



TRANSPORTS AND INTERFACES:

**SIEBEL eBUSINESS
APPLICATION INTEGRATION
VOLUME III MIDMARKET EDITION**

VERSION 7.5

12-BCK96S

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	9
Additional Resources	10
Revision History	10

Chapter 1. EAI Transports and Interfaces Overview

EAI Transports	11
Transport Methods	13
Using Named Subsystems for Transport Parameters	15
Rules of Precedence for Parameter Specification	16
Common EAI Transport Parameters	16
Configuring Named Subsystems and Receiver Server Components	19
Object Interfaces	22
Database Level Interfacing	22

Chapter 2. EAI MQSeries Transports

Siebel EAI MQSeries Transports	23
EAI MQSeries Transport Named Subsystems	23
EAI MQSeries Server Transport	24
EAI MQSeries AMI Transport	27
EAI MQSeries Transport Re-Entrance	29

Chapter 3. EAI MSMQ Transport

About MSMQ	31
----------------------	----

About the eAI MSMQ Transport	32
Methods for Sending and Receiving Messages	32
MSMQ Transport Named Subsystems	34
Configuring the EAI MSMQ Transport Servers	35
MSMQ Primary Enterprise Controller	35
Regional Enterprise Server/MSMQ Client	35
NT Server Setup	36
Configuring EAI MSMQ Transport for Various Send and Receive Scenarios	38
MSMQ Transport Prerequisites	38
MSMQ Parameters	38
Defining Integration Objects	39
Sending Outbound Messages with MSMQ	40
Receiving Inbound Messages with MSMQ	46

Chapter 4. EAI HTTP Transport

About the EAI HTTP Transport	57
System Requirements	58
Selecting the Appropriate Business Service for HTTP	58
Using POST and GET	59
HTTP Transport Named Subsystems	60
General Information on How to Send a Message	61
Using the EAI HTTP Transport for Inbound Integration	62
Specifying HTTP Parameters for Inbound Integration	62
Calling EAI HTTP Transport Over a Network (HTTP Protocol)	64
Using the HTTP Transport for Inbound Messages	70
HTTP Transport Business Service Error Handling	74
How to Use Messages Returned from an External System	74
Using the EAI HTTP Transport for Outbound Messages	75
Specifying HTTP Transport Parameters for Outbound Integration	75
Examples Using HTTP Request	78
EAI HTTP Transport Method Arguments	80

Chapter 5. Using Siebel OLE DB Provider

Microsoft OLE DB	85
Siebel OLE DB Provider	85
Software Architecture	86