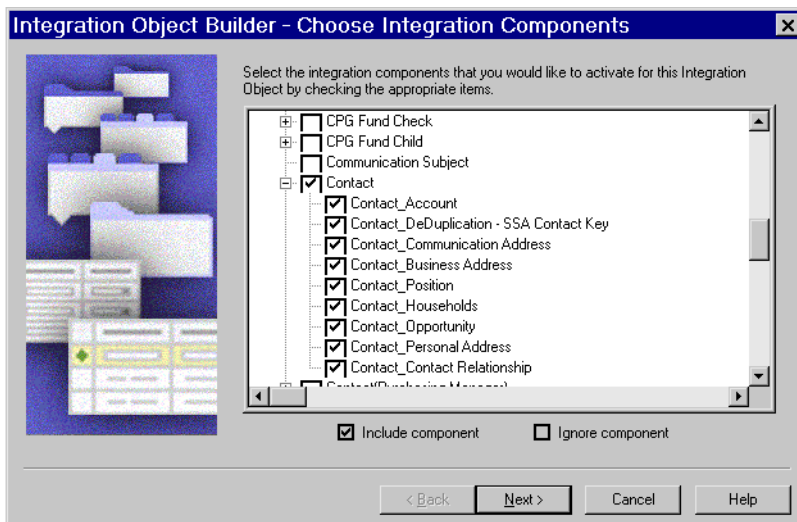


Figure 22. Activated Components in the Contact Integration Object

After the wizard creates your integration object, you can edit the object in Siebel Tools, as shown in [Figure 23](#). You might choose to drill down into the integration components and activate or inactivate particular components or even particular fields within one or more components.

NOTE: You should always deactivate the fields and not delete them even though the net effect (as well as the DTD generated) will be the same. When you execute the synchronization task, using the Siebel eAI sync utility in Siebel Tools, inactivated fields will remain inactive, while the deleted fields will be created as active fields in the Integration Object.

Integration Components					
W	External Name Context	Name	Changed	Parent Integration Component	Ext
		Contact_Contact Relationship	✓	Contact	Cont
		Contact_DeDuplication - SSA Contact Key	✓	Contact	DeDu
		Contact_Households	✓	Contact	Hous
		Contact_Opportunity	✓	Contact	Opp
		Contact_Personal Address	✓	Contact	Pers

Integration Component Fields					
W	Name	Changed	Data Type	Length	Precision
	Address Name	✓	DTYPE_TEXT	100	
	Integration Id	✓	DTYPE_TEXT	30	
	Personal City	✓	DTYPE_TEXT	50	
	Personal Country	✓	DTYPE_TEXT	50	
	Personal Postal Code	✓	DTYPE_TEXT	30	
	Personal State	✓	DTYPE_TEXT	10	
	Personal Street Address	✓	DTYPE_TEXT	200	

Figure 23. Activated Components in the Contact Integration Object

Siebel Integration Objects Maintenance and Upgrade

Sometimes you may change the underlying business objects, which necessitates maintenance of the integration object. Synchronize the integration object by clicking the synchronize button.

To make maintenance of integration objects easier, adhere to the following guidelines when creating or editing your integration objects:

- Name any User Key that you add differently from the generated User Keys. Using meaningful names will help with debugging.
- Inactivate User Keys instead of deleting them.
- Inactivate fields instead of deleting them.

Permission Rules for Integration Components

Each Business Component, Link, MVG, and Integration Object user property has settings such as No Update, No Delete, and No Insert, which indicate the type of operations that cannot be performed on that object. In order for the EAI Siebel Adapter to successfully perform an operation, that operation needs to be allowed at all levels. If the operation is allowed at every level but the field level, a warning message is logged in the log file and processing will continue. Otherwise an error message is returned, and the transaction is rolled back.

Table 2 illustrates which permissions influence which operation type on an integration component with an example of Account Business Address and Business Address in the last column.

Table 2. Permission Rules for an Integration Component

Permission Layer	Checked by...	Integration Component Type		
		Standard	MVG	Association
Integration Object Component	EAI Siebel Adapter	✓	✓	✓
Integration Component		✓	✓	✓
Integration Field (Update Only)		✓	✓	✓
Link	Object Manager	✓	✓	✓
Multi-Value Link (MVL)			✓	
Business Component (Overridden by AdminMode)		✓	✓	✓
Business Component Field		✓	✓	✓

NOTE: The transaction will be rolled back if any of the permissions (excluding field-level permissions) are denied.

EAI Siebel Adapter Access Control

You can use the following mechanisms to control EAI Siebel Adapter access to the database:

- **Restricted access to a static set of integration objects.** You can configure the EAI Siebel Adapter Business Service, or any business service that is based on the CSEEAISiebelAdapterService, to restrict access to a static set of integration objects. To do this, set a Business Service User Property called `AllowedIntObjects`, which contains a comma-separated list of integration object names that this configuration of the EAI Siebel Adapter can use. This allows you to minimize the number of integration objects your users need to expose outside of Siebel applications through HTTP inbound or MQ Series Receiver server components. If this user property is not specified, the EAI Siebel Adapter uses any integration objects defined in the current Siebel Repository.
- **ViewMode.** You can specify the visibility mode of Business Components that the EAI Siebel Adapter uses. This mode is specified as the integration object user property `viewMode`. This user property can take different values, as defined by LOV type `REPOSITORY_BC_VIEWMODE_TYPE`.

NOTE: For details, see *Siebel Tools Online Help, MidMarket Edition*.

Integration Object User Properties

You can define user properties for your integration objects. These user properties help determine special processing and behavioral requirements of integration objects for a specific eAI adapter.

The Level column shown in [Table 3](#) can take on the following values:

- O, for Object Level
- C, for Component Level
- F, for Field Level

Table 3. Integration Objects User Properties

User Property	Allowable Values	Level	Default Value	Description
Association	Y,N	C	N	If set to Y, it labels the integration component as having a many-to-many relationship with its parent integration component configured by a Link with an intersection table. Not applicable to root integration component.
MVG	Y,N	C	N	If set to Y, it labels the integration component as having a M:1 relationship with its parent integration component configured by a Multi Value Link defined over a Link without an intersection table. Not applicable to root integration component.
Picklist	Y,N	F	N	If Y field is based on a picklist and the EAI Siebel Adapter validates the field value using an associated picklist, bounded or non-bounded. If the picklist is non-bounded and the value does not match then the EAI Siebel Adapter logs a warning but the value will still be set accordingly. If this property is set to N, or is not defined, then the EAI Siebel Adapter leaves the validation to the Object Manager, aborts the processing if the validation fails for the bounded picklist, and logs a warning. See “Performance Considerations” on page 64 .

Table 3. Integration Objects User Properties

User Property	Allowable Values	Level	Default Value	Description
Ignore Bounded Picklist	Y,N	O,C,F	N	If this property is set to “N” and the Picklist is set to “Y”, and the value provided does not match any of the values in the picklist, then the EAI Siebel Adapter will stop processing, write an error to the log file, and roll back the transaction. If Ignore Bounded Picklist is set to “Y” and the value provided does not match any of the values in the picklist, then the EAI Siebel Adapter reports a warning in the log file and sets the field to Null.
MVGAssociation	Y,N	C	N	If set to “Y”, it means the integration component has a M:M relationship with its parent integration component configured by a Multi-Value Link defined over a Link with an intersection table. Not applicable to the root integration component.
MVGFieldName	Any valid field name in the business component	F	N/A	If the component that owns this integration field is labeled MVG, the value of this user property gives the name of the business component field as the MVG component knows it. In this case, the External Name property of Integration Component Fields references the field name in the parent business component.
AssocFieldName	Any valid field name in the business component	F	N/A	If the component that owns this integration field is labeled MVGAssociation, the value of this user property gives the name of the business component field as the Association MVG component knows it. In this case, the External Name property of Integration Component Fields references the field name in the parent business component.
NoInsert	Y,N	C	N	If this property is set to “Y” at the component level, it means that the EAI Siebel Adapter is prevented from inserting a new record into the component and the EAI Siebel Adapter will abort the processing of the EAI Message and returns an error message. This is to allow limiting the functionality of an Integration Object to a subset of what the underlying Business Object allows but it cannot override limitations imposed by the Business Object, Business Component, Link, and Multi-Value-Link.

Table 3. Integration Objects User Properties

User Property	Allowable Values	Level	Default Value	Description
NoDelete	Y,N	C	N	If this property is set to “Y” at the Component level, no records in that component can be deleted and the EAI Siebel Adapter aborts the processing of the EAI Message and returns an error message. This is to allow limiting the functionality of the Integration Object to a subset of what the underlying Business Object allows but it cannot override limitations imposed by the Business Object, Business Component, Link, and Multi-Value-Link.
NoUpdate	Y,N	C,F	N	If this property is set to “Y” at the Component level, no fields in that component can be updated. If an existing record needs to be updated, the EAI Siebel Adapter aborts the processing of the EAI Message and returns an error message. If this property is set to ‘Y’ on the field level, then the EAI Siebel Adapter only logs a warning message and skips updating the field, and continues processing. This is to allow limiting the functionality of the Integration Object to a subset of what the underlying Business Object allows but it cannot override limitations imposed by the Business Object, Business Component, Link, Multi-Value-Link, or Field.
FieldDependencyXXX	Any active integration component field name within the same integration component	F	N/A	Defines a dependency between the integration field that has this user property and the integration field specified by the value of the user property. Multiple dependencies are specified by separate user property entries. The field specified in the value must be from the same component as the field that has the user property. Dependencies constrain the order of field processing within an integration component. If field A depends on field B, then field B will be processed before field A. Also see Chapter 1, “About Integration Objects.”

Table 3. Integration Objects User Properties

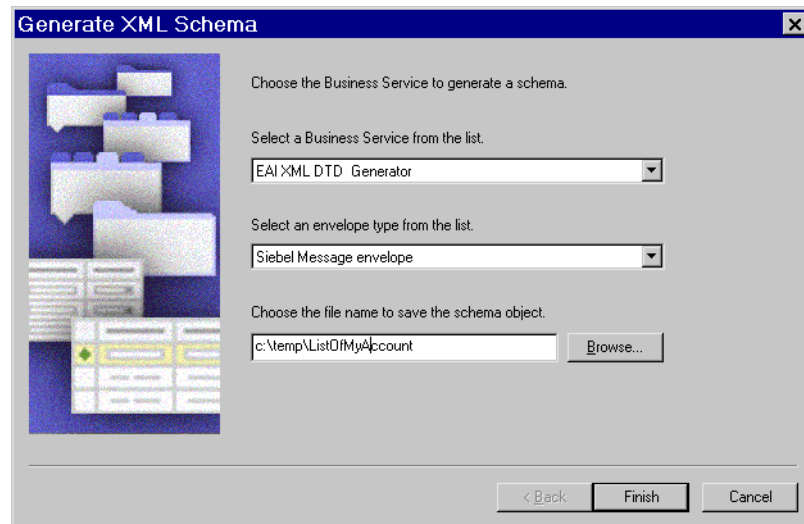
User Property	Allowable Values	Level	Default Value	Description
AdminMode	Y,N	C,O	N	Sets AdminMode on the business component. Some business components, such as Internal Product, allow only administrators to make modifications. You may allow modification of such components during integration, by setting the AdminMode property to Y on either the Integration Component or the Integration Object level. The setting at the Integration Component level will override the setting at the Business Object level. This property can not be used in MVGs. For details, see <i>Siebel Tools Online Help, MidMarket Edition</i> .
ViewMode	Manager, Sales Rep, Personal, Catalog, Group, Organization, All	O	All	Specifies the visibility mode of the business component that the EAI Siebel Adapter uses. The allowable values are based on REPOSITORY_BC_VIEWMODE_TYPE LOV.
AllLangIndependent Vals	Y,N	O	N	If set to Y, this user property forces the EAI Siebel Adapter to use language-independent values for LOV-based integration component fields. This is useful when there is a requirement to support integration between systems that use multiple languages. If set to N, all LOV-based fields will use language-dependent values. If this user property is not defined for the integration object, multilingual LOV-based fields (MLOV) will use language independent values, while single-language LOV fields will use language dependent values.

Generating Schemas

At certain points in your integration project, you may want to generate schemas from an integration object. If you export Siebel Integration Objects as XML to other applications, you may need to publish the schemas of such objects so that other applications can learn about the structure of the XML to expect.

To generate an integration object schema

- 1 In Siebel Tools, click on an integration object to make it the active object.
- 2 Click Generate Schema to access the Generate XML Schema wizard shown in the following figure.



- 3 Choose the EAI XML DTD Generator business service.
- 4 Choose an envelope type to use in generated DTD.
- 5 Choose a location where you want to save the resulting DTD file and click Finish.

The wizard generates a DTD of the integration object you selected. Use this DTD to help you map external data directly to the integration object. The DTD serves as the definition for the XML elements you can create using an external application or XML editing tool.

Performance Considerations

To optimize your integration object performance, you may want to consider the following.

Size of Integration Object

The size of an integration object and its underlying business components can have a large impact on the latency of the EAI Siebel Adapter operations. You should inactivate unnecessary fields and components in your integration objects.

Force-Active Fields

You should reexamine any fields in the underlying business component that are force-active. Such fields are processed during integration even if they are not included in the integration component. You might want to consider removing the force-active specification from such fields, unless you absolutely need them.

Picklist Validation

Siebel applications have two classes of picklists, static picklists based on list of values and dynamic picklists based on joins.

Setting the property PICKLIST to Y in the Integration Field directs the EAI Siebel Adapter to validate that all operations conform to the picklist specified in the field. For dynamic picklists, this setting is essential to make sure the joins are resolved properly. However, for unbounded static picklists, this validation may be unnecessary and can be turned off by setting the property to N. Even for bounded static picklists, validation in the adapter can be turned off because the Object Manager can perform the validation. Turning off the validation at the EAI Siebel Adapter means that picklist related warnings and debugging messages will not show up along with other EAI Siebel Adapter messages (although validation errors will still be reported in a different format). This also means that bounded picklist errors will not be ignored even if Ignore Bounded Picklist is set to Y.

NOTE: Validation of a bounded picklist done in the EAI Siebel Adapter is about 10% faster than performing the validation in the Object Manager.

Business Component Restrictions

The business components underlying the Integration Components may have certain restrictions. For example, Internal Product can only be modified by an administrator. The same restrictions also apply during integration. In many cases, the Siebel Integration Object Builder wizard will detect the restrictions and set properties such as No Insert or No Update on the Integration Components.

System Fields

Integration Object fields marked as System are not exported during a query operation. This setting prevents EAI Siebel Adapter from treating the field as a data field, which means for Query and QueryPage method EAI Siebel Adapter will not output the field. For the Synchronize and Update method, the field will not be directly set in the Business Component unless the ISPrimaryMVG is set to Y.

NOTE: If you want to include System fields in the exported message, change the Integration Component field type to Data. System fields are read only. If you attempt to send in a message with the value set for a System field, the setting will be ignored and a warning message will be logged.

Best Practices

- Familiarize yourself with the business logic in the business components. Integration designers should use the presentation layer or the user interface to get a good sense of how the business component behaves and what operations are allowed and not allowed.
- Design with performance in mind. See [“Performance Considerations” on page 64](#).
- Design with maintenance in mind. See [“Siebel Integration Objects Maintenance and Upgrade” on page 56](#).

This chapter outlines the basic concepts of a business service, its structure and purpose, and how you can customize and create your own business service. This chapter also describes how to test your business service before it is implemented.

Overview of Business Services

A business service is an object that encapsulates and simplifies the use of some set of functionality. Business components and business objects are objects that are typically tied to specific data and tables in the Siebel data model. Business services, on the other hand, are not tied to specific objects, but rather operate or act upon objects to achieve a particular goal.

Business services can simplify the task of moving data and converting data formats between the Siebel application and external applications. Business services can also be used outside the context of Siebel eAI to accomplish other types of tasks, such as performing a standard tax calculation, a shipping rate calculation, or other specialized functions.

These services can then be accessed by Siebel VB or Siebel eScript code that you write and call from workflow processes. For the purposes of your integration projects using Siebel eAI, you can use Siebel eScript to write your scripts to use the DTE scripts.

Creating Business Services

A Siebel application provides a number of prebuilt business services to assist you with your integration tasks. These are based on specialized classes and are called Specialized Business Services. Many of these are used internally to manage a variety of tables.

Caution: As with other specialized code such as Business Components, you should use only the specialized services that are documented in Siebel documentation. The use of undocumented services is not supported and can lead to undesired and unpredictable results.

In addition to the prebuilt business services, you can build your own business service and its functionality in two different ways to suit your business requirements:

- **In Siebel Tools.** Created at design time in Siebel Tools using Siebel VB or Siebel eScript. Design-time business services are stored in the Siebel repository (*.srf), so you have to compile the repository before testing them. Once your test is completed then you need to compile and disseminate the .srf to your clients. The business services stored in the repository will automatically come over to the new repository during the upgrade process. General business services are based on the class CSSService; however, for the purposes of Siebel eAI, you base your data transformation business services on the CSSEAIIDTEScriptService class.

See [“Creating Business Services in Siebel Tools” on page 72.](#)

- **In the Siebel Client.** Created at run time in the Siebel Client using the Business Service Administration screens. Run-time business services are stored in the Siebel database so they can be tested right away. The run-time business services have to be manually moved over after an upgrade process.

See [“Creating a Business Service in the Siebel Client” on page 78.](#)

NOTE: To use the DTE scripts, you need to write your business service in eScript. Otherwise, you can write your business service in Siebel VB.

Business Service Structure

Business services allow developers to encapsulate business logic in a central location, abstracting the logic from the data it may act upon. A business service is much like an object in an object-oriented programming language.

A service has properties and methods and maintains a state. Methods take arguments that can be passed into the object programmatically or, in the case of Siebel eAI, declaratively by way of workflows.

NOTE: For more details on business service methods and method arguments, see *Siebel Tools Online Help, MidMarket Edition*.

About Property Sets

Property sets are used internally to represent Siebel eAI data. A property set is a logical memory structure that is used to pass the data between business services. [Figure 24](#) illustrates the concept of a property set.

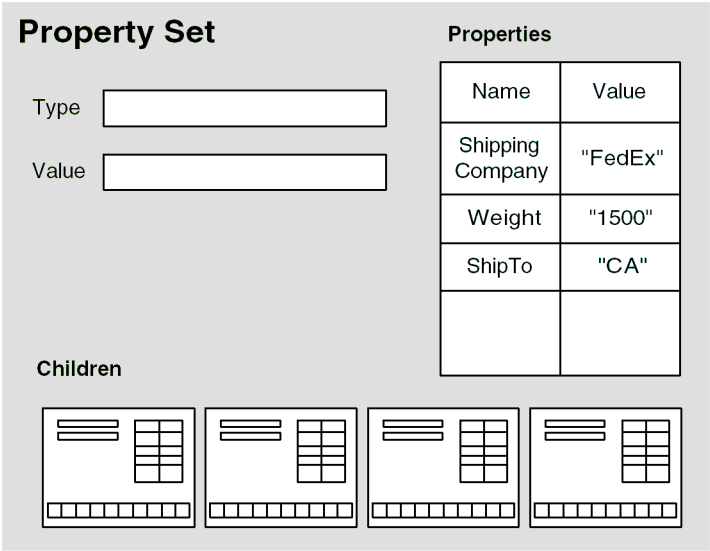


Figure 24. Property Set Structure

The property set consists of four parts:

- **Type.** Used to describe what type of object is being represented.
- **Value.** Used to hold serialized data, such as a string of XML data.

NOTE: In Siebel Tools, a Value argument to a method is shown with the name of <Value>, including the brackets. You can also define a Display Name for the Value argument in the Business Service Simulator. This Display Name appears in the Workflow Process Designer when you are building integration workflows. In this guide, the Display Name Message Text is shown when referring to the Value argument and the Name <Value> is shown when referring to the Value of the value argument.

- **Properties.** A table containing name-value pairs. The properties can be used to represent column names and data, field names and data, or other types of name-value pairs.
- **Children.** An array of child-level property sets. The array can be used to represent instances of integration objects; for example, a result set may contain an Account with some set of contact records from the database. Each contact record is represented as a child property set.

NOTE: For details on property sets and their methods, see *Siebel Tools Online Help, MidMarket Edition*.

Creating Business Services in Siebel Tools

The following sections explain how to create business services and business service scripts in Siebel Tools.

Checklist

-
- | | |
|--------------------------|---|
| <input type="checkbox"/> | Define the Business Service
For details, see “To define a business service in Siebel Tools” on page 73. |
| <hr/> | |
| <input type="checkbox"/> | Define the Business Service Methods
For details, see “To define a business service method” on page 74. |
| <hr/> | |
| <input type="checkbox"/> | Define the Business Service Methods Arguments
For details, see “To define the business service method arguments” on page 74. |
| <hr/> | |
| <input type="checkbox"/> | Define Business Service Scripts
For details, see “To define and write the business service script” on page 75. |
| <hr/> | |
| <input type="checkbox"/> | Define Business Service Subsystem
For details, see “To specify a business service subsystem” on page 76. |
| <hr/> | |
| <input type="checkbox"/> | Define Business Service User Properties
For details, see “To define business service user properties” on page 77. |
-

NOTE: Business services you create in Siebel Tools must be compiled into the Siebel .srf file. If you intend to run the business services on your Siebel server, then copy the compiled .srf file to your server\objects\ < lang > directory.

Defining a Business Service in Siebel Tools

You declaratively define the business service in Siebel Tools and then add your scripts to the business service in the Script Editor.

To define a business service in Siebel Tools

- 1** Start Siebel Tools.
- 2** Select and lock the project you want to associate your business service with.

NOTE: Each business service must belong to a project and the project must be locked. For details, see *Siebel Tools Reference, MidMarket Edition*.

- 3** Select the EAI Business Services object in the Tools Object Explorer.
The list of predefined business services appears in the right panel.
- 4** Choose Edit New Record to create a new business service.
- 5** Type a name for your business service in the Name field.
- 6** Type the name of the project you locked in [Step 2](#), in the Project field.
- 7** Choose the appropriate class for your business service, from the Class picklist.
 - Data transformation business services should use the CSSEAITEScriptService class.
 - Other business services will typically use the CSSService class.
- 8** Step off the current record to save your changes.

Defining Business Service Methods

Business services contain related methods that provide the ability to perform a particular task or set of tasks.

NOTE: For details, see *Siebel Tools Online Help, MidMarket Edition*.

To define a business service method

- 1 With your business service selected, double-click the Business Services Methods folder in the Siebel Tools Object Explorer.

The Business Services Methods list appears below the list of business services. If you have already defined methods for the selected business service, the method names appear in the Business Services Methods list.

- 2 Choose Edit > New Record to create a new method.
- 3 Type the name of the method in the Name field.

Defining Business Service Method Arguments

Each method can take one or more arguments. The argument is passed to the method and consists of some data or object that the method processes to complete its task.

To define the business service method arguments

- 1 With your business service selected, double-click the Business Service Method Arg folder, in the Tools Object Explorer, to display the Business Service Method Args list.
- 2 Choose Edit > New Record to create a blank method argument record.
- 3 Type the name of the argument in the Name field.

NOTE: If you plan to use this business service in a Siebel Client, you need to specify Display Name as well.

- 4 Enter the data type in the Data Type field.

- 5 Check the Optional check box if you do not want the argument to be required for the method.
- 6 Choose a Type for the argument. Refer to the following table for a list of different types and their descriptions.

Argument	Description
Input	This type of argument serves as input to the method.
Input/Output	This type of argument serves as both input to the method and output from the method.
Output	This type of argument serves as output from the method.

Defining and Writing Business Service Scripts

Business service scripts supply the actual functionality of the business service in either Siebel VB or Siebel eScript. As with any object, the script you provide is attached to the business service.

To define and write the business service script

- 1 Start Siebel Tools.
- 2 Select the business service for which you want to write a script.
- 3 Right-click to display a pop-up menu.
- 4 Choose Edit Server Scripts.
- 5 Select either eScript or Visual Basic for your scripting language.

Service-PreInvokedMethod is selected as the service.

NOTE: To write any Siebel VB script in the Business Services, your deployment platform must support Siebel VB.

- 6 Type your script into the Script Editor.

NOTE: You need to write your business service in eScript if you want to use the DTE scripts. For details on scripting, see *Siebel Tools Online Help, MidMarket Edition*.

Specifying Business Service Subsystems

You can optionally specify a business service subsystem. A business service subsystem is a server component that encapsulates a large amount of functionality and that is already included in the Siebel repository. Business service subsystems define particular events upon which the subsystem will be called. The subsystems can also trigger other events, depending on how they are defined. Examples of business service subsystems are listed in [Table 4](#).

Table 4. Business Service Subsystems

Subsystem	Description
EAISubsys	Defines events for a variety of eAI operations, including the initiation of eAI wizards, calls to eAI adapters, and calls to eAI validation routines.
SAPSubsys	Defines a variety of parameters to help determine the type of SAP object being integrated, the transport mechanism, user name and password combinations, and SAP program ID.
Workflow	Defines both events and parameters to signal and determine behaviors based on the initiation of workflow processes, search specifications, and Row ID.
XMLCnv	Defines events regarding debugging information and responses from the XML parser.

To specify a business service subsystem

- 1 With your business service selected, double-click the Business Service Subsystem folder in the Tools Object Explorer to display a list of subsystems.
- 2 Choose Edit > New Record to create a blank business service subsystem record.
- 3 Choose an existing business service subsystem name from the Subsystem picklist.

Defining Business Service User Properties

User properties, also known as User Props, are optional variables that you can use to define default values for your business services. When a script or control invokes your business service, one of the first tasks the service performs is to check the user properties to gather any default values that will become input arguments to the service's methods.

To define business service user properties

- 1** With your business service selected, double-click the Business Service User Prop folder in the Tools Object Explorer to display the list of Business Service User Props.
- 2** Choose Edit > New Record to create a blank user property record.
- 3** Type the name of the user property in the Name field.
- 4** Type a value in the Value field.

The value can be an integer, a quoted string, or a Boolean.

Creating a Business Service in the Siebel Client

You can define business services in the Siebel client using the Business Service Administration screens. The business services you create in the client are stored in the Siebel database. This section illustrates the creation of business services using the Business Service Methods screen, which includes applets to create and display the business service.

To define a business service in the Siebel Client

- 1** From the application-level menu, choose View > Site Map > Business Service Administration > Business Service Methods.

- 2** Click New to create a new record in the Business Service list applet.

Name. Name of the business service

Cache. If checked then the business service instance remains in existence until the user's session is finished; otherwise, the business service instance will be deleted after it finishes executing.

Inactive. Check if you do not want to use the business service.

- 3** Define methods for the business service in the Methods list applet.

Name. Name of the method.

Inactive. Check if you do not want to use the method.

- 4** Define method arguments for the methods in the Method Arguments list applet.

Name. Name of the method argument.

Type. The type of the business service method argument. Valid values are Output, Input, and Input/Output.

Optional. Check if you do not want this argument be optional.

Inactive. Check if you do not want to use the argument.

- 5 Write your Siebel eScript or VB code in the Business Service Scripts list applet.

NOTE: To write any Siebel VB script in the Business Services, your deployment platform must support Siebel VB.

- 6 Click Check Syntax to check the syntax of the business service script.

Business Service Export and Import

Business services can be exported into an XML file by clicking the Export Service... button in the Business Service list applet. This writes the definition of the business service including every method, method argument, and script into the XML file.

You can also import a business service from an external XML file by clicking the Import Service button in the Business Service list applet.

Testing Your Business Service

You can use the Business Service Simulator to test your business services in an interactive mode.

To run the Business Service Simulator

- 1 From the application-level menu, choose View > Site Map > Business Service Administration > Business Service Simulator.

NOTE: The contents of the Simulator screen are not persistent. To save the data entered in the applets, click the Save To File button. This will save the data for the active applet in an XML file. The data can then be loaded into the next session from an XML file by clicking on the Load From File button.

- 2 In the Service Methods list applet, click New to add the business service you want to test.
- 3 Specify the Service Name and the Method Name.
- 4 Enter the number of iterations you want to run the business service.
 - Specify the input parameters for the Business Service Method in the Input Property Set applet. Multiple Input Property Sets can be defined and are identified by specifying a Test Case #.
 - If the Input Property Set has multiple properties, these can be specified by clicking on the glyph in the Property Name field. Hierarchical Property Sets can also be defined by clicking on the glyph in the Child Type field.
- 5 Click Run to run the business service.

The Simulator runs the specified number of iterations and loops through the test cases in order. If you have defined multiple input arguments, you can choose to run only one argument at a time by clicking Run On One Input.

The result appears in the Output Property Set applet.

NOTE: Once the Output arguments are created, you can click Move To Input to test the outputs as inputs to another method.

Example

Consider an example of a form on a corporate Web site. Many visitors during the day enter their personal data into the fields on the Web form. The field names represent arguments, whereas the personal data represent data. When the visitor clicks Submit on the form, the form's CGI script formats and sends the data by way of the HTTP transport protocol to the corporate Web server. The CGI script can be written in JavaScript, Perl, or another scripting language.

The CGI script may have extracted the field names and created XML elements from them to resemble the following XML tags.

```
First Name = <FirstName></FirstName>
Last Name = <LastName></LastName>
```

The CGI script may then have wrapped each data item inside the XML tags:

```
<FirstName>Hector</FirstName>
<LastName>Alacon</LastName>
```

To insert the preceding data into the Siebel database as a Contact, your script calls a business service that formats the XML input into a property set structure that the Siebel application recognizes.

Code Sample

An example of the code you need to write to create the property set may look something like this:

```
x = TheApplication.InvokeMethod("WebForm", inputs, outputs);

var svc; // variable to contain the handle to the Service
var inputs; // variable to contain the XML input
var outputs; // variable to contain the output property set
svc = TheApplication().GetService("EAI XML Read from File");

inputs = TheApplication().ReadEAIMsg("webform.xml");

outputs = TheApplication().NewPropSet();

svc.InvokeMethod("Read XML Hierarchy", inputs, outputs);
```

The following functions could be called from the preceding code. You attach the function to a business service in Siebel Tools:

```
Function Service_PreInvokeMethod(MethodName, inputs, outputs)

{

  if (MethodName=="GetWebContact")
  {
    fname = inputs.GetProperty("<First Name>");
    lname = inputs.GetProperty("<Last Name>");
    outputs.SetProperty("First Name",fname);
    outputs.SetProperty("Last Name", lname);
    return(CancelOperation);
  }
  return(ContinueOperation);
}

Function Service_PreCanInvokeMethod(MethodName, CanInvoke)

{
  if (MethodName=="GetWebContact")
  {
    CanInvoke ="TRUE";
    return (CancelOperation);
  }
  else
  {
    return (ContinueOperation);
  }
}
```

NOTE: You cannot pass a Business Object as an argument to a business service method.

This chapter describes Web Services, their uses, and how to create, implement, and publish Siebel Web Services. This chapter also provides examples of how to invoke an external Web Service and a Siebel Web Service.

Overview

Web Services combine component-based development and Internet standards and protocols that include HTTP, XML, Simple Object Application Protocol (SOAP), and Web Services Description Language (WSDL). Web Services can be reused regardless of how they are implemented. Web Services can be developed on any computer platform and in any development environment as long as they can communicate with other Web Services using these common protocols.

Web Services can be implemented in Siebel eBusiness applications as business services or workflow processes. The Siebel Web Services Framework can consume a WSDL document and create a proxy business service through the WSDL Import Wizard provided in Siebel Tools.

To specify the structure of XML used in the body of SOAP messages, Web Services use an XML Schema Definition (XSD) standard. The XSD standard describes an XML document structure in terms of XML elements and attributes. It also specifies abstract data types, and defines and extends the value domains.

Users or programs interact with Web Services by exchanging XML messages that conform to Simple Object Access Protocol (SOAP). For Web Services support, SOAP provides a standard SOAP envelope, standard encoding rules that specify mapping of data based on an abstract data type into an XML instance and back, and conventions for how to make remote procedure calls (RPC) using SOAP messages.

How the Web Services Dispatch Works

The Web Service Inbound Dispatcher is a business service that is called by an inbound transport server component (or Outbound Web Service Dispatcher business service) and analyzes input XML, converts XML data to business service method arguments, and calls the appropriate method for the appropriate service. After the called method has finished, the Web Service Inbound Dispatcher converts the output arguments to XML data, or creates a SOAP fault block (if there is an exception) and then returns the XML embedded in the SOAP envelope.

Publishing Siebel Web Services

To publish a Siebel Web Service, you can use the Web Services Administration view in the Siebel client to create a WSDL document based on a business service or workflow process.

Administration

The following procedure describes how to create a new Inbound Web Services record that will be used when generating a WSDL document.

To create a new Inbound Web Services record

- 1** Navigate to the Web Services Administration view.
- 2** Select Inbound Web Services from the Web Services Administration view.
- 3** In the Inbound Web Services List applet, create a new Inbound Web Services record.
 - a** Enter the namespace for your organization's Web Services in the Namespace column.
 - b** Enter the name of the inbound Web Service in the Name column.
 - c** In the Status field, select Active or Inactive.

NOTE: If the Web Service is inactive, then the external applications cannot invoke the Web Service. If the status is changed from one status to another, the server component requires a restart for the change to take effect.

- 4** Enter a description of the Web Service in the Comment column.

5 In the Service Ports List applet, create a new Inbound Service Ports record.

a Enter the name of the Web Service port in the Name column.

b Click the Type field and select a Port Type from the picklist.

If the required Port Type is not already available, click New to add a new Port Type and follow [Step c](#) through [Step f](#).

c Click New and select the implementation type (Business Service or Workflow).

d Select the implementation name (the business service or workflow that implements the port type).

e Name the new Port Type and click Save.

f Click Pick in the Port Type picklist to complete the process of adding a new Port Type.

g Enter the name of the business service or workflow process in the Business Service/Business Process Name column.

h Select a transport name for the protocol or queuing system for the Transport.

i Enter the address for the Siebel Server to listen for the Web Service.

For example, the default URL for HTTP in this case is:

```
URL = http://<webserver>/eai_<lang>/  
state.swe?SWEExtSource=WebService&SWEExtCmd=Execute&User  
Name=<username>&Password=<password>
```

where <lang> is the default language of Siebel Server and <webserver> is the machine name of the Siebel Web Server.

To specify the EAI MQSeries or EAI MSMQ Server Transports, the format is:

```
mq://<service>@<policy>  
msmq://<queue>@<machine>
```

- 6** Choose the default RPC encoding option for Web Service support.

NOTE: The Siebel application supports only one type of binding for each Web Service.

- 7** Enter a description of the Port in the Comment column.
- 8** In the Operations List applet, create a new Operations record for each business service or workflow process method that you want to publish.

- a** Enter the name of the Web Service operation to be specified in the generated WSDL file in the Name column.

The generated WSDL file will be used in [“Consuming External Web Services” on page 89](#).

- b** Select the name of the business service method in the Business Service Method column.

NOTE: If you chose Workflow Process as the Type of object to expose as a Web Service in [Step 5](#), the Business Service Method column should have a value of Run Process.

WSDL Document Generation

The following procedure describes how to generate a WSDL document once you have created a new Inbound Web Services record.

To generate a WSDL document

- 1** From the application-level menu, choose View > Site Map > Web Services Administration.
- 2** Choose Inbound Web Services to specify the Inbound Web Service you want to publish and click GenerateWSDL.
- 3** Specify the file system location where you want the exported WSDL document stored.

Once you have generated the WSDL document, you can publish it to a directory or to any other discovery mechanism.

Consuming External Web Services

To consume external Web Services, you can use the WSDL Import Wizard to read an existing WSDL document (provided in an XML or WSDL file) for one or more Web Service definitions. The Web Service definition is then added to the Siebel repository as a business service object.

WSDL Import Wizard

The following procedure describes how to use the wizard to read an existing WSDL document.

To read a WSDL document

- 1** Start Siebel Tools.
- 2** Create a new project and lock the project, or lock an existing project in which you want to create your integration object.
- 3** Choose File > New Object... to display the New Object Wizards.
- 4** Select the EAI tab, select the Web Service icon, and click OK.

The WSDL Import Wizard appears.

- 5** Provide information required to import the Web Service or Web Services definition into the Siebel repository.
- 6** Select the Project where you want the objects to be held after they are created from the WSDL document.
- 7** Specify the WSDL document that contains the Web Service or Web Services definition that you want to import.
- 8** Specify the file where you want to store the run-time data extracted from the WSDL document.
- 9** Specify the log file where you want errors, warnings, and other information related to the import process to be logged.
- 10** Click Next to view and verify a summary of your import information.
- 11** Click Finish to complete the process of importing the business service into the Siebel repository.

Administration

After you use the WSDL Import Wizard, the data that needs to go to the run-time database (the Web Services address) is exported to a file, and this file needs to be imported in the Outbound Web Services administration screen.

In the case of manual proxy definition, the run-time data also needs to be manually entered in the Outbound Web Service administration screen.

To create a new Outbound Web Services record

- 1** Navigate to the Web Services Administration view.
- 2** Select Outbound Web Services from the Web Services Administration view.
- 3** In the Outbound Web Services List applet, create a new Outbound Web Services record.
 - a** Enter the global namespace of the Web Service in the Namespace column.
 - b** Enter the name of the Web Service (as specified in the WSDL file) in the Name column.
 - c** In the Status field, select Active or Inactive.
- 4** Enter a description of the Web Service in the Comment column.

NOTE: If the Web Service was created through a WSDL file, then the name of the file should be inserted into this column by default.

- 5 In the Service Ports List applet, create a new Outbound Service Ports record.
 - a Enter the name of the Web Service port in the Name column.
 - b Select a type of proxy for the Port Type column.
 - c Select a transport name for the protocol or queuing system for the Transport.
 - d Enter the address for the Local Workflow and the Local Business Service so that the Siebel Server knows where to listen for the Web Service.

The address for the Local Web Service should be the name of the inbound port.

For example, the default URL for HTTP in this case is:

```
URL = http://<webserver>/eai_<lang>/  
state.swe?SWEExtSource=WebService&SWEExtCmd=Execute&User  
Name=<username>&Password=<password>
```

where <lang> is the default language of Siebel Server and <webserver> is the machine name of the Siebel Web Server.

To specify the EAI MQSeries or EAI MSMQ Server Transports, the format is:

```
mq://<service>@<policy>
```

```
msmq://<queue>@<machine>
```

- 6 Select whether the port uses SOAP document, SOAP RPC, or property set Binding, which is used when the input to the proxy will be forwarded without changes.

NOTE: The Siebel application supports only one type of binding for each Web Service.

- 7** Enter a description of the Port in the Comment column.
- 8** In the Operations Bindings applet, create a new Operations record.
 - a** Enter the name of the Web Service in the Name column.
 - b** Enter the name of the Binding Property in the Binding Property column; for example, SOAPAction.
 - c** Enter the value of the Binding Property in the Binding Value column; for example, CreateOrder.

NOTE: Before importing the run-time definition of the external Web Service, the Siebel Server (or Mobile Web Client) should be restarted with a recompiled version of the .srf file that includes the new objects created by the Web Services Import Wizard. If the .srf file has not been updated, importing the definition of the external Web Services will cause an error stating the referenced proxy does not exist in the .srf file.

To import the WSDL file

- 1** Navigate to the Web Services Administration view.
- 2** Select Outbound Web Services from the Web Services Administration view.
- 3** Click Import to display the EAI Web Service Import dialog box.
- 4** Specify the export file created with the Web Services Import Wizard.
- 5** Click Import.

Examples

The following two examples show sample flows of how to invoke an external Web Service from a Siebel application and also how to invoke a Siebel Web Service from an external application.

Invoking an External Web Service Using Workflow or Scripting

Figure 25 shows the process for invoking an external Web Service. The flow shows the various conversion steps from the external Web Service, through the Siebel application, to the external application and back.

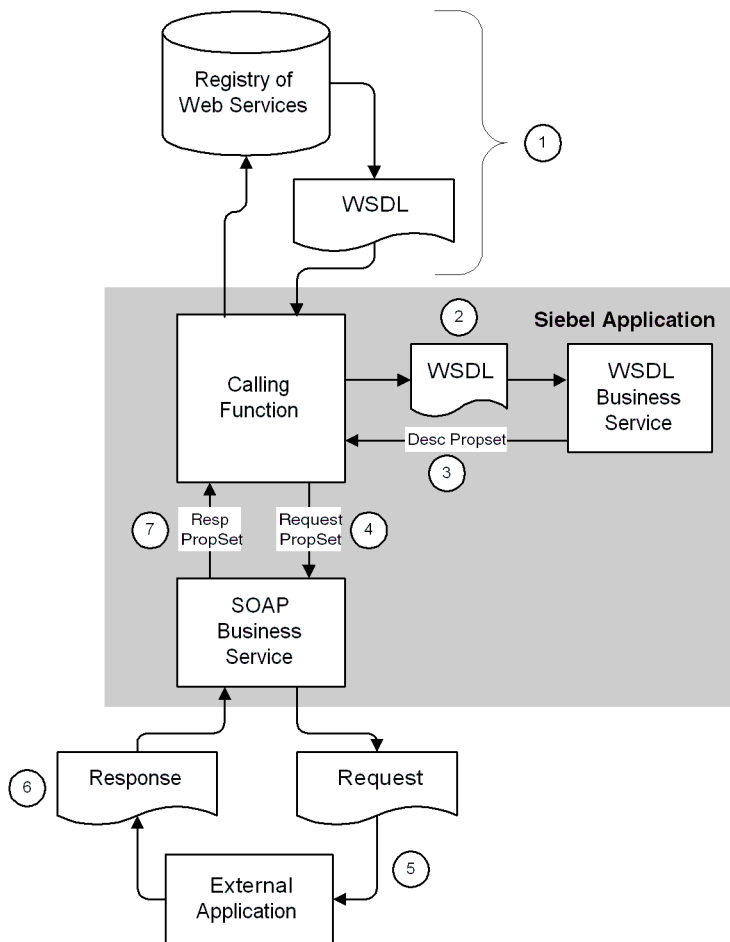


Figure 25. Invoking an External Web Service

- 1** The WSDL file is imported by the WSDL Import Wizard and converted to one or more outbound proxies and integration objects.
- 2** The Web Service run-time parameters (port, address, and operations) are generated by the WSDL Import Wizard, and are imported into the Outbound Web Services view.
- 3** The property set is passed from calling code to the Outbound Web Service proxy.
- 4** The property set is converted to an outbound SOAP request.
- 5** The SOAP request is sent to the external application.
- 6** The SOAP response is converted into a property set.
- 7** The Descriptor Property Set reads the contents of the property set.

Invoking a Siebel Web Service Using an External Application

Figure 26 shows the process for invoking a Siebel Web Service from an external application.

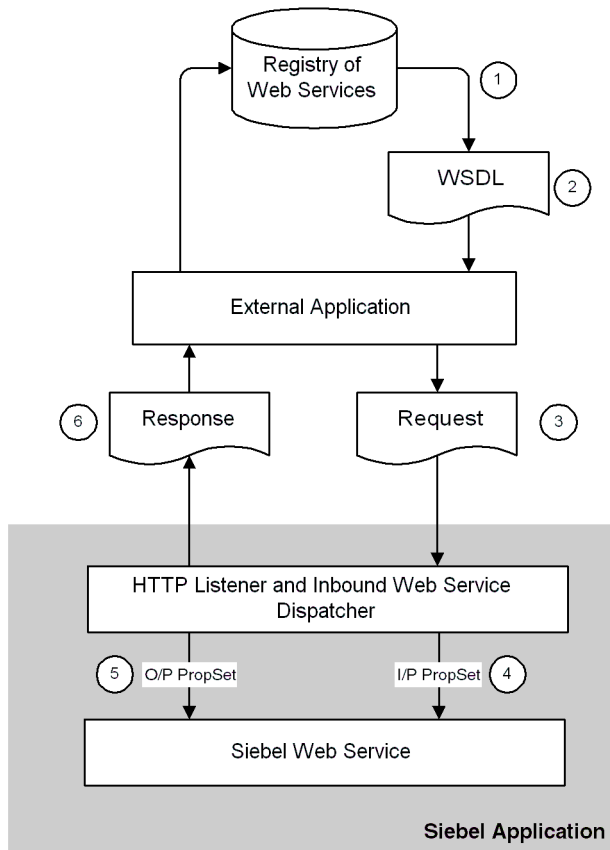


Figure 26. Invoking a Siebel Web Service

- 1** A WSDL document is published based on an active Web Service as defined in the Inbound Web Service view.
- 2** A proxy is created based on the WSDL definition and this proxy is implemented.

NOTE: This process will vary for each Web Services provider.

- 3** The external application sends a SOAP request using HTTP.
- 4** The Inbound Web Service Dispatcher converts the SOAP request to several property sets and invokes the corresponding business service or business process.
- 5** The result of the business service (output property set) is converted into a SOAP response document and then returned to the external application.
- 6** The response is deciphered by the external application based on the WSDL definition.

The EAI Siebel Adapter

5

This chapter describes the functionality of the EAI Siebel Adapter and the different methods and arguments you can use with the EAI Siebel Adapter to manipulate the data in the Siebel database.

EAI Siebel Adapter Overview

The EAI Siebel Adapter is a general-purpose integration business service that allows you to:

- Read Siebel business objects from the Siebel database into integration objects.
- Write an integration object whose data originates externally into a Siebel business object.
- Update multiple corresponding top-level parent business component records with data from one XML file. For examples, see [“XML Examples” on page 110](#).

NOTE: EAI Message is considered to be one transaction. The transaction is committed when there is no error. If there is an error, the transaction is aborted and rolled back.

The EAI Siebel Adapter is based on the *CSSEAISiebelAdapterService* class.

EAI Siebel Adapter Methods

The EAI Siebel Adapter supports the following methods:

- QueryPage
- Query
- Synchronize
- Upsert
- Update
- Insert
- Delete
- Execute

EAI Siebel Adapter Method Arguments

Each of the EAI Siebel Adapter methods takes arguments that allow you to specify required and optional information to the adapter. You can locate the arguments for each method in [Table 5](#).

Table 5. EAI Siebel Adapter Method Arguments

Argument	QueryPage	Query	Sync	Upsert	Update	Insert	Delete	Execute
IntObjectName							Input	Input
NumOutputObjects	Output	Output	Output	Output	Output	Output	Output	Output
OutputIntObjectName	Input	Input						Input
PrimaryRowId		Input	Output	Output	Output	Output	Input	Input/ Output
QueryByUserKey		Input						Input
DeleteByUserKey							Input	Input
ErrorOnNonExistingDelete							Input	Input

Table 5. EAI Siebel Adapter Method Arguments

Argument	QueryPage	Query	Sync	Upsert	Update	Insert	Delete	Execute
SiebelMessage	Input/ Output	Input/ Output	Input/ Output	Input/ Output	Input/ Output	Input/ Output	Input/ Output	Input/ Output
SearchSpec	Input	Input					Input	Input
StatusObject			Input	Input	Input	Input	Input	Input
MessageId	Input	Input	Input	Input	Input	Input	Input	Input
BusObjCacheSize	Input	Input	Input	Input	Input	Input	Input	Input
LastPage	Output							Output
NewQuery	Input							Input
PageSize	Input							Input
StartRowNum	Input							Input
ViewMode	Input	Input	Input	Input	Input	Input	Input	Input
SortSpec	Input							Input

Table 6 describes each argument of the EAI Siebel Adapter methods.

Table 6. EAI Siebel Adapter Method Arguments

Argument	Display Name	Description
IntObjectName	Integration Object Name	The name of the integration object that is to be deleted.
NumOutputObjects	Number of Output Integration Objects	Number of output integration objects.
OutputIntObjectName	Output Integration Object Name	The name of the integration object that is to be output.
PrimaryRowId	Object Id	<p>The PrimaryRowId refers to the ID field in the Business Component, Row_ID at the table level.</p> <p>PrimaryRowId is only returned as an output argument if you are passing in one integration object instance. If you are passing multiple integration object instances then this argument is not returned as an output argument. To obtain the ID field when multiple Integration Objects are processed, use the StatusObject argument.</p>

Table 6. EAI Siebel Adapter Method Arguments

Argument	Display Name	Description
QueryByUserKey	Query By Key	A Boolean argument. Forces the EAI Siebel Adapter to only use the User Keys to perform query.
DeleteByUserKey	Delete By User Key	A Boolean argument. Forces the EAI Siebel Adapter to only use the User Keys to identify a record.
ErrorOnNonExistingDelete	Error On Non Existing Delete	A Boolean argument. Determines whether or not the EAI Siebel Adapter should abort the operation if no match is found.
SiebelMessage	Siebel Message	The input or the output Integration Object Instance.
SearchSpec	Search Specification	This argument allows you to specify complex search specifications as free text in a single method argument. See “Search Specification” on page 114 for details.
StatusObject	Status Object	This argument tells the EAI Siebel Adapter whether or not to return a status message.
MessageId	Message Id	The MessageId can be used to specify the ID for the generated message. By default, the EAI Siebel Adapter generates a unique ID for each message. However, if you want to use the workflow process instance id then you can use this argument to specify the desired ID.
BusObjCacheSize	Business Object Cache Size	Default is 5. Maximum number of Business Objects instances cached by the current instance of the EAI Siebel Adapter. If set to zero, then the EAI Siebel Adapter does not use the cache.
LastPage	Last Page	Boolean indicating whether or not the last record in the query result set has been returned.
PageSize	Page Size	Default is 10. Indicates the maximum number of Integration Object Instances to be returned. If you change this value, the value True will be used for NewQuery.
SortSpec	Sort Specification	Default is the SortSpec of the underlying Business Component. This argument allows you to specify complex sort criteria as a free text in a single method argument, using any Business Component fields and standard Siebel sort syntax. For examples, see <i>Siebel Tools Reference, MidMarket Edition</i> .
StartRowNum	Starting Row Number	Default is 0. Indicates the row (by its order number) in the result set of the first root component record to be returned.

Table 6. EAI Siebel Adapter Method Arguments

Argument	Display Name	Description
ViewMode	View Mode	Default is All. Visibility mode to be applied to the Business Object. Valid values are: Manager, Sales Rep, Personal, Organization, Sub-Organization, Group, Catalog, and All. Note that the ViewMode user property on the Integration Object has priority over the ViewMode method argument.
NewQuery	New Query	Default is False. Boolean indicating whether or not a new query should be executed. If set to True, a new query is executed flushing the cache for that particular integration object. The True setting is used if the PageSize value changes.

QueryPage Method

The QueryPage method returns a subset of records for a query request, as specified by the PageSize and the StartRowNum method arguments. You can use additional parameters such as SearchSpec, SortSpec, and ViewMode respectively to modify the SearchSpecification, SortSpecification, and visibility mode used by the method.

You can use the QueryPage method to limit the number of records returned by the query to improve performance. Even though the QueryPage returns a limited number of records, it keeps the data in the cache which you can then retrieve by calling the EAI Siebel Adapter with the same PageSize and the NewQuery argument set to False.

Query Method

You pass the Query method a Query By Example (QBE) integration object instance, a Primary Row Id, or a Search Specification. The adapter uses this input as criteria to query the base business object and to return a corresponding integration object instance. For example, to query Contact records with first name *David* you need to pass the following required input arguments to the Query method of EAI Siebel Adapter:

- SiebelMessage.IntObjName with value set to *Test Contact*
- SiebelMessage.ListOfTest Contact.Contact.First Name with value set to *David*

Now, if you need to further limit the output based on a value in the child component of the Test Contact (for example, to only query the Contact records with first name *David* and Action Type of *Call*), then you need the following required input arguments:

- SiebelMessage.ListOfTest Contact.Contact.First Name with Value set to *David*
- SiebelMessage.IntObjName with value set to *Test Contact*
- SiebelMessage.ListOfTest Contact.Contact.ListOfAction.Action.Type, with Value set to *Call*

Note that this still returns the contacts with the first name *David*, even if they do not have an activity of type *Call*, but it does not list their activities.

NOTE: When using the EAI Siebel Adapter in your workflow process, to query all the business component records, you do not need to specify any value in the Object Id process property of the workflow process. Not identifying any value in the Object Id works as a wildcard. But if you want to query Siebel data using the EAI Siebel Adapter with the Query method and a property set containing a query by example search criteria, then all the fields that make up the user key for the underlying integration object component must exist in the property set. You can use an asterisk (*) as a wildcard for each one of the fields, but all of the fields must exist. If you leave any of the user keys blank, then no record is returned.

Synchronize Method

You can use the Synchronize method to make the values in a business object instance match those of an integration object instance. This operation can result in updates, inserts, or deletes on business components. Some rules apply to the results of this method:

- If a child component is not present in the integration object instance, the corresponding business component rows are left untouched.
- If a child component is present in the integration object instance, but contains no instances so that there is only an empty container, then records in the corresponding business component are deleted.

- If a child component is present in the integration object instance, and contains some instances, the business component rows corresponding to the instances are updated or created and any business component row that does not have a corresponding integration component instance is deleted.
- The Sync method applies the operation sequentially to each root Integration Component (because each previous Integration Component is written to the database) but does not do this for any child Integration Component.

NOTE: The Synchronize method only updates the fields specified in the integration component instance.

Upsert Method

The Upsert method is similar to the Synchronize method with the following exceptions:

- The Upsert method does not delete any records.
- The Upsert method applies the operation sequentially to each root Integration Component (because each previous Integration Component is written to the database) but does not do this for any child Integration Component.

Update Method

This method is similar to the Upsert method except that the EAI Siebel Adapter errors out if no records are found; otherwise, it updates the matching record.

Insert Method

The method is also similar to the Sync method with the exception that the EAI Siebel Adapter errors out if a match is found.

Delete Method

You can delete one or more records in a business component that is mapped to the root integration component, given an integration object. A business component is deleted as specified by an integration object. If you specify any child integration component instances, then the fields of an integration component instance are used to query a business component.

NOTE: To have the EAI Siebel Adapter perform a delete operation, define an integration object that contains the minimum fields on the primary business component for the business object. The EAI Siebel Adapter attempts to delete matching records in the business component before deleting the parent record.

Execute Method

The Execute method can be specified on the EAI Siebel Adapter to perform combinations of various operations on components in an integration object instance. This method uses the following operations:

- delete
- upsert
- sync (default operation)
- query
- update
- insert
- updatesync
- insertsync
- none

NOTE: A none operation is equivalent to an operation not being specified. This is equivalent to a sync operation.

These operations perform the same tasks as the related methods. For example, the delete operation makes the EAI Siebel Adapter delete the Business Component record matched to the particular Integration Component instance. However, what will be done to the children depends on the combination of the parent operation and the child operation. Operations including the word *sync* in the name cause deletion of unmatched child records, whereas *update*, *insert*, and *upsert* do not delete any children. [Table 7](#) gives the overview of the six related operations.

Table 7. EAI Siebel Adapter Execute Method Operations

Operation	Error on Match Found	Error on Match Not Found	Delete Unmatched Children
upsert	No	No	No
sync	No	No	Yes
update	No	Yes	No
updatesync	No	Yes	Yes
insert	Yes	No	No
insertsync	Yes	No	Yes

NOTE: You should use the Execute method when you need to mix different operations on different components within a single integration object; otherwise, you should use the Upsert, Synchronize, Delete, or Query methods.

An XML document sent to a Siebel application can include operations that describe whether a particular data element needs to be inserted, updated, deleted, synchronized, and so on. These operations can be specified as an attribute at the component level. They cannot be specified for any other element.

Execute Method Operation

Specify an attribute named operation, in lowercase, to the component's XML element. The legal values for this attribute are upsert, sync, delete, query, update, insert, updatesync, insertsync, and none. If the operation is not specified on the root component, the sync operation is used as the default.

NOTE: Specifying operation within <ListOf> tag is not supported. For details on the <ListOf> tag, see *XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition*.

Supported Operations for the Parent and Its Child Components

[Table 8](#) lists the supported operations for a child component based on its parent component's operation.

Table 8. Supported Operations

Parent Operation	upsert	sync	update	update sync	insert	insert sync	query	delete
upsert	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
sync	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
update	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
updatesync	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
insert	Yes	Yes	Yes	Yes	Yes	Yes	No	No
insertsync	Yes	Yes	Yes	Yes	Yes	Yes	No	No
query	No	No	No	No	No	No	Yes	No
delete	No	No	No	No	No	No	No	Yes

XML Examples

The following XML example demonstrates using upsert and delete operation to delete a particular child without updating the parent.

```
<SiebelMessage MessageId=" " MessageType="Integration Object"
IntObjectName="Sample Account">

  <ListofSampleAccount>

    <Account operation="upsert">

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

      <Location>HQ-Distribution</Location>

      <Organization>North American Organization</Organization>

      <Division/>

      <CurrencyCode>USD</CurrencyCode>

      <Description>This is the key account in the AK Parker
Family</Description>

      <HomePage>www.parker.com</HomePage>

      <LineofBusiness>Manufacturing</LineofBusiness>

      <ListOfContact>

        <Contact operation="delete">

          <FirstName>Stan</FirstName>

          <JobTitle>Senior Mgr of MIS</JobTitle>

          <LastName>Graner</LastName>

          <MiddleName>A</MiddleName>

          <PersonalContact>N</PersonalContact>

          <Account>A. K. Parker Distribution</Account>

          <AccountLocation>HQ-Distribution</AccountLocation>

        </Contact>

      </ListOfContact>

    </Account>

  </ListofSampleAccount>

</SiebelMessage>
```

```

        </ListOfContact>

    </Account>

</ListofSampleAccount>

</SiebelMessage>

```

The following example illustrates updating multiple corresponding top level parent business component records with one XML file.

```

<SiebelMessage MessageId=" " MessageType="Integration Object "
IntObjectName="Transaction">

<ListofTransaction>

    <Transaction>

        <Field1>xxxx</Field1>

        <Field2>yyyy</Field2>

        .....

    </Transaction>

    <Transaction>

        <Field1>aaaa</Field1>

        <Field2>bbbb</Field2>

        .....

    </Transaction>

    .....

</ListofTransaction>

</SiebelMessage>

```

MVGs in the EAI Siebel Adapter

Multi-value groups (MVGs) are mapped to integration components of type MVG or MVGAssociation. MVGs are specified by creating a user property called MVG on the integration component and setting it to Y. For details on MVGs, see [Chapter 1, “About Integration Objects.”](#)

Also, an integration component instance that corresponds to a primary MVG is denoted by a field *IsPrimaryMVG* set to Y. This field does not need to have a corresponding integration component field in the metadata.

Each MVG that appears on the client UI is mapped to an integration component. For example, in the Orders - Sales Orders - Terms screen, you see an account address, a bill-to address, and a ship-to address. Each MVG maps to a separate integration component definition.

NOTE: Setting a primary record in an MVG is supported only when the *Auto Primary* property of the underlying MVLink is specified as *Selected* or *None*. If *Auto Primary* is defined as *Default*, then the EAI Siebel Adapter cannot set the primary. This happens because the Object Manager does not allow explicitly setting the primary unless it is a visibility MVG component where records in the component are used by visibility rules to determine who is going to see the related records. For details, see *Siebel Tools Reference, MidMarket Edition*.

Setting a Primary Address for an Account

You have an account with multiple shipping addresses in a Siebel application. None of these addresses are marked as the primary address for the account and you want to select one of them as the primary shipping address.

To specify a address as a primary

- 1 Create your XML file and insert `<IsPrimaryMVG= 'Y'>` before the address you want to identify as the primary address for the account as shown below:

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

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

- <SiebelMessage MessageId="1-69A" IntObjectFormat="Siebel
Hierarchical" MessageType="Integration Object"
IntObjectName="Sample Contact">
```



```

- <ListOfSampleContact>
- <Contact>
  <FirstName>Pal888</FirstName>
  <IntegrationId>65454398</IntegrationId>
  <JobTitle>Manager</JobTitle>
  <LastName>John888</LastName>
  <MiddleName />
  <PersonUID>1-Y88H</PersonUID>
  <PersonalContact>N</PersonalContact>
- <ListOfContact_Position>
- <Contact_Position IsPrimaryMVG="Y">
  <EmployeeFirstName>Siebel</EmployeeFirstName>
  <EmployeeLastName>Administrator</EmployeeLastName>
  <Position>Siebel Administrator</Position>
  <RowStatus>N</RowStatus>
  <SalesRep>SADMIN</SalesRep>
</Contact_Position>
</ListOfContact_Position>
</Contact>
</ListOfSampleContact>
</SiebelMessage>.

```

- 2 Use the Upsert or Sync method to update the account.

Search Specification

The SearchSpec input method argument is applicable to Query, Delete, and Execute methods. This method argument allows you to specify complex search specifications as free text in a single method argument. Expressions within a single integration component are restricted only by the Siebel Query Language supported by the Object Manager. For example, given an integration object definition with two integration components, Account as the root component and Contact as the child component, the following search specification is allowed:

```
([Account.Site] LIKE "A*" OR [Account.Site] IS NULL) AND  
[Contact.PhoneNumber] IS NOT NULL
```

This search specification queries accounts that either have a site that starts with the character A, or do not have a site specified. In addition, for the queried accounts, it queries only those associated contacts that have a phone number.

NOTE: The AND operator is the only allowed operator among different integration components. You use DOT notation to refer to integration components and their fields.

You can include the child integration component in a search specification only if its parent components are also included. For example, using the same integration object definition as in previous examples, the following query:

```
[Contact.PhoneNumber] IS NOT NULL
```

queries every account. Then for each account, it queries only contacts that have a phone number. If you want to query only accounts that are associated with contacts that have a phone number specified, then you need to create another business object, and an integration object based on that business object, which has contact as a root component, and account as its child component.

EAI Siebel Adapter Concurrency Control

The EAI Siebel Adapter supports concurrency control to guarantee data integrity and avoid overriding data by simultaneous users or integration processes. To do so, the EAI Siebel Adapter uses the Integration Component Key called Modification Key.

Modification Key

A Modification Key is an Integration Component Key of the type *Modification Key*. A Modification Key is a collection of fields that together should be used to verify the version of an integration component instance. Typically, Modification Key fields are Mod Id fields for the tables used. Multiple Modification Key fields may be needed because a business component may be updating multiple tables, either as extension tables or through implicit or explicit joins.

EAI Siebel Adapter methods (Insert, Update, Synchronize, Upsert) check for the existence of a Modification Key. If no Modification Key is specified in the integration component definition or if Modification Key fields are not included in the XML request, the EAI Siebel Adapter does not check for the record version and proceeds with the requested operation. If a valid Modification Key is found but the corresponding record can not be found, the EAI Siebel Adapter assumes that the record has been deleted by other users and returns the error `SSASqlErrWriteConflict`.

If a valid Modification Key as well as the corresponding record can be found, the EAI Siebel Adapter checks if the Modification Key fields in the XML request and the matched record are consistent. If any of the fields are inconsistent, the EAI Siebel Adapter assumes that the record has been modified by other users and again returns the error `SSASqlErrWriteConflict`. If all the fields are consistent, the EAI Siebel Adapter proceeds with the requested operation.

Modification IDs

To determine which Mod Id fields need to be used as part of a Modification Key, you expose Mod Id fields for tables whose columns may be updated by that Integration Object. In some situations you might need to add corresponding Integration Component Fields as well as Business Component Fields.

NOTE: The EAI Siebel Adapter can update base and extension tables. It may even update joined table columns through picklists that allow updates.

Modification ID for a Base Table

Integration Component field Mod Id, for a base table, is created by the Integration Object Wizard but you need to make sure it is active if it is needed for your business processes.

Modification ID for an Extension Table

An extension table's Mod Id field is accessible as *extension table name*.Mod Id in the Business Component (for example, S_ORG_EXT_X.Mod Id). However, you need to manually add it to the Integration Object definition if your business processes requires use of this field. This can most easily be done by copying 'Mod Id' field and changing the properties.

Modification ID for a Joined Table

A joined table's Mod Id field needs to be manually added in both business component and integration object definitions. Business Component Mod Id fields for joined tables should be:

- Prefixed with CX string preferably followed by the name of the join
- Joined over the correct join
- Have MODIFICATION_NUM specified as underlying column of type DTYPE_INTEGER

MVG and MVGAssociation Integration Components

For integration components that are of type MVG or MVG Association, in addition to the above steps, for each Modification ID integration component field you need to create user properties MVGFieldName and AssocFieldName, respectively, and set the name of the field as the value.

Language-Independent Code

If the user Property `AllLangIndependentVals` is set to Y at the integration object level, then EAI Siebel Adapter uses the language-independent code for its LOVs.

In the outbound direction, for example the Query method, if the `AllLangIndependentVals` is set to Y then the EAI Siebel Adapter translates the language-dependent values in the Siebel database to their language-independent counterpart based on the List Of Values entries in the database.

In the inbound direction, for example the Synchronize method, if the `AllLangIndependentVals` is set to Y then the EAI Siebel Adapter expects language-independent values in the input message, and translates them to language-dependent values based on the current language setting and the entries in the List Of Values in the database.

NOTE: The LOV-based fields are always validated using language-dependent values. Using language independent values adversely affects the performance by 5 %-10 %; however, this applies only to LOV-bound fields. In typical scenario where about one fifth of the fields in the Integration Object are LOV-bound, the overall performance penalty would be between 1 % and 2 %.

LOV Translation

Siebel Applications distinguish two types of lists of values (LOV):

- Multilingual LOV (MLOV) which stores a language independent code (LIC) in the Siebel database that gets translated to a language dependent value (LDV) for active language by Object Manager. MLOVs are distinguished by having 'Translation Table' specified on the Column definition.
- Single-language LOV for which the LDV for the current language is stored in the Siebel database.

The Boolean Integration Object User Property `AllLangIndependentVals` determines whether the Siebel Adapter should use LDV or LIC. For details, see [Table 3 on page 59](#).

NOTE: Using this feature impacts performance, but allows easier communication between systems that operate on different languages.

Siebel eAI and Run Time Events

Siebel allows triggering workflows based on Run Time Events or Workflow Policies.

Run Time Events. Siebel eAI supports triggering workflows based on run time events such as Write Record, which gets triggered whenever a record is written. If you use both the EAI Siebel Adapter to import data into Siebel and run time events, you should pay attention to the following:

For the EAI Siebel Adapter, one call to Siebel Adapter with an input message is a transaction. Within a transaction, Siebel Adapter makes multiple Write Record calls. At any point in the transaction, if the EAI Siebel Adapter encounters a problem the transaction is rolled back entirely. However, if you have specified events to trigger at Write Record, such events are invoked as soon as the EAI Siebel Adapter makes Write Record calls even though Siebel Adapter may be in the middle of a transaction. If you have export data workflows triggered on such events, this may lead to exporting data from Siebel applications that is not committed in Siebel applications and may get rolled back. It is also possible that your events get triggered when the record is not completely populated, which leads to situations that are not handled by your specified event processing workflow.

It is possible to avoid the effects of this interaction between the EAI Siebel Adapter and Run Time Events by using the business service EAI Transaction Service to figure out if a transaction (typically, Siebel Adapter) is in process. You may then want to skip processing that is not desirable when Siebel Adapter is in process.

For example, suppose you have a workflow to export Orders from Siebel applications that is triggered whenever the Order record is written. You also import Orders into Siebel applications using EAI. In such a situation, you do not want to export Orders while they are being imported because the import may get aborted and rolled back. You achieve this using the business service EAI Transaction Service as the first step of the export workflow. If you find that a transaction is in process you can branch directly to the end step.

Workflow Policies. In addition to Run Time Events, Siebel applications also support Workflow Policies as a triggering mechanism for workflows. You can use workflow policies instead of run time events to avoid the situation discussed above. You should use Workflow Policies instead of Run Time Events when possible.

Siebel eAI and File Attachments

6

This chapter describes the file attachment functionality of Siebel EAI using MIME or inline XML.

Overview

Siebel eAI supports file attachments for exchanging business documents such as sales literature, activity attachments, and product defect attachments with another Siebel instance or an external system.

For example, if you are exchanging service requests with another application or partner, you can include attachments associated with the service request—such as screen captures, email, log files, and contract agreements—in the information being exchanged. Siebel eAI support for file attachments allows comprehensive integration.

In order to use file attachments you first need to create Integration Objects. For details, see [Chapter 1, “About Integration Objects,”](#) and [Chapter 2, “Creating and Maintaining Integration Objects.”](#)

Siebel eAI offers the choice of integrating file attachments using MIME (the industry standard for exchanging multi-part messages), or including the attachment within the body of the XML document, referred to as an inline XML attachment. You should consider using inline XML attachments when integrating two instances of Siebel applications using file attachments.

Exchange of Attachments with External Applications

Siebel eAI supports bidirectional attachments exchange with external applications using the following two message types:

- **MIME (Multipurpose Internet Mail Extensions).** MIME is the industry standard for exchanging multipart messages. The first part of the MIME message is an XML document representing the business object being exchanged and attachments to the object are included as separate parts of the multipart message. MIME is the recommended choice for integrating Siebel applications with other applications.
- **Inline XML attachments (Inline Extensible Markup Language).** With inline XML attachments, the entire business object you are exchanging, including any attachments, is sent as a single XML file. In this case, attachments are included within the body of the inline XML attachment. Inline XML attachments should be considered when integrating two instances of Siebel applications using file attachments. For details, see *XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition*.

Using MIME Messages to Exchange Attachments

To send or receive file attachments using MIME messages, Siebel eAI uses the MIME Hierarchy Converter and MIME Doc Converter. The following checklist shows the high-level procedures you need to perform to use MIME to exchange attachments between Siebel applications and another external system.

Checklist

- | | |
|--------------------------|---|
| <input type="checkbox"/> | Create an integration object using the EAI Siebel Wizard.
For details, see “Creating the Integration Object” on page 122 . |
| <input type="checkbox"/> | Create an inbound or outbound Workflow process.
For details, see “Creating Workflow Processes Examples” on page 123 . |
| <input type="checkbox"/> | Test your workflow process using Workflow Simulator.
For details, see “The EAI MIME Hierarchy Converter” on page 128 . |

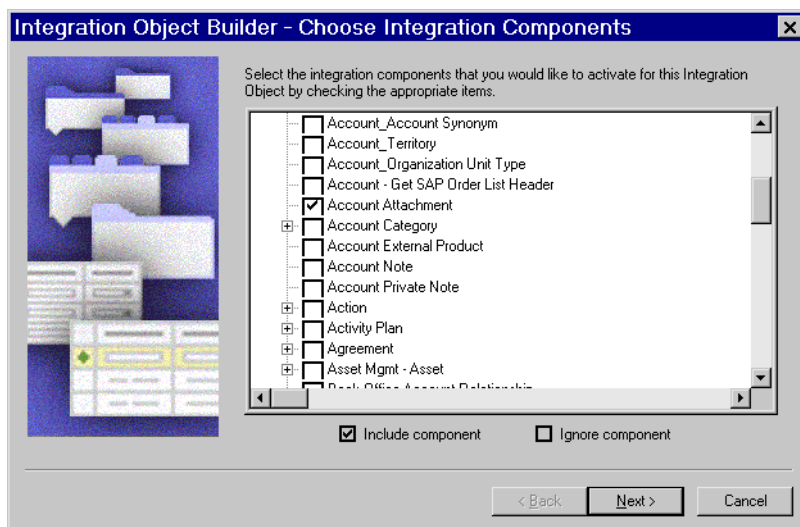
Creating the Integration Object

The following procedure guides you through the steps of creating an integration object.

To create a new Siebel integration object

- 1 Start Siebel Tools.
- 2 Create a new project and lock the project, or lock an existing project in which you want to create your integration object.
- 3 Choose File > New Object... to display the New Object Wizards.
- 4 Select the EAI tab, select the Integration Object icon, and click OK.

NOTE: When creating your integration object you need to select the Attachment integration object. The following figure illustrates this when the source object is Account.



- 5 Click Next to see a list of the warnings and errors generated by the Integration Object Builder.

- 6 Review and take necessary actions to address the issue.
- 7 Click Finish to complete the process of building the integration object.
- 8 In the Object Explorer, select Integration Object > Integration Component > Integration Component Field object.

The Integration Component and Integration Component Field applets appear.
- 9 Select the XXX_Attachment Component and the Attachment Id Component fields, and verify that the Data Type for the Attachment Id field is set to DTYPE_ATTACHMENT.
- 10 Compile the .srf file and copy it to the object directory under your Siebel Server directory as well as under your Tools directory.

NOTE: You need to stop the services before copying the .srf file. For details on the .srf file, see *Siebel Tools Reference, MidMarket Edition*.

Creating Workflow Processes Examples

Depending on whether you are preparing for an outbound or an inbound attachment exchange, you need to design different workflow process as described in the following two procedures.

Outbound Workflow Process

To process the attachment for an outbound request you need to create a workflow process to query the database, convert the Integration Object and its attachments into a MIME hierarchy, and then create a MIME document to send to the File Transport business service.

To create an outbound workflow process

- 1 Navigate to Workflow Process Designer.
- 2 Create a workflow process consisting of Start, End, and four Business Services. Set up each Business Service according to the task it needs to accomplish.
- 3 Define your process properties.

- 4 Set workflow process properties when you need a global property for the entire workflow. The first business service queries the Account information from the database using the EAI Siebel Adapter business service with the Query method. This step requires the following input and output arguments.

Input Argument	Type	Value	Property Name	Property Data Type
Output Integration Object Name	Literal	Sample Account		
SearchSpec	Process Property		SearchSpec	String

Property Name	Type	Output Argument
SiebelMessage	Output Argument	Siebel Message

NOTE: For more information on using the EAI Siebel Adapter, see [Chapter 5, “The EAI Siebel Adapter.”](#)

- 5 The second business service in the workflow converts the Account integration object and its attachments to a MIME hierarchy using the EAI MIME Hierarchy Converter business service with the SiebelMessage to MIME Hierarchy method. This step requires the following input and output arguments.

Input Argument	Type	Property Name	Property Data Type
Siebel Message	Process Property	SiebelMessage	Hierarchy

Property Name	Type	Output Argument
MIMEHierarchy	Output Argument	MIME Hierarchy

NOTE: For more information on the EAI MIME Hierarchy Converter, see [“The EAI MIME Hierarchy Converter” on page 128.](#)

- 6** The third business service of the workflow converts the MIME hierarchy to a document to be sent to File Transport business service. This step uses the EAI MIME Doc Converter business service with the MIME Hierarchy To MIME Doc method. This step requires the following input and output arguments.

Input Argument	Type	Property Name	Property Data Type
MIME Hierarchy	Process Property	MIMEHierarchy	Hierarchy

Property Name	Type	Output Argument
MIMEMsg	Output Argument	MIME Message

NOTE: For more information on the EAI MIME Doc Converter, see [“The EAI MIME Doc Converter” on page 130](#).

- 7** For the final step, you need to set up the last business service of the workflow to write the information into a file using the EAI File Transport business service with the Send method. This step requires the following input arguments.

Input Argument	Type	Value	Property Name	Property Data Type
Message Text	Process Property		MIMEMsg	String
File Name	Literal	c:\temp\account.txt		

NOTE: For details on File Transport, see *Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition*.

Inbound Workflow Process Example

To process the attachment for an inbound request, you need to create a workflow process to read the content from a file, convert the information into a Siebel Message, and send to the EAI Siebel Adapter to update the database accordingly.

To create an inbound workflow process

- 1 Navigate to Workflow Process Designer.
- 2 Create a workflow process consisting of Start, End and four Business Services. Set up each Business Service according to the task it needs to accomplish.
- 3 Define your process properties.

Set workflow process properties when you need a global property for the entire workflow.

- 4 The first business service in the workflow reads the Account information from a file using the EAI File Transport business service with Receive method. This step requires the following input and output arguments.

Input Argument	Type	Value
File Name	Literal	c:\temp\account.txt

Property Name	Type	Output Argument
MIMEMsg	Output Argument	Message Text

NOTE: For details on File Transport, see *Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition*.

- 5** The second business service of the workflow converts the Account information to a MIME hierarchy using the EAI MIME Doc Converter business service with the MIME Doc to MIME Hierarchy method. This step requires the following input and output arguments.

Input Argument	Type	Property Name	Property Data Type
MIME Message	Process Property	MIMEMsg	String

Property Name	Type	Output Argument
MIMEHierarchy	Output Argument	MIME Hierarchy

- 6** The third business service of the workflow converts the MIME hierarchy to a document and sends it to the EAI Siebel Adapter business service. This step uses the EAI MIME Hierarchy Converter business service with the MIME Hierarchy to Siebel Message method. This step requires the following input and output arguments.

Input Argument	Type	Property Name	Property Data Type
MIME Hierarchy	Process Property	MIMEHierarchy	Hierarchy

Property Name	Type	Output Argument
SiebelMessage	Output Argument	Siebel Message

- 7** The last step of the workflow writes the information into the database using the EAI Siebel Adapter business service with the Insert or Update method. This step requires the following input argument.

Input Argument	Type	Property Name	Property Data Type
Siebel Message	Process Property	SiebelMessage	Hierarchy

The EAI MIME Hierarchy Converter

The EAI MIME Hierarchy Converter transforms the Siebel Message into a MIME (Multipurpose Internet Mail Extensions) hierarchy for outbound integration. For inbound integration, it transforms the MIME Hierarchy into a Siebel Message.

Outbound Integration

The EAI MIME Hierarchy Converter transforms the input Siebel Message into a MIME Hierarchy. [Figure 27](#) illustrates the Siebel Message of a sample Account with attachments.

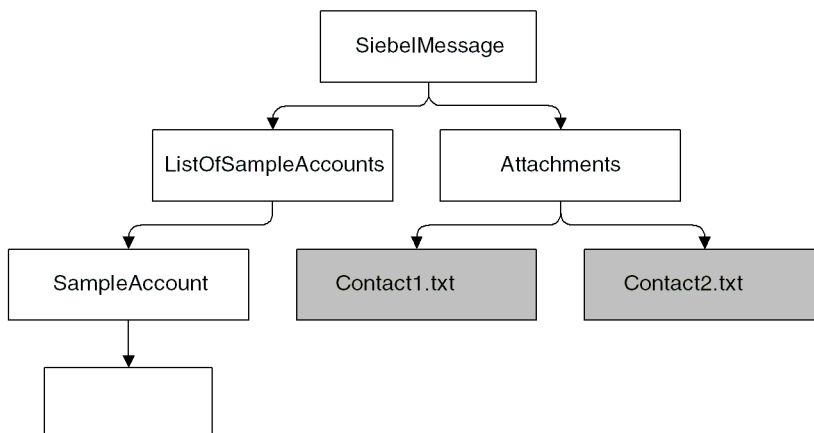


Figure 27. Sample Account with Attachments as Input to the MIME Hierarchy Converter

The output of this process is illustrated in [Figure 28](#).

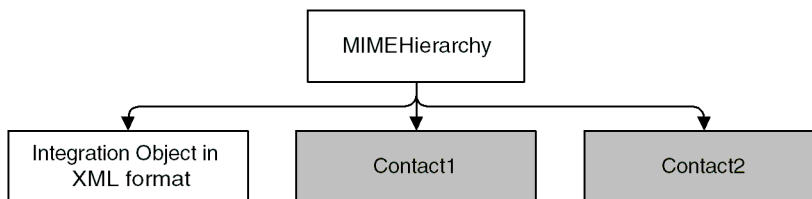


Figure 28. Output of a MIME Hierarchy Converter

The first child of a MIME Hierarchy is the XML format of the Sample Account Integration Object instance found in the Siebel Message. The remaining two children are the corresponding children found under Attachments. In the event that there is no child of type Attachments in the Siebel Message, the output will be just a MIME Hierarchy with a child of type Document. This document will contain the XML format of the Sample Account Integration Object instance.

Inbound Integration

The MIME Hierarchy Converter transforms a MIME Hierarchy input into a Siebel Message. For the inbound process, the first child of the MIME Hierarchy has to be the XML format of the Integration Object instance—otherwise an error is generated. [Figure 29](#) illustrates the incoming hierarchy.

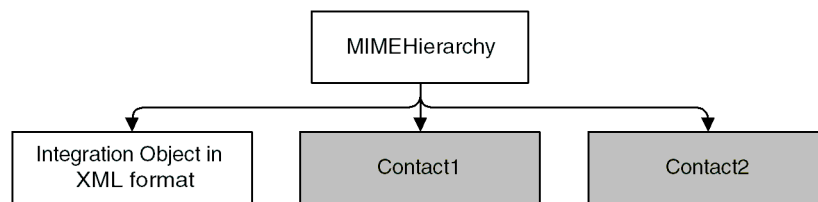


Figure 29. Output of a MIME Hierarchy Converter

And the output of this process is illustrated in [Figure 30](#).

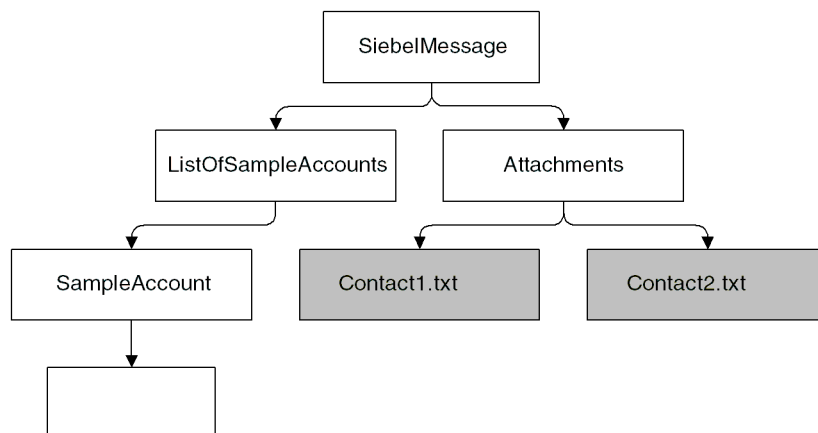


Figure 30. Output of MIME Hierarchy Message

The EAI MIME Doc Converter

The MIME Doc Converter converts a MIME Hierarchy into a MIME Message and a MIME Message into a MIME Hierarchy. A MIME Hierarchy consists of two different types of property sets.

Property	Description
MIME Hierarchy	Mapping to a MIME multi-part
Document	Mapping to MIME basic-part

Table 9 illustrates some examples of how a MIME Message maps to a MIME Hierarchy.

Table 9. Examples of MIME Message and MIME Hierarchy

MIME Message	MIME Hierarchy
MIME-Version: 1.0 Content-Type: application/xml Content-Transfer-Encoding: 7bit This is a test.	<div>Type: Document Value: This is a test</div>
MIME-Version: 1.0 Content-Type: multipart/related; type = "application/xml"; boundary = --abc ----abc Content-Type: application/xml Content-Transfer-Encoding: 7bit This is test2. ----abc--	<div>Type: MIMEHierarchy ↓ Type: Document Value: This is test2</div>

EAI MIME Doc Converter Properties

The business service needs the following properties on the child property set as shown in [Table 10](#). These properties reflect the most accurate information on the data contained in the child property set.

Table 10. Properties for EAI MIME Doc Converter

Property	Default Value	Possible Values	Type	Description
ContentId	N/A	Any value	Document	The ContentId is the value used to identify the file attachment when the receiver parses the MIME message. When importing attachments, you should use a unique value for this property and not repeat it for the rest of the file attachments. This is required in the actual document as well as in the SiebelMessage. This property is automatically populated when you are exporting an attachment from Siebel application.
Extension	N/A	txt, java, c, C, cc, CC, h, hxx, bat, rc, ini, cmd, awk, html, sh, ksh, pl, DIC, EXC, LOG, SCP, WT, mk, htm, xml, pdf, AIF, AIFC, AIFF, AU, SND, WAV. gif, jpg, jpeg, tif, XBM, avi, mpeg, ps, EPS, tar, zip, js, doc, nsc, ARC, ARJ, B64, BHX, GZ, HQX	Document	If ContentType and ContentSubType are not defined, the Extension will be used to retrieve the appropriate values from this property. If all three values are specified, the ContentType and ContentSubType values will override the values retrieved from the Extension. If either the Extension or both ContentType and ContentSubType are not specified, the ContentType will be set to application and ContentSubType will have the value of octet-stream.
ContentType	application	application, audio, image, text, video	Document	The ContentType value has to be specified if you want to set the content type of the document instead of using the extension to get a value from the MIME utility function. If the value is not provided, the default value will be used. The ContentType of multipart is used to represent file attachments in a MIME message. Other forms of values to describe a multipart will not be supported.

Table 10. Properties for EAI MIME Doc Converter

Property	Default Value	Possible Values	Type	Description
ContentSubType	octet-stream	plain, richtext, html, xml (used with ContentType of Text) octet-stream, pdf, postscript, x-tar, zip, x-javascript, msword, x-conference, x-gzip (used with ContentType of application) aiff, basic, wav (used with ContentType of audio) gif, jpeg, tiff, x-xbitmap (used with ContentType of image) avi, mpeg (used with ContentType of video)	Document	The ContentSubType value has to be specified if you want to set the content subtype of the document instead of using the extension to get a value from the MIME utility function. If the value is not provided the default value will be used.

NOTE: On the inbound direction, the business service is independent of the transport. It assumes that the input property set contains the MIME message and outputs a property set representation of the MIME message. A property set is used to represent each part of the MIME message. When decoding the MIME message, the business service automatically sets the properties based on the values in the MIME message.

Siebel Virtual Business Components

7

This chapter describes the virtual business component (VBC), and its uses and restrictions. This chapter also describes how you can create a new VBC in Siebel Tools.

Overview of Virtual Business Components

A virtual business component (VBC) provides a way to access data that resides in an external data source using a Siebel business component. The VBC does not map to an underlying table in the Siebel database. You create a new VBC in Siebel Tools and compile it into the siebel.srf file. The VBC calls a Siebel business service to provide a transport mechanism.

You can take two approaches to use virtual business components, as illustrated in [Figure 31](#).

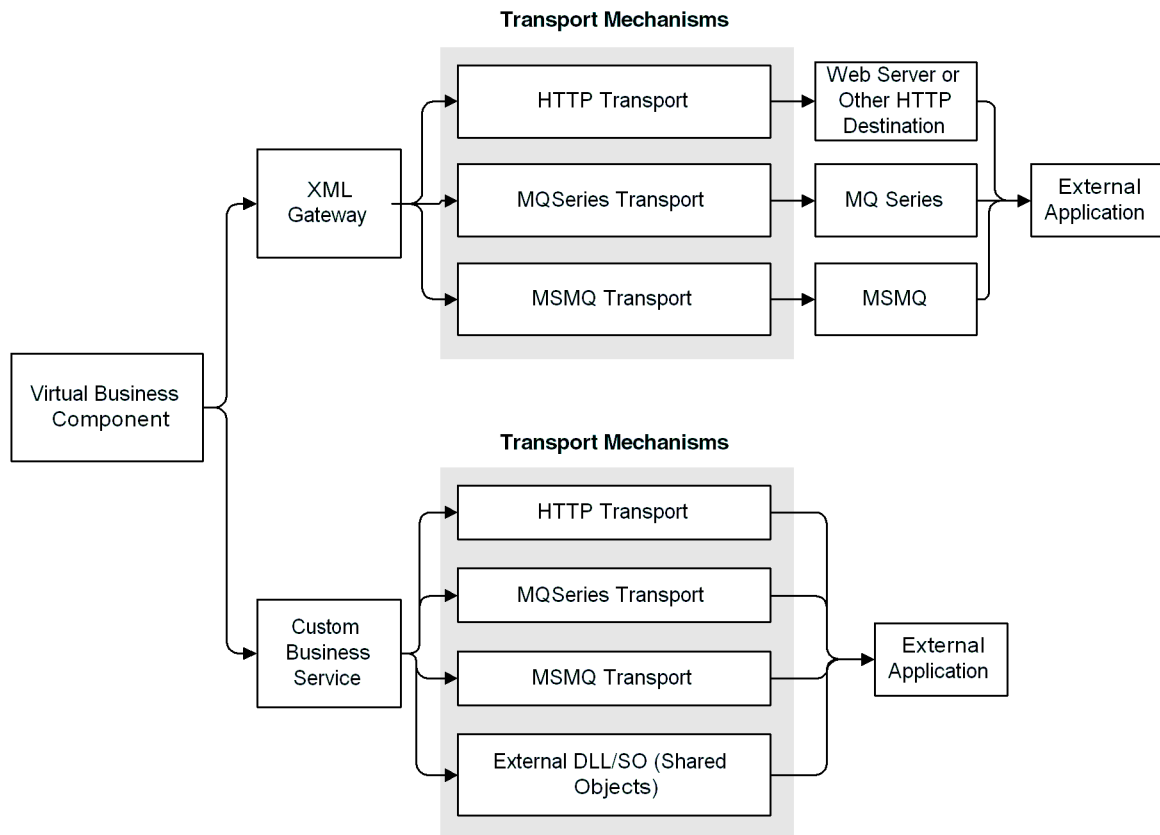


Figure 31. Two Approaches to Building Virtual Business Components

- Use the XML Gateway business service to pass data between the virtual business component and one of the Siebel transports, such as the EAI HTTP Transport, the EAI MQSeries AMI Transport, or the EAI MSMQ Transport.
- Write your own business service in Siebel eScript or in Siebel VB to implement the methods described in this chapter.

Enhancements to VBCs for This Version

The following new features and enhancements have been implemented in this version to enhance the functionality of the VBCs to better assist you in achieving your business requirements:

- Virtual business components (VBCs) support drill down from a VBC.
- A parent applet can be based on a VBC.
- You can define virtual business components that can participate as a parent in a business object.
- You still can use an older version of XML format or property set by setting the VBC Compatibility Mode parameter to the appropriate version. For details, see [Table 11 on page 138](#).
- You can pass search and sort specifications to the business service used by a VBC.
- You can use Validation, Pre Default Value, Post Default Value, Link Specification, and No Copy attributes of VBC fields.
- You can use Predefined queries and Picklists with VBC.
- You can use Cascade Delete, Search Spec, Sort Spec, No Insert, No Update, and No Delete Link properties.
- You can use No Insert, No Update, No Delete, Search Spec, Sort Spec, and maximum cursor size Business Component properties.

Usage and Restrictions

- You can define a business object as containing both standard business components and virtual business components.
- Use CSSFrame (Form) and CSSFrameList (List) instead of specialized applet classes.
- Using the same name for the VBC field names and the remote data source field names may reduce the amount of required programming. (Optional)
- Virtual business components cannot be docked, so they do not apply to remote users.
- Virtual business components cannot contain a multi-value group (MVG).
- Virtual business components do not support many-to-many relationships.
- Virtual business components cannot be loaded using Enterprise Integration Manager.
- Standards business component can not contain multi-value group based on Virtual business components.
- The Specialized business components class—for example, Quotes and Forecasts—cannot be implemented as virtual business components.
- You cannot use Workflow Monitor to monitor virtual business components.

Virtual Business Components

To use VBCs to share data with an external applications you need to perform the following high-level tasks:

Checklist

-
- ☐ Create a new Virtual Business Component.
For details, see [“Creating a New Virtual Business Component.”](#)
 - ☐ Set the User Properties on Virtual Business Components (VBCs).
For details, see [“Setting User Properties for the Virtual Business Component” on page 138.](#)
 - ☐ Configure your VBC Business Service:
 - Configure your XML Gateway Service or write your own Business Service.
For details, see [“XML Gateway Service” on page 140](#) and [“Custom Business Service Methods” on page 150.](#)
 - Configure your external application.
For details, see [“External Application Setup” on page 149.](#)
-

Creating a New Virtual Business Component

You create a new virtual business component in Siebel Tools.

To create a new virtual business component

- 1** Lock the appropriate project.
- 2** Create a new record in the Business Component list applet in Siebel Tools.
- 3** Name the business component.
- 4** Select the project you locked in [Step 1](#).
- 5** Set the Class to the *CSSBCVExtern* class. This class provides the virtual business component functionality.

Setting User Properties for the Virtual Business Component

When defining the virtual business component, you must provide the user properties shown in [Table 11](#).

Table 11. Setting Virtual Business Component User Properties

User Property	Description
Service Name	The name of the business service.
Service Parameters	Any parameters required by the business service (Optional). The Siebel application passes this user property, as an input argument, to the business service.
Remote Source	External data source that the business service is to use (Optional). This property allows the VBC to pass a root property argument to the underlying Business Service, but it does not allow a connection directly to the external datasource. The Siebel application only passes this user property as an input argument.
VBC Compatibility Mode	<p>Determining the format the property set passed from a VBC to a business service or the outgoing XML from the XML Gateway will be in. A valid value is Siebel xxx, where xxx can be any Siebel release number. Some examples would be “Siebel 6” or “Siebel 7.0.4”. If xxx is less than 7.5, the format will be in pre 7.5. Otherwise, a new property set and XML format will be passed.</p> <p>If you are creating a VBC in 7.5, there is no need to define this new user property since the default would be to use the new PropertySet from VBC and the new outgoing XML from the XML Gateway.</p> <p>For your existing VBC implementation you need to update your VBC definition by adding this new user property and setting it to Siebel xxx, where xxx is your desired version number.</p>

To define user properties

- 1 Lock the appropriate project.
- 2 Click the Business Component folder in the Object Explorer to expand the hierarchical tree.
- 3 Select the business component you want to define user properties for.

- 4** Click the Business Component User Prop folder in the Object Explorer.
The Business Component User Properties list applet appears.
- 5** Choose Edit New Record to create a new blank user property record.
- 6** Type the name of the user property, such as Service Name, in the Name field.
- 7** Type the value of the user property, such as a business service name, in the Value field.
- 8** Repeat the process for every user property you want to define for this virtual business component.

NOTE: See [“XML Gateway Service” on page 140](#) for list of different property sets and their format.

XML Gateway Service

The XML Gateway business service communicates between Siebel applications and external data sources using XML as the data format. For details on XML format, see [“Examples of Outgoing XML Format” on page 143](#) and [“Examples of Incoming XML Format” on page 147](#). The XML Gateway business service can be configured to use one of the following transports:

- EAI MQSeries AMI Server Transport
- EAI MQSeries Server Transport
- EAI HTTP Transport
- EAI MSMQ Transport

You can configure the XML Gateway by specifying the transport protocol and the transport parameters you use in the Service Parameters User Property of the virtual business component as shown in [Table 12](#). When using the XML Gateway, you need to specify the following user properties for your virtual business component.

Table 12. Transport Protocol

Name	Value
Service Name	XML Gateway
Service Parameters	< variable1 name > = < variable1 value > ; < variable2 name > = < variable2 value > ;... >
Remote Source	< External Data Source >
VBC Compatibility Mode	Siebel xxx, where xxx can be any Siebel release number.

NOTE: You can concatenate multiple name-value pairs using a semicolon (;).

For example, if you want to specify the EAI HTTP Transport, you may use something like the following which is also illustrated in [Figure 32](#):

```
"Transport=EAI HTTP Transport;HTTPRequestURLTemplate=<your URL>;HTTPRequestMethod=POST"
```

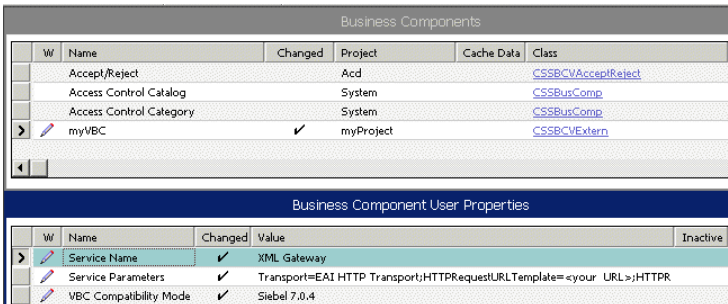


Figure 32. Setting Virtual Business Component User Properties

or if you want to specify the EAI MQSeries AMI Transport, you may use something like:

```
"Transport=EAI MQSeries AMI Transport;MqPolicyName=<policy name>;MqSenderServiceName=<sender service name>;MqModelQueueName=<queue name>;MqPhysicalQueueName=<p queue name>;..."
```

XML Gateway Methods

The XML Gateway provides the methods described in [Table 13](#).

Table 13. XML Gateway Methods

Method	Description
Init	Initializes the XML Gateway business service for every business component.
Delete	Deletes a given record in the remote data source.
Insert	Inserts a record into a remote data source.
PreInsert	Performs an operation that tests for the existence of the given business component. Only default values are returned from the external application.
Query	Queries the given business component from the given data source.
Update	Updates a record in the remote data source.

XML Gateway Method Arguments

The XML Gateway init, delete, insert, preInsert, query, and update methods take the arguments described in [Table 14](#).

Table 14. XML Gateway Arguments for the Init Method

Method	Description
Business Component Id	Unique key for the given business component.
Business Component Name	Name of the business component or its equivalent, such as a table name.
Parameters	A set of string parameters required for initializing the XML Gateway.
Remote Source	<p>The remote source from which the service is to retrieve data for the business component. This must be a valid connect string.</p> <p>When configuring the repository business component on top of the specialized business component class CSSBCVExten, a user property RemoteSource can be defined to allow the Transport Services to determine the remote destination and any connect information. If this user property is defined, it will be passed to every request as the <remote-source> tag.</p>

Examples of Outgoing XML Format

Examples of the XML documents generated and sent by the XML Gateway to the external system are described in [Table 15](#). See [Appendix C, “DTDs for XML Gateway Business Service,”](#) for examples of the DTDs that correspond to each of these methods.

Table 15. Outgoing XML Tags and Descriptions

Method	Format of the XML Stream	Description
Init Request	<pre>< siebel-xmlxt-fields-req > < buscomp id = "1" > People < /buscomp > < remote-source > MQ:comp:cust < /remote-source > < /siebel-xmlxt-fields-req ></pre>	<p>siebel-xmlxt-fields-req. This tag fetches the list of fields supported by this instance.</p> <p>buscomp Id. The business component id.</p> <p>remote-source. The remote source from which the service is to retrieve data for the business component.</p>

Table 15. Outgoing XML Tags and Descriptions

Method	Format of the XML Stream	Description
Query Request	<pre> < siebel-xmltext-query-req > < buscomp id = "1" > Transaction VBC < / buscomp > < remote-source > http://trothwein2:9080/ examples/servlet/jdbcReqTran < /remote-source > < max-rows > 6 < /max-rows > < search-string > ([CustomerID] = "12-E4W8I" AND [AccountNumber] = "0000-0000") < /search-string > < search-spec > < node node-type = "Binary Operator" > AND < node node-type = "Binary Operator" > = < node node-type = "Identifier" > CustomerID < / node > < node value-type = "TEXT" node- type = "Constant" > 12-E4W8I < /node > < /node > < node node-type = "Binary Operator" > = < node node- type = "Identifier" > AccountNumber < /node > < node value-type = "TEXT" node- type = "Constant" > 0000-0000 < /node > < /node > < /node > < /search-spec > < /siebel-xmltext-query-req > </pre>	<p>siebel-xmltext-query-req. This tag queries by example. The query-req XML stream contains parameters necessary to set up the query. In this example, the query requests that record information be returned from the remote system.</p> <p>max-rows. Maximum number of rows to be returned. The value is defaulted to the Maximum Cursor Size defined at the VBC plus one. If the Maximum Cursor Size property is not defined at the VBC, then the max-rows property will not be passed.</p> <p>search-string. The search specification used to query and filter the information.</p> <p>search-spec. Hierarchical representation of the search-string.</p>
PreInsert Request	<pre> < siebel-xmltext-preinsert-req > < buscomp id = "1" > People < /buscomp > < remote-source > MQ:comp:cust < /remote-source > < /siebel-xmltext-preinsert-req > </pre>	<p>siebel-xmltext-preinsert-req. This tag allows the connector to provide default values. This operation is called when a new row is created, but before any values are entered through the BusComp interface.</p>

Table 15. Outgoing XML Tags and Descriptions

Method	Format of the XML Stream	Description
Update Request	<pre> < siebel-xmlxt-update-req > < buscomp > People < /buscomp > < remote-source > MQ:comp:cust < /remote-source > < row > < value field = "Name" changed = "false" > John Q. Smith < /value > < value field = "Age" changed = "true" > 36 < /value > < value field = "Source" changed = "false" > Siebel XML < /value > < value field = "Credit Limit" changed = false > 0 < /value > < /row > < /siebel-xmlxt-update-req > </pre>	<p>siebel-xmlxt-Update-req. This tag requests changes to the field values for an existing row.</p> <p>All values for the record are passed in with < value > tags with the changed attribute identifying the ones which have been changed through the Siebel application.</p>
Insert Request	<pre> < siebel-xmlxt-insert-req > < buscomp > People < /buscomp > < remote-source > MQ:comp:cust < /remote-source > < row > < value field = "Name" > John Q. Smith < /value > < value field = "Age" > John Q. Smith < /value > < value field = "Source" > Siebel XML < /value > < value field = "Credit Limit" > 0 < /value > < /row > < /siebel-xmlxt-insert-req > </pre>	<p>siebel-xmlxt-Insert-req. This tag requests the commit of a new record in the remote system.</p> <p>The insert-req XML stream contains values for fields entered through the business component.</p>

Table 15. Outgoing XML Tags and Descriptions

Method	Format of the XML Stream	Description
Delete Request	< siebel-xmltext-delete-req > < buscomp id = "1" > People < /buscomp > < remote-source > MQ:comp:cust < /remote-source > < row > < value field = "Name" > John Q. Smith < /value > < value field = "Age" > 36 < /value > < value field = "Source" > Siebel XML < /value > < value field = "Credit Limit" > 0 < /value > < /row > < /siebel-xmltext-delete-req >	siebel-xmltext-delete-req. This tag requests removal of a single record in the remote system.

Examples of Incoming XML Format

Table 16 contains examples of XML documents that are sent from an external system to the XML Gateway in response to a request. See Appendix C, “DTDs for XML Gateway Business Service,” for examples of the DTDs that correspond to each of these methods.

Table 16. Incoming XML Tags and Descriptions

Method	Format of the XML Stream	Description
Error	<pre>< siebel-xmltext-status > < status-code > 4 < /code > < error-field > Name < /error-field > < error-text > Name must not be empty < /error-text > < /siebel-xmltext-status ></pre>	<p>Format of the XML stream expected by the Siebel application in case of an error in the external application. The tags for this XML stream, including the entire XML stream, are optional. If the error is specific to a field, the field name should be specified.</p> <p>siebel-xmltext-status. This tag is used to check the status returned by the external system.</p> <p>status-code. This tag overrides the return value.</p> <p>error-text. This tag specifies textual representation of the error, if it is available. This tag appears in addition to the standard error message. For example, if the Siebel application attempts to update a record in the external system with a NULL Name, and this is not allowed in the external system, then the error text will be set to “Name must not be empty.”</p>

Table 16. Incoming XML Tags and Descriptions

Method	Format of the XML Stream	Description
Init Return	<pre>< siebel-xmllex-fields-ret > < support field = "Name" / > < support field = "Age" / > < support field = "Source" / > < support field = "Credit Limit" / > < /siebel-xmllex-fields-ret ></pre>	<p>siebel-xmllex-fields-ret. The FieldsRet XML stream returned contains the list of fields supported by the external system for this instance.</p> <p>The following field names are reserved by the Siebel application and should not appear in this list:</p> <ul style="list-style-type: none"> Id Created Created By Updated Updated By
Query Return	<pre>< siebel-xmllex-query-ret > < row > < value field = "Name" > John Q. Smith < /value > < value field = "Age" > 35 < /value > < value field = "Source" > DMM < /value > < value field = "Credit Limit" > 0 < /value > < /row > < /siebel-xmllex-query-ret ></pre>	<p>siebel-xmllex-query-ret. The query-ret XML stream contains the result set which matches the criteria of the query.</p> <p>row. This tag indicates number of rows returned by query. Each row should contain one or more < values > . The attributes which appear in < row > tags must be able to uniquely identify rows. If there is a unique key in the remote data source, it should appear in the result set. If not, a unique key should be generated. It is necessary to identify specific rows for DML operations.</p> <p>value. This tag specifies the field and value pairs and should be the same for each row in the set.</p>
PreInsert Return	<pre>< siebel-xmllex-preinsert-ret > < row > < value field = "Source" > Siebel XML < /value > < /row > < /siebel-xmllex-preinsert-ret ></pre>	<p>siebel-xmllex-preinsert-ret. Returns default values for each field if there is any default value.</p>

Table 16. Incoming XML Tags and Descriptions

Method	Format of the XML Stream	Description
Insert Return	<pre>< siebel-xmltext-insert-ret > < row > < value field = "Credit Limit" > 15000.00 < /value > < /row > < /siebel-xmltext-insert-ret ></pre>	siebel-xmltext-insert-ret. If the remote system has inserted records, they can be returned to be reflected in the business component in an insert-ret XML stream in the < row > tag format as the insert-ret stream.
Update Return	<pre>< siebel-xmltext-update-ret > < row > < value field = "Credit Limit" > 15000.00 < /value > < /row > < /siebel-xmltext-update-ret ></pre>	siebel-xmltext-update-ret. If the remote system updated fields, they can be returned to be reflected in the business component in an update-ret XML stream in the < row > tag format as the update-ret stream.
Delete Return	< siebel-xmltext-delete-ret / >	siebel-xmltext-delete-ret. Only the XML stream tag is returned.

External Application Setup

Once you have your XML Gateway Service configured, you need to set up your external application accordingly to be able to receive and respond to the requests.

NOTE: At a minimum, the external application needs to support the Init() and Query() methods, and depending upon the functionality provided by the VBC, the remaining methods may or may not be necessary.

Custom Business Service Methods

Your business service must implement the Init and Query methods as described in this section. The Delete, PreInsert, Insert, and Update methods are optional, and dependent upon the functionality required by the Virtual Business Component.

NOTE: Custom business services can be only based on the CSSService class, as specified in Siebel Tools.

These methods pass property sets between the virtual business component and the business service. Virtual business component methods take property sets as arguments. Each method takes two property sets: an Inputs property set and an Outputs property set. The methods are called by the *CSSBCVExtern* class in response to requests from other objects that refer to or are based on the virtual business component.

When you are building a custom business service to allow virtual business component functionality with Siebel VB or Siebel eScript you can use one of the following methods to connect to an external database in the Service code:

- **Siebel VB Only.** Use the SQL functions using ODBC.
- **Siebel eScript Only.** Call out to a CORBA interface using the CORBACreateObject function.
- **Siebel VB or eScript.** Use a COM connection via the CreateObject or COMCreateObject functions to call an API supported by your RDBMS vendor or to call a COM object such as ActiveX DLL.

NOTE: You may also choose to use the XML Gateway service to allow the connection for your VBC. For details, see [“XML Gateway Service” on page 140](#).

For more information about property sets, programming in Siebel eScript, and programming in Siebel VB, see *Siebel Tools Reference, MidMarket Edition* and *Siebel Tools Online Help, MidMarket Edition*.

Common Method Parameters

[Table 17](#) shows the input parameters common to every method.

Table 17. Common Input Parameters

Parameter	Description
Remote Source	Specifies the name of an external data source. Argument in the root property set.
Business Component Name	Name of the active virtual business component. Argument in the root property set.
Business Component Id	Internally generated unique value that represents the virtual business component. Argument in the root property set.

Once a response has been received, a method on the service is called. The method packages the response from the external data source into the Outputs property set. [Table 18](#) shows the output parameters common to every method.

Table 18. Common Output Parameters

Parameter	Description
Status Code	A numeric value representing the result of the operation. Any non-zero value signifies an error.
Error Text	Optional text that can be provided for a given error.

Business Services Methods and Their Property Sets

The following examples display each method, property set, and additional input and output parameters using a virtual business component *Transactions* that displays credit card transactions for a given Contact.

Delete The Delete method is called when a record is deleted. [Figure 33](#) illustrates the property set for the Delete input and is followed by its XML representation.

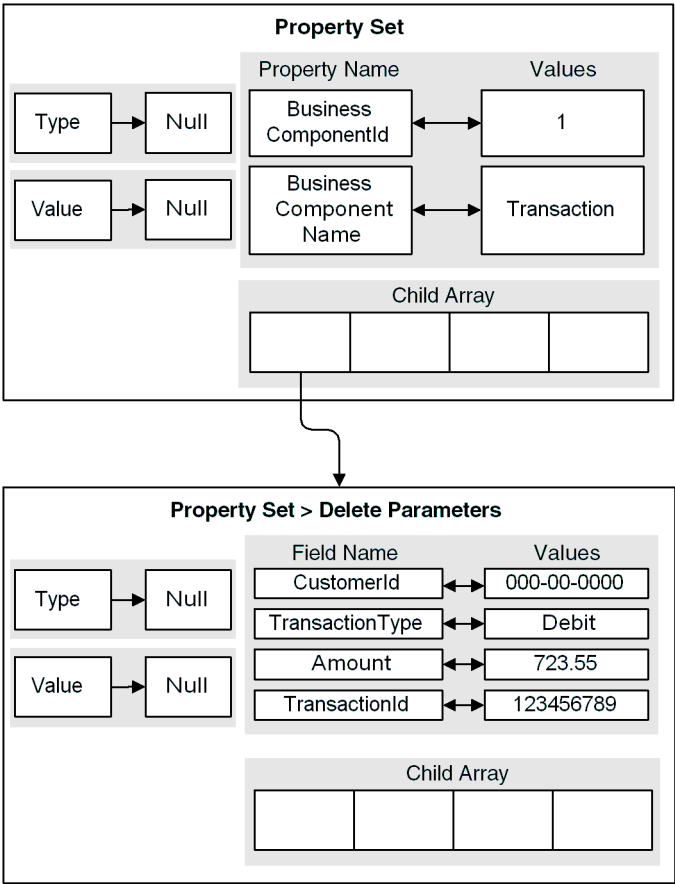


Figure 33. Delete Input Property Set


```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="false"?>
<PropertySet
  BusinessComponentId="1"
  BusinessComponentName="Transaction">
  <PropertySet
    CustomerId="000-00-0000"
    TransactionType="Debit"
    Amount="723.55"
    TransactionId="123456789"
  </PropertySet>
</PropertySet>
```

Figure 34 illustrates the property set for the Delete output and is followed by its XML representation.

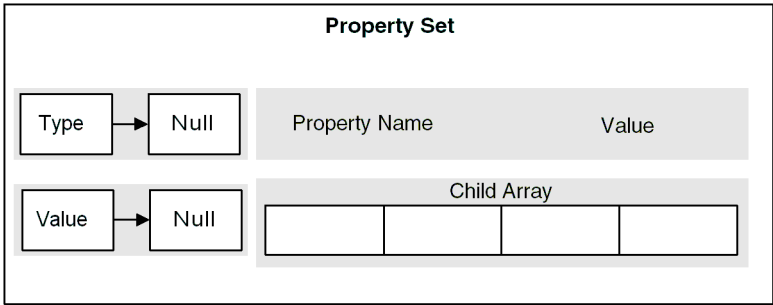


Figure 34. Delete Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="false"?>
<PropertySet
</PropertySet>
```

Init The Init method is called when the virtual business component is first instantiated. It initializes the virtual business component. It expects to receive the list of fields supported by the external system. [Figure 35](#) illustrates the property set for Init input and is followed by its XML representation.

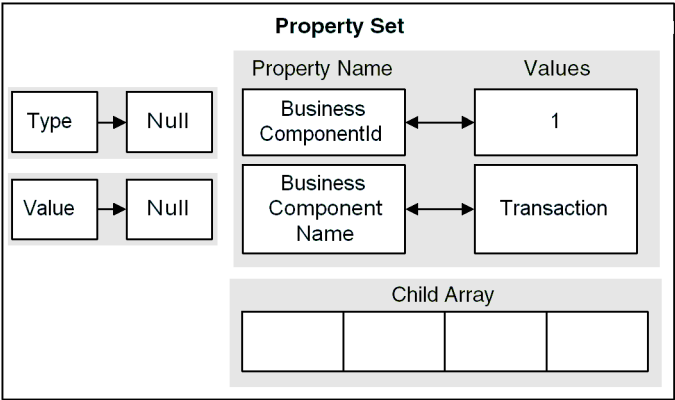


Figure 35. Init Input Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet
  BusinessComponentId="1"

  BusinessComponentName="Transaction">
</PropertySet>
```

Figure 36 illustrates the property set for Init output and is followed by its XML representation.

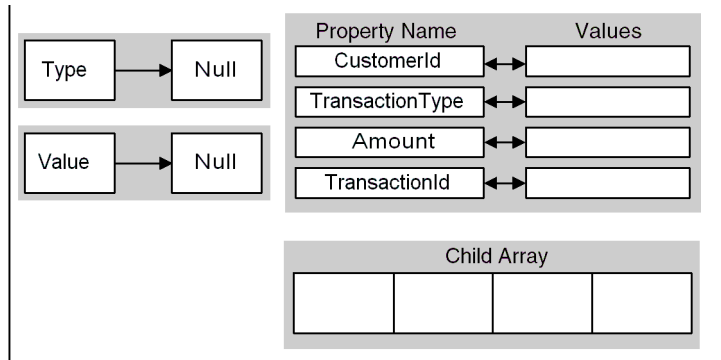


Figure 36. Init Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet
  CustomerId=" "
  TransactionType=" "
  Amount=" "
  TransactionId=" ">
</PropertySet>
```

Insert The Insert method is called when a New Record is committed. [Figure 37](#) illustrates the property set for Insert input and is followed by its XML representation.

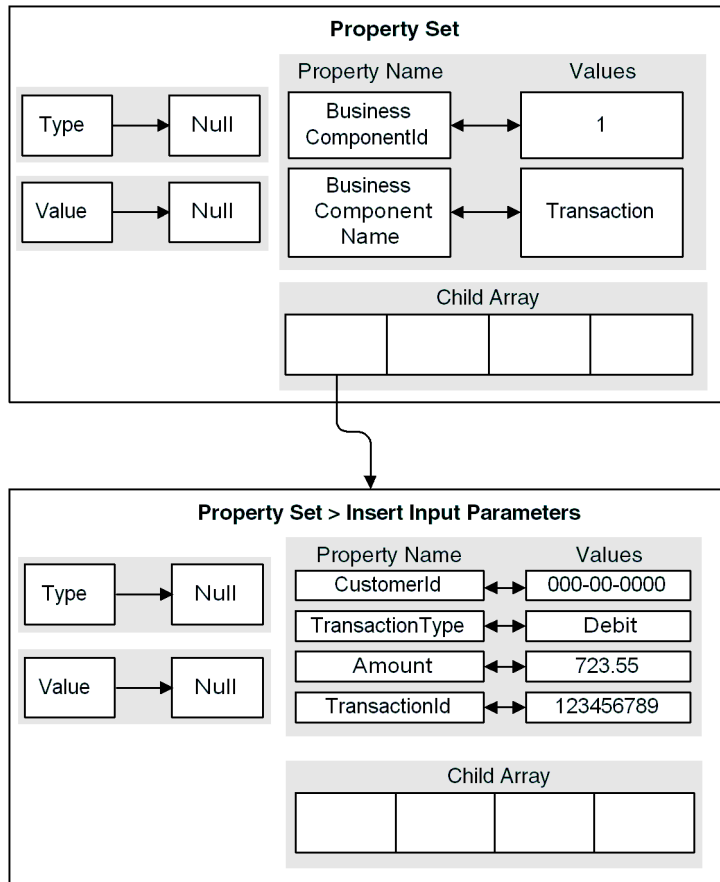


Figure 37. Insert Input Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>

<PropertySet
  BusinessComponentId="1"
  BusinessComponentName="Transaction"
```

```

<PropertySet
  CustomerId="000-00-0000"
  TransactionType="Debit"
  Amount="723.55"
  TransactionId="123456789">
  </PropertySet>
</PropertySet>

```

Figure 38 illustrates the property set for Insert output and is followed by its XML representation.

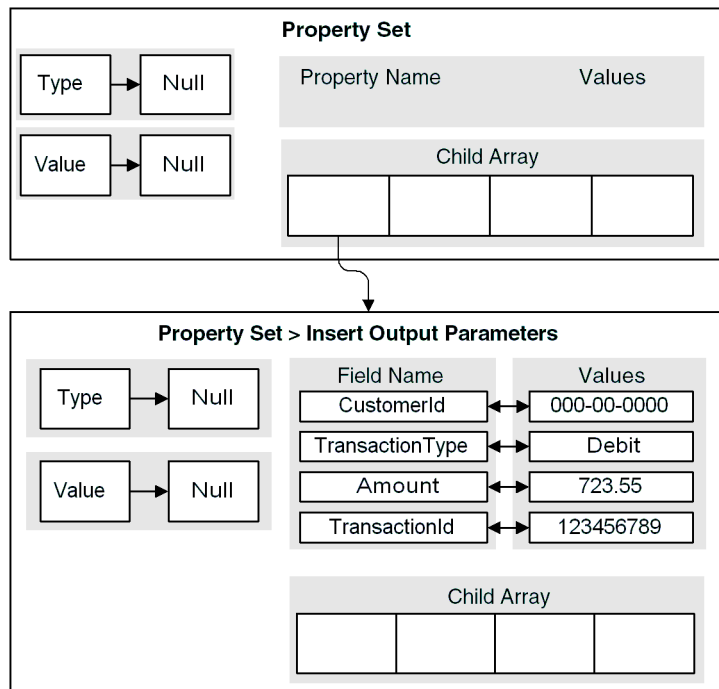


Figure 38. Insert Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet
  <PropertySet
    CustomerId="000-00-0000"
    TransactionType="Debit"
    Amount="723.55"
    TransactionId="123456789">
  </PropertySet>
</PropertySet>
```

PreInsert The PreInsert method is called when a New Record operation is performed. It supplies default values. [Figure 39](#) illustrates the property set for PreInsert input and is followed by its XML representation.

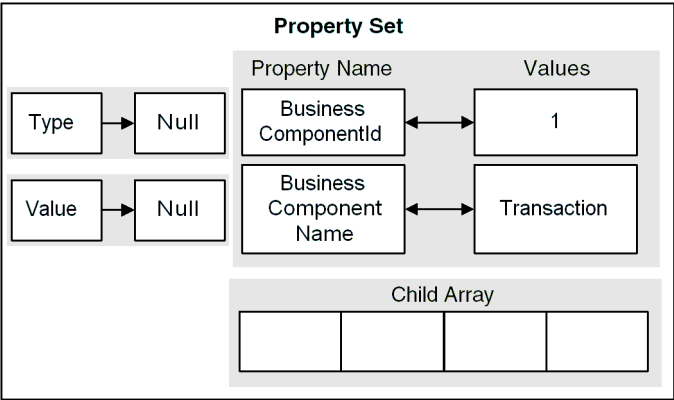


Figure 39. PreInsert Input Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet
  BusinessComponentId="1"
  BusinessComponentName="Transaction">
</PropertySet>
```

Figure 40 illustrates the property set for PreInsert output and is followed by its XML representation.

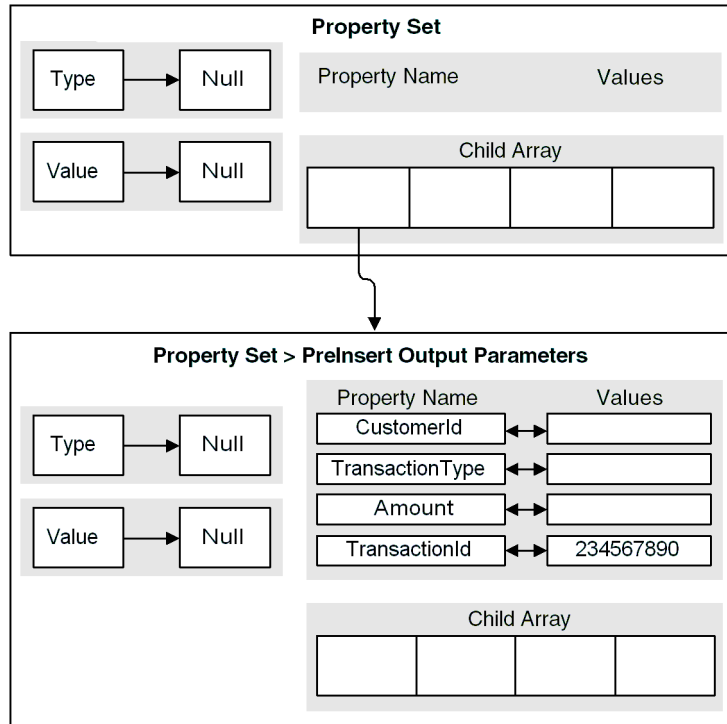


Figure 40. Preinsert Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet>
  <PropertySet
    CustomerId=""
    TransactionType=""
    Amount=""
    TransactionId="234567890">
  </PropertySet>
</PropertySet>
```

Query The Query method is called when a search is performed. The Query method must be supported by every virtual business component. A single child property set containing a list of field name–field value pairs representing the query is shown here. Each record that matches the query will be represented as a property set. For example, if 5 records match the query, there will be 5 child property sets. Each property set will contain a list of field name—field value pairs representing the values of each field for that particular record. [Figure 41](#) illustrates the property set for Query input and is followed by its XML representation.

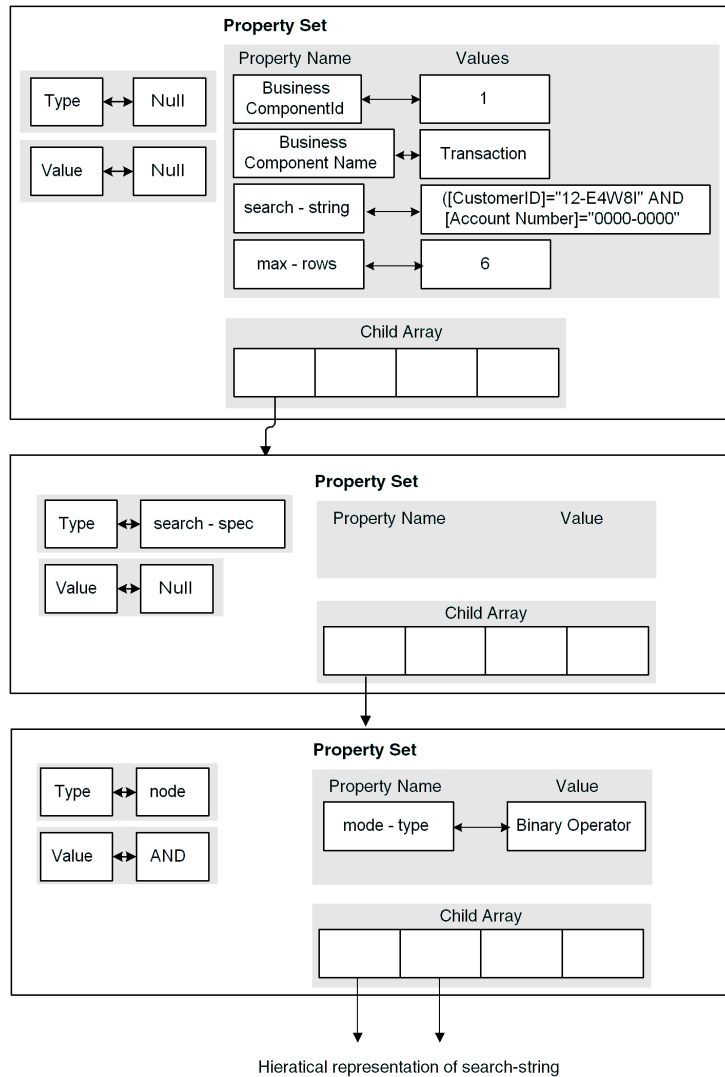


Figure 41. Query Input Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet

    search-string="([CustomerID] = "12-E4W8I" AND [AccountNumber] =
"0000-0000")"

    BusinessComponentId="1"

    BusinessComponentName="Transaction VBC">

<max-rows> 6 </max-rows>

<search-spec>

    <node node-type="Binary Operator">AND

        <node node-type="Binary Operator">=

            <node node-type="Identifier">CustomerID</node>

            <node value-type="TEXT" node-type="Constant">12-
E4W8I</node>

        </node>

        <node node-type="Binary Operator">=

            <node node-type="Identifier">AccountNumber</node>

            <node value-type="TEXT" node-type="Constant">0000-
0000</node>

        </node>

    </node>

</search-spec>

</PropertySet>
```

Figure 42 illustrates the property set for Query output and is followed by its XML representation.

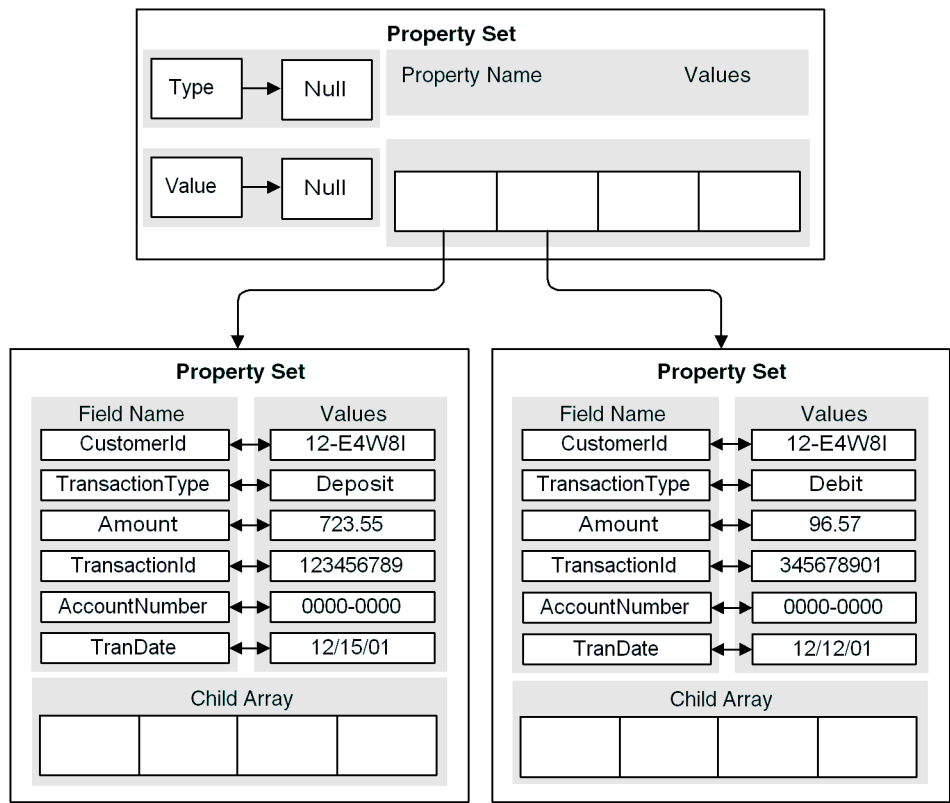


Figure 42. Query Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet>
  <PropertySet
    Amount="723.55"
```

```
        CustomerId="12-E4W8I"  
        AccountNumber="0000-0000"  
        TransactionId="123456789"  
        TransactionType="Deposit"  
        TranDate="12/15/2001">  
    </PropertySet>  
    <PropertySet  
        Amount="96.57"  
        CustomerId="12-E4W8I"  
        AccountNumber="0000-0000"  
        TransactionId="345678901"  
        TransactionType="Debit"  
        TranDate="12/12/2001">  
    </PropertySet>  
</PropertySet>
```

Update The Update method is called when a record is modified. [Figure 43](#) illustrates the property set for Update input and is followed by its XML representation.

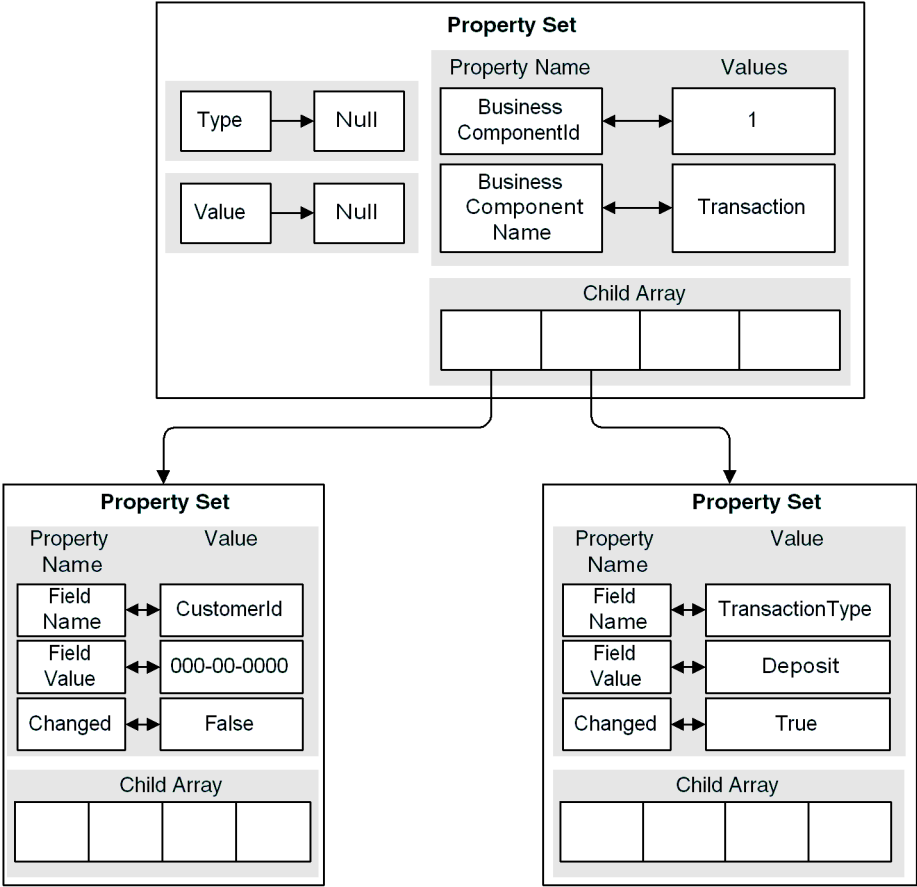


Figure 43. Update Input Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="true"?>
<PropertySet
  BusinessComponentId="1"
  BusinessComponentName="Transaction">
  <PropertySet
```

```
Field Name="CustomerId"
Field Value="000-00-0000"
Changed="false">
</PropertySet>
<PropertySet
Field Value="Deposit"
Field Name="TransactionType"
Changed="true">
</PropertySet>
</PropertySet>
```

Figure 44 illustrates the property set for the Update output and is followed by its XML representation.

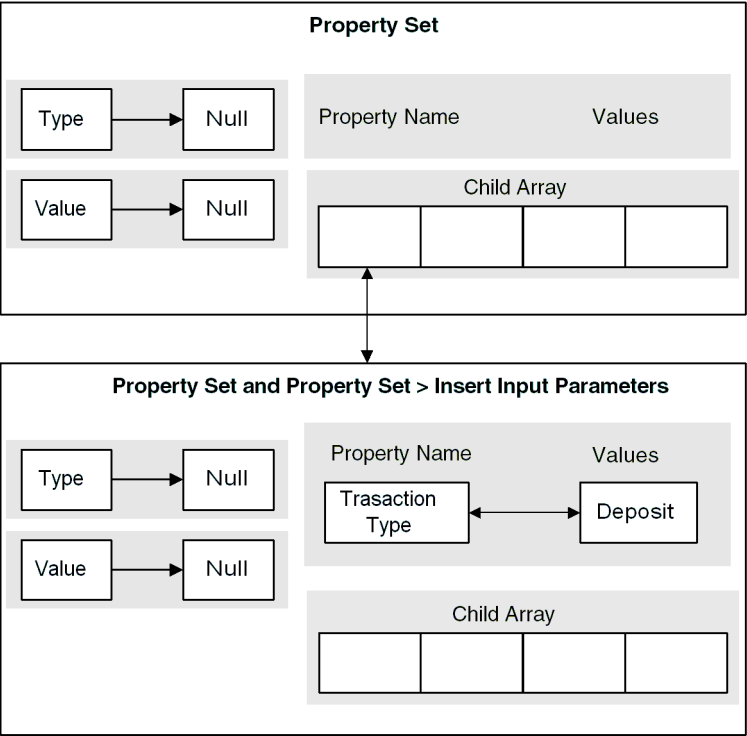


Figure 44. Update Output Property Set

```
<?xml version="1.0" encoding="UTF-8"?>
<?Siebel-Property-Set EscapeNames="false"?>
<PropertySet>
  <PropertySet
    TransactionType="Deposit">
  </PropertySet>
</PropertySet>
```

Custom Business Service Example

The following example of a partial, simplified Siebel eScript implementation of a business service can be used with the virtual business component. The fields supported by this example virtual business component are First Name, Last Name, Account Balance, and Creation Date.

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    if (MethodName == "Init")
    {
        return (Init (Outputs));
    }
    else if (MethodName == "Query")
    {
        return (Query (Inputs, Outputs));
    }
    else if (MethodName == "Update")
    {
        return (Update (Inputs, Outputs));
    }
    else if (MethodName == "PreInsert")
    {
        return (PreInsert (Inputs, Outputs));
    }
    else if (MethodName == "Insert")
    {
        return (Insert (Inputs, Outputs));
    }
    else if (MethodName == "Delete")
    {
        return (Delete (Inputs, Outputs));
    }
    else
    {
        return (CancelOperation);
    }
}

function Init (Outputs)
{
    Outputs.SetProperty ("First Name", "");
    Outputs.SetProperty ("Last Name", "");
    Outputs.SetProperty ("Account Balance", "");
    Outputs.SetProperty ("Creation Date", "");
    return (CancelOperation);
}
```



```
function Query(Inputs, Outputs)
{
    var maxRows, searchString, childCount, i;
    var child = TheApplication().NewPropertySet();
    ...

    // get max-rows
    maxRows = Inputs.GetProperty("max-rows");

    // get search-string
    searchString = Inputs.GetProperty("search-string");

    // get search-spec, sort-spec
    childCount = Inputs.GetChildCount();
    for (i = 0; i < childCount; i++)
    {
        child = Inputs.GetChild(i);

        if (child.GetType() == "search-spec")
        {
            // child contains the search-spec

            ...

        }
        else if (child.GetType() == "sort-spec")
        {
            // child contains the sort-spec

            ...

        }
    }
    ...
}
```

```
// Your script would perform additional processing
// here but this example shows hard-coded
// results returned.

var row;

...

for( var i=0; i< 2; i++ )
{
    row=TheApplication ().NewPropertySet ();
    ...

    Outputs.AddChild (row.Copy());
}

row.Reset();
```

NOTE: These examples only work if the format of the date in the database is MM/DD/YYYY HH:MM:SS.

For more examples of VBCs, see *Developing and Deploying Siebel eBusiness Applications, MidMarket Edition*.

Predefined EAI Business Services

A

Siebel eBusiness Applications provide a number of business services. These services do not require any modification, but they do require that you choose and configure them to suit your requirements.

NOTE: For general information on using business services, refer to [Chapter 3, “Business Services.”](#)

Predefined EAI Business Services

Table 19 describes the predefined Siebel eAI business services.

Table 19. Predefined EAI Business Services

Business Service	Class	Description
EAI XSD Wizard		Used to create integration objects based on XSD files.
EAI XML XSD Generator		Used to generate an XSD file from an integration object.
EAI Transaction Service	CSSBeginEndTransactionService	EAI Transaction service for working with Siebel transactions such as begin, end, or find out whether in transaction.
EAI XSLT Service	CSSXSLTService	EAI XSL Transformation Service.
Workflow Process Manager (Server Request)	CSSSrmService	Submits workflow requests to a workflow process manager server component (WfProcMgr).
EAI MSMQ Transport	CSSMsmqTransService	EAI MSMQ Transport.
EAI MQSeries Server Transport	CSSMqSrvTransService	EAI MQSeries Server Transport.
EAI MQSeries AMI Transport	CSSMqAmiTransService	EAI MQSeries AMI Transport. For details, see <i>Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition</i> .
EAI HTTP Transport	CSSHTTPTransService	EAI HTTP Outbound Transport. For details, see <i>Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition</i> .
EAI Utility Service	CSSEAIUtilService	EAI Utility Service.
EAI Siebel Adapter	CSSEAISiebelAdapterService	EAI Siebel Adapter. For details, see Chapter 5, “The EAI Siebel Adapter.”

Table 19. Predefined EAI Business Services

Business Service	Class	Description
EAI Query Spec Service	CSSEAIQuerySpecService	Used internally by EAI Siebel Adapter to convert SearchSpec method argument as string to an Integration Object Instance that EAI Siebel Adapter can use as a Query By Example object.
EAI Import Export	CSSEAIImportExportService	EAI Import Export Service (import and export integration object from or to XML).
EAI BTS COM Transport	CSSEAIbtsComService	EAI Siebel to BTS COM Transport.
EAI DLL Transport	CSSDllTransService	EAI DLL Transport. For details, see <i>Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition</i> .
EAI Data Mapping Engine	CSSDataTransformationEngine	EAI Data Transformation Engine. For details, see <i>Business Processes and Rules: Siebel eBusiness Application Integration Volume IV, MidMarket Edition</i> .
No Envelope	CSSEAINullEnvelopeService	EAI Null Envelope Service. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
Siebel Message Envelope	CSSEAISMEnvelopeService	EAI Siebel Message Envelope Service. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
EAI Dispatch Service	CSSEAIDispatchService	Dispatch Service. For details, see <i>Business Processes and Rules: Siebel eBusiness Application Integration Volume IV, MidMarket Edition</i> .
EAI Integration Object to XML Hierarchy Converter	CSSEAIIntObjHierCnvService	EAI Integration Object Hierarchy (also known as SiebelMessage) to XML hierarchy converter service. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .

Table 19. Predefined EAI Business Services

Business Service	Class	Description
EAI MIME Hierarchy Converter	CSSEAIMimePropSetService	EAI MIME Hierarchy Conversion Service. For details, see Chapter 6, “Siebel eAI and File Attachments.”
EAI MIME Doc Converter	CSSEAIMimeService	MIME Document Conversion Service. For details, see Chapter 6, “Siebel eAI and File Attachments.”
EAI XML Converter	CSSEAIXMLCnvService	Converts between XML and EAI Messages. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
EAI XML Write to File	CSSEAIXMLPrtService	Print a property set to a file as XML. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
EAI XML Read from File	CSSEAIXMLPrtService	Read an XML file and parse to a property set. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
XML Converter	CSSXMLCnvService	Converts between XML documents and arbitrary Property Sets. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .
XML Hierarchy Converter	CSSXMLCnvService	Converts between XML documents and XML Property Set or Arbitrary Property Set. For details, see <i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i> .

Property Set Representation of Integration Objects

B

Property sets are in-memory representations of integration objects. This appendix describes the relationship between the property set and the integration object. For an overview of property sets, see *Siebel Tools Reference, MidMarket Edition*.

Property Sets and Integration Objects

Many eAI business services operate on integration object instances. Since business services take property sets as inputs and outputs, it is necessary to represent integration objects as property sets. The mapping of integration objects, components, and fields to property sets is known as the Integration Object Hierarchy.

Using this representation, you can pass a set of integration object instances of a specified type to an eAI business service. You pass the integration object instances as a child property set of the business service method arguments. This property set always has a type of SiebelMessage. You can pass the SiebelMessage property set from one business service to another in a workflow without knowing the internal representation of the integration objects.

Property Set Node Types

When passing integration object instances as the input or output of a business service, you can use property sets to represent different node types, as described in [Table 20](#).

Table 20. Property Set Node Types

Name	Parent	Value of Type Attribute	Properties	Description
Service Method Arguments	N/A	Ignored	The properties of this property set contain any service specific parameters, such as <i>PrimaryRowId</i> for the EAI Siebel Adapter.	This is the top-level property set of a business service's input or output. The properties of this property set contain any service-specific parameters (for example, <i>PrimaryRowId</i> for the EAI Siebel Adapter).
SiebelMessage	Service Method Arguments	SiebelMessage	The properties of this property set contain header attributes associated with the integration object, for example, <i>IntObjectName</i> .	This property set is a wrapper around a set of integration object instances of a specified type. To pass integration objects between two business services in a workflow, this property set is copied to and from a workflow process property of type <i>Hierarchy</i> .
Object List	SiebelMessage	ListOf < ObjectType >	Not used.	This property set identifies the object type that is being represented. The root components of the object instances are children of this property set.

Table 20. Property Set Node Types

Name	Parent	Value of Type Attribute	Properties	Description
Root Component	Object List	< Root Component Name >	The property names of the property set represent the field names of the component, and the property values are the field values.	This property set represents the root component of an integration object instance.
Child Component Type	Root Component or Component	ListOf < Component Name >	Not used.	An integration component can have a number of child component types, each of which can have zero or more instances. The Integration Object Hierarchy format groups the child components of a given type under a single property set. This means that child components are actually grandchildren of their parent component's property set.
Child Components	Child Component Type	< Component Name >	The property names of the property set represent the field names of the component, and the property values are the field values.	This property set represents a component instance. It is a grand-child of the parent component's property set.

Example of a Sample Account

This example shows an Account integration object in which the object has two component types: Account and Business Address (which is a child of Account). The hierarchy of component types—from a Siebel Tools perspective—looks like that shown in [Figure 45](#).

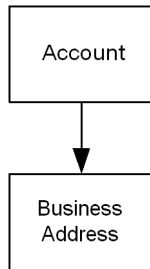


Figure 45. Sample Account Integration Object

[Figure 46 on page 181](#) shows an example instance of this object type, using the Integration Object Hierarchy representation. There are two Sample Account instances. The first object instance has an Account component and two Business Address child components. The second object instance has only an Account component with no child components.

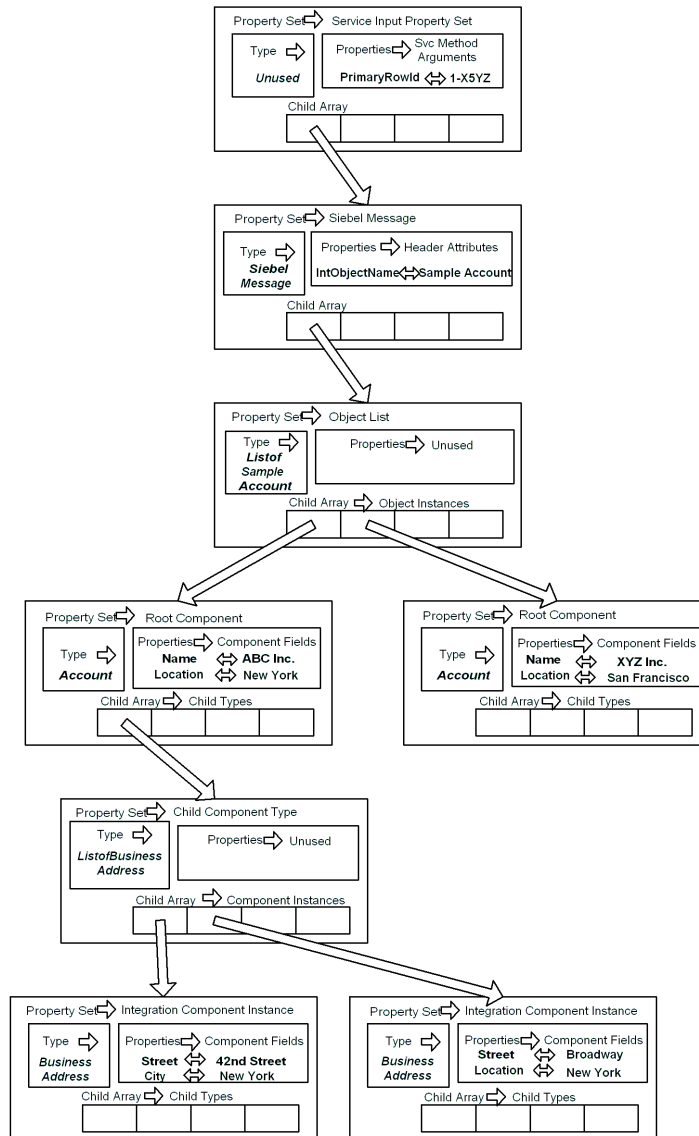


Figure 46. Partial Instance of Sample Account Integration Object

DTDs for XML Gateway Business Service

C

This appendix lists the various inbound and outbound DTDs for the XML Gateway business service.

Outbound DTDs

The following sections contain examples of DTDs representing the %methodName% request sent from the XML Gateway to the external application.

Delete

```
<!ELEMENT siebel-xmlxt-delete-req (buscomp, remote-source, row)>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED>
<!ELEMENT remote-source ( #PCDATA )*>
<!ELEMENT row (value+)>
<!ELEMENT value (#PCDATA)*>
<!ATTLIST value field CDATA #REQUIRED>
```

Init

```
<!ELEMENT siebel-xmlxt-fields-req (buscomp, remote-source?)>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED >
<!ELEMENT remote-source (#PCDATA)*>
```

Insert

```
<!ELEMENT siebel-xmlxt-insert-req (buscomp, remote-source?, row)>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED>
<!ELEMENT remote-source (#PCDATA)*>
<!ELEMENT row (value+)>
<!ELEMENT value (#PCDATA)*>
<!ATTLIST value field CDATA #REQUIRED>
```


PreInsert

```
<!ELEMENT siebel-xmlxt-preinsert-req (buscomp, remote-source?)>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED >
<!ELEMENT remote-source (#PCDATA)*>
```

Query

```
<!ELEMENT siebel-xmlxt-query-req (buscomp , remote-source?, max-
rows?, search-string?, match?, search-spec?, sort-spec? )>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED>
<!ELEMENT remote-source (#PCDATA)*>
<!ELEMENT max-rows (#PCDATA)>
<!ELEMENT search-string (#PCDATA)>
<!ELEMENT match (#PCDATA)>
<!ATTLIST match field CDATA #REQUIRED>
<!ELEMENT search-spec (node)>
<!ELEMENT node (#PCDATA | node)*>
<!ATTLIST node node-type (Constant | Identifier | Unary Operator |
Binary Operator) #REQUIRED>
<!ATTLIST node value-type (TEXT | NUMBER | DATETIME | UTCDATETIME |
DATE | TIME) #IMPLIED>
<!ELEMENT sort-spec (sort+)>
<!ELEMENT sort (#PCDATA)>
<!ATTLIST sort field CDATA #REQUIRED>
```

Update

```
<!ELEMENT siebel-xmlxt-update-req (buscomp, remote-source?, row)>
<!ELEMENT buscomp (#PCDATA)>
<!ATTLIST buscomp id NMTOKEN #REQUIRED>
<!ELEMENT remote-source (#PCDATA)*>
<!ELEMENT row (value+)>
<!ELEMENT value (#PCDATA)*>
<!ATTLIST value changed ( true | false ) #REQUIRED>
<!ATTLIST value field CDATA #REQUIRED>
```

Inbound DTDs

The following sections contain examples of DTDs representing the %methodName% response sent from the external application to the XML Gateway.

Delete Response

```
<!ELEMENT siebel-xmlxt-dekete-ret EMPTY >
```

Init Response

```
<!ELEMENT siebel-xmlxt-fields-ret (support+)>
<!ELEMENT support EMPTY >
<!ATTLIST support field CDATA #REQUIRED>
```

Insert Response

```
<!ELEMENT siebel-xmlxt-preinsert-ret (row)>
<!ELEMENT row (value+)>
<!ELEMENT value (#PCDATA)*>
<!ATTLIST value field CDATA #REQUIRED >
```

PreInsert Response

```
<!ELEMENT siebel-xmlxt-preinsert-ret (row)>
<!ELEMENT row (value)*>
<!ELEMENT value (#PCDATA)*>
<!ATTLIST value field CDATA #REQUIRED >
```

Query Response

```
<!ELEMENT siebel-xmlxt-query-ret (row*)>  
<!ELEMENT row (value+)>  
<!ELEMENT value (#PCDATA)*>  
<!ATTLIST value field CDATA #REQUIRED >
```

Update Response

```
<!ELEMENT siebel-xmlxt-update-ret (row)>  
<!ELEMENT row (value+)>  
<!ELEMENT value (#PCDATA)>  
<!ATTLIST value field CDATA #REQUIRED >
```

Index

Symbols

%methodName% request, sample inbound DTDs 187
%methodName% request, sample outbound DTDs 184

A

activating fields 55
AdminMode user property 62
AllLangIndependentVals user property 62
AllowedIntObjects business service user property 58
arguments
 EAI Siebel Adapter arguments, described 102
 Init method, XML Gateway business service 142
 IsPrimaryMVG 112
AssocFieldName user property 22, 60
Association user property 22, 59
association, defined 22

B

body data, contents of 15
buscomp Id tag 143
Business Component Id argument 142
Business Component Name argument 142
business components
 association, role of 22
 inner joins 29
 integration restrictions 65
 linking 26
 multi-value field example 24
 multi-value group example 27
 relation to business services 68

 specialized 136
 update permission rules 56
business objects
 as arguments to business service methods 82
 business service methods 82
 EAI Siebel Adapter, role of 100
 external data, creating from 100
 integration object maintenance, about 56
 relation to business services 68
 structure of 19
 user key requirement 32
business service methods
 arguments, defining 74
 Business Service Methods screen 78
 defining 74
 described 69
Business Service Simulator 80
business services 70
 argument types 75
 creating in Siebel Client 78
 creating in Siebel Tools, process overview 72
 customized business services, type of 69
 defined 68
 defining in Siebel Tools 73
 EAI MIME Hierarchy Converter 124, 127
 general uses 68
 importing and exporting 79
 predefined business services, table of 172
 and projects 73
 property set code example 81

- scripts, defining 75
- Specialized Business Services 68
- subsystems 76
- testing 80
- user properties, defining 77
- XML Gateway 140

C

- calculated fields 29
- child integration components
 - about 20
 - structure example 21
 - supported operations 109
- child property sets, about 71
- classes
 - classes and predefined business services 172
 - CSSBCVExtern 137
 - CSSBCVXMLExten 142
 - CSSEAIIDTScriptService 69
 - CSSEAISiebelAdapterService 100
- COM connection, external database and custom business service 150
- components, defined 14
- ContentId property 131
- ContentSubType property 132
- ContentType property 131
- CORBA connection, external database and custom business service 150
- CSEEAISiebelAdapterService 58
- CSSBCVExtern class 137, 150
- CSSBCVXMLExten class 142
- CSSBeginEndTransactionService 172
- CSSDataTransformationEngine 173
- CSSDllTransService 173
- CSSEAIbtsComService 173
- CSSEAIDispatchService 173
- CSSEAIIDTScriptService class 69
- CSSEAIImportExportService 173
- CSSEAIIntObjHierCnvService 173
- CSSEAIMimePropSetService 174
- CSSEAIMimeService 174
- CSSEAINullEnvelopeService 173

- CSSEAIQuerySpecService 173
- CSSEAISiebelAdapterService 172
- CSSEAISiebelAdapterService class 100
- CSSEAISMENvelopeService 173
- CSSEAIUtilService 172
- CSSEAIXMLCnvService 174
- CSSEAIXMLPrtService 174
- CSSHTTPTransService 172
- CSSMqAmiTransService 172
- CSSMqSrvTransService 172
- CSSMsmqTransService 172
- CSSSrmService 172
- CSSXMLCnvService 174
- CSSXSLTService 172
- custom business service
 - Delete method, example 152, 153, 154, 155, 156, 157, 159, 163, 166
 - external database, connection method 150
 - input and output parameters 151
 - PreInsert method, example 158
 - sample code 168

D

- data and arguments, contrasted 81
- Data Type Definitions. *See* DTDs
- databases
 - access, controlling 58
 - multi-valued attributes 23
- deactivating fields 55
- Delete method
 - custom business service example 152, 153, 154, 155, 156, 157, 159, 163, 166
 - DTD example 184
 - EAI Siebel Adapter 107
 - overview 107
 - SearchSpec input example 114
 - virtual business component example 152
 - XML code example 110
- Delete Response method, DTD example 187
- Display Name field 70

- docking, restrictions on 136
- DTDs
 - Integration Object Builder wizard,
 - about 18
 - sample inbound DTDs 187
 - sample outbound DTDs 184
 - DTTYPE_ATTACHMENT 123
- E**
 - EAI BTS COM Transport business
 - service 173
 - EAI Data Mapping Engine business
 - service 173
 - EAI Design project, editing integration
 - objects, warning 19
 - EAI Dispatch Service business service 173
 - EAI DLL Transport business service 173
 - EAI HTTP Transport 140
 - EAI HTTP Transport business service 172
 - EAI Import Export business service 173
 - EAI Integration Object to XML Hierarchy
 - Converter business service 173
 - EAI MIME Doc Converter business
 - service 174
 - EAI MIME Hierarchy Converter business
 - service 174
 - EAI MQSeries AMI Server Transport 140
 - EAI MQSeries AMI Transport business
 - service 172
 - EAI MQSeries Server Transport business
 - service 172
 - EAI MQSeries Transport 140
 - EAI MSMQ Transport 140
 - EAI MSMQ Transport business service 172
 - EAI Query Spec Service business
 - service 173
 - EAI Siebel Adapter
 - argument descriptions 102
 - database access, controlling 58
 - Delete method 107
 - described 100
 - Execute method 107
 - IsPrimaryMVG argument 112
 - method arguments 101
 - methods 101
 - multi-value groups 112
 - Query method 107
 - Synchronize method 105
 - transaction, defined 118
 - understanding method arguments 101, 108
 - Upsert method 106
 - EAI Siebel Adapter business service 172
 - EAI Siebel Wizard
 - about 54
 - integration objects, creating 41
 - EAI Transaction Service business
 - service 172
 - EAI Utility Service business service 172
 - EAI XML Converter business service 174
 - EAI XML Read from File business
 - service 174
 - EAI XML Write to File business
 - service 174
 - EAI XSLT Service business service 172
 - EAISubsys 76
 - error-text tag 147
 - Execute method 107
 - EAI Siebel Adapter 107
 - SearchSpec input example 114
 - specifying 109
 - Extension property 131
 - external application
 - custom business service and database,
 - connecting 150
 - data sharing, process overview 137
 - sample inbound DTDs 187
 - sample outbound DTDs 184
 - external data source, specifying 138
 - External Name user property 22
- F**
 - field, defined 14
 - FieldDependency user property 30, 61
 - fields
 - activating and deactivating 55
 - calculated 29
 - field dependencies 30

- inner join 29
- integration component fields,
 - viewing 43
- multi-value groups, working with 27
- property set fields 70
- user keys, about 31
- file attachments
 - See also* MIME
 - about using 120
 - data type 123
 - message types 121
- force active fields, performance
 - considerations 64
- foreign keys 34
- function code sample 82

H

- header data, contents of 15

I

- Ignore Bounded Picklist user property 60
- Init method
 - DTD example 184
- Init Response method, DTD example 187
- Inline XML attachments 121
- inner joins 29
- Input/Output type 75
- Insert method 156
- instance, defined 14
- integration component fields
 - defined 14
 - field names, assigning 28
 - multi-value groups, working with 27
- Integration Component Key. *See* user keys
- integration components
 - activating 54
 - child components, supported
 - operations 109
 - component fields, viewing 43
 - defined 14
 - deleting during synchronization 52
 - multi-value groups, working with 27

- selecting 42
- update permission rules 56

- integration messages
 - body data 15
 - defined 14
 - header data 15
- Integration Object Builder wizard
 - about 18
 - Code Generator wizard 18
 - EAI Siebel Wizard 54
 - Generate XML Schema wizard 18
 - integration components, selecting 42
 - integration objects, creating 41
 - user keys, about building 32
 - user keys, validating 32
- integration object instance
 - and actual data 16
 - defined 14
- integration objects
 - See also* child integration components;
parent integration components
 - about 15
 - calculated fields 29
 - choosing source object 41
 - complex hierarchy example 181
 - creating 41
 - defined 14
 - EAI Design project, editing warning 19
 - external data, creating from 100
 - in-memory updating 50
 - integration components, deleting during
synchronization 52
 - maintaining, about 56
 - metadata, about synchronizing 45
 - metadata, relation to 16
 - MIME message objects, creating 122
 - performance considerations 64
 - primaries, about setting 30
 - schema, generating 63
 - simple hierarchy example 179
 - structure example 21
 - terminology 14
 - update permission rules 56

- updating 46
- user properties, table of 59
- validating 44
- wizards process diagram 17
- integration projects
 - integration objects, use described 19
 - planning 15
- IntObjectName argument 101, 102
- IsPrimaryMVG argument 112

J

- Java class files, generating 18

L

- language-independent code 117
- links
 - and associations 22
 - between business components 26
 - update permission rules 56

M

- many-to-many relationships, virtual
 - business components 136
- MessageId argument 102, 103
- metadata
 - defined 14
 - integration objects, updating 46
 - processing example 81
 - relation to integration objects 16
 - synchronizing, integration objects,
 - about 45
- methods
 - business objects as arguments 82
 - business service method arguments,
 - defining 74
 - business service method arguments,
 - types of 75
 - business services methods, about 69
 - business services methods, defining 74
 - connecting custom business service and
 - external database 150
 - custom business services, input
 - parameters 151

- custom business services, output
 - parameters 151
- EAI Siebel Adapter method
 - arguments 101
- EAI Siebel Adapter, supported
 - methods 101, 109
- Execute, EAI Siebel Adapter 107
- incoming XML tags by method 147
- outgoing XML tags by method 143
- XML Gateway business service methods,
 - listed 142

MIME

- about 121
- inbound workflow process, creating
 - (example) 126
- integration objects, creating 122
- messages and hierarchies 130
- MIME hierarchy, converting to 127
- outbound workflow process, creating
 - (example) 123
- sending and receiving attachments,
 - process overview 121
- workflow process properties 124, 126

MIME Doc Converter

- about 130
- converting hierarchy to document 125
- converting to a hierarchy 127
- properties 131

MIME hierarchy

- converting hierarchy to document 125
- converting integration object to MIME
 - hierarchy 124
- converting to a hierarchy 127
- inbound transformation 129
- MIME Doc Converter 130
- outbound transformation 128
- property sets 130

MIME Hierarchy Converter

- business service 124, 127
- inbound transformation 129
- outbound transformation 128

- mobile users and virtual business
 - components 136

- Multi Value Link field 24
- Multipurpose Internet Mail Extensions. *See* MIME
- multi-value groups
 - See also* integration objects
 - EAI Siebel Adapter 112
 - example 24
 - field names, assigning 28
 - integration components, creating 27
 - multiple fields 26
 - primary record, setting 112
 - types of 23
 - update permission rules 56
 - virtual business components, restriction 136
- multi-value links, setting primaries 30
- multi-valued attributes 23
- MVG user property 59
- MVG. *See* multi-value groups
- MVGAssociation user property 22, 27, 60
- MVGFieldName user property 60
- N**
- name-value pairs
 - concatenating 140
 - role in property sets 71
- No envelope business service 173
- NoDelete user property 61
- NoInsert user property 60
- NoUpdate user property 61
- NumOutputObjects argument 101, 102
- O**
- ODBC connection, external database and custom business service 150
- Output Integration Object Name argument 102
- Output type 75
- P**
- Parameters argument 142
- parent business component
 - multi-value group example 27
 - multi-value group field names, assigning 28
- parent integration component
 - about 19
 - child integration component, supported operations 109
 - identifying 43
 - structure example 21
- performance
 - force-active fields, considerations 64
 - integration object considerations 64
 - picklist considerations 64
- PICKLIST user property 59
- picklists, performance considerations 64
- PreInsert method
 - custom business service example 158
 - DTD example 184, 185
 - virtual business component example 158
- PreInsert Response method, DTD example 187
- primaries, about setting 30
- primary business component 19
- primary integration component. *See* parent integration component
- PrimaryRowId argument 101, 102
- property sets
 - about 70
 - child 71
 - code sample 81
 - Delete method example 152, 153, 154, 155, 156, 157, 159, 163, 166
 - Display Name field 70
 - fields 70
 - Inputs 150
 - MIME hierarchy 130
 - nodes types 177
 - Outputs 150
 - PreInsert method example 158
 - role of 70
 - Update method example 165

as virtual business component
arguments 150

Q

Query method
description 107
DTD example 185
overview 104
SearchSpec input example 114
query operation
integration component keys, role of 31
role in integration projects 19
Query Response method, DTD
example 188

R

Remote Source argument 142
Remote Source user property 138, 140
REPOSITORY_BC_VIEWMODE_TYPE 58
root component. *See* parent integration
component
row tag 148

S

SAPSubsys 76
schema
Generate XML wizard 18
generating 63
scripts, attaching to business service 75
SearchSpec argument 102, 103, 114
Service Name user property 138, 140
Service Parameters 140
Service Parameters User Property 140
Service Parameters user property 138, 140
Siebel business component, defined 14
Siebel business objects
defined 14
structure of 19
Siebel Client, defining business
services 78
Siebel integration component field. *See*
integration component field

Siebel integration component. *See*
integration components
Siebel integration objects. *See* integration
objects
Siebel Message envelope business
service 173
Siebel Message object. *See* integration object
instance
Siebel Tools
business services, defining 73
creating business services, process
overview 72
integration objects, creating 41
user key, identifying 32
virtual business component,
creating 137
siebel.srf 134
SiebelMessage argument 103
siebel-xmlxt-delete-req tag 146
siebel-xmlxt-fields-req tag 143
siebel-xmlxt-fields-ret tag 148
siebel-xmlxt-Insert-req tag 145
siebel-xmlxt-insert-ret tag 149
siebel-xmlxt-preinsert-req tag 144
siebel-xmlxt-preinsert-ret tag 148
siebel-xmlxt-query-req tag 144
siebel-xmlxt-query-ret tag 148
siebel-xmlxt-status tag 147
siebel-xmlxt-Update-req tag 145
siebel-xmlxt-Update-ret tag 149
simulation, business service 80
source object, defining 41
Specialized Business Services 68
status keys 37
status-code tag 147
StatusObject argument 102, 103
synchronization process
about 45
in-memory updating 50
integration object components,
deleting 52
integration objects, updating 46
role in integration projects 19

- update rules, about 49
- Synchronize method, overview 105

T

- testing business services 80
- transaction, defined 118
- transports, used with XML Gateway 140

U

- Update method
 - DTD example 186
 - virtual business component example 165
- Update Response method, DTD example 188
- Upsert method
 - overview 106
 - XML code example 110
- user keys
 - about building with Object Builder wizard 32
 - building and validating, example 32
 - deactivating, warning 36
 - defined 31
 - definitions, confirming after build 40
 - field in Siebel Tools 32
 - foreign keys 34
 - Integration Component Key 32
 - locating in Tables screen 32
 - status key 37
 - validity, checking 33
- user properties
 - AssocFieldName 22
 - Association 22
 - business service user properties, defining 77
 - defining for virtual business components 138
 - External Name 22
 - FieldDependency 30
 - integration objects, table of 59
 - MVGAssociation 22

- transport protocol, virtual business components 140
- virtual business component user properties 138
- virtual business components 138

V

- value tag 148
- VBCs. *See* virtual business components
- ViewMode integration object user property 58
- ViewMode user property 62
- virtual business components
 - about 134
 - creating 137
 - CSSBCVExtern class 150
 - custom code example 168
 - docking restrictions 136
 - mobile users, restriction 136
 - multi-value groups 136
 - process overview 137
 - restrictions 136
 - specialized business components, restriction 136
 - transport protocol user properties 140
 - usage 136
 - user properties 138
 - user properties, defining 138
 - XML Gateway business service vs. custom business service 134
 - XML Gateway business service, configuring 140
- virtual business components, methods
 - Delete method example 152
 - Insert method example 156
 - PreInsert method example 158
 - Update method example 165

W

- Workflow business service subsystem, described 76

Workflow Process Manager (Server
Request) business service 172

workflows

- inbound MIME request 126
- outbound MIME request 123
- run-time event triggers 118
- workflow policy triggers 118

X

XML

- attribute-named operation,
specifying 109
- business services, importing 79
- Generate XML Schema wizard 18
- Inline XML attachments 121
- metadata example 81
- upsert and delete code example 110

XML Converter business service 174

XML Gateway business service

- about 140
- configuring 140
- incoming XML tags and
descriptions 147
- init method arguments 142
- methods 142
- outgoing XML tags and descriptions 143
- sample inbound DTDs 187
- sample outbound DTDs 184

XML Hierarchy Converter business
service 174

XMLCnv 76

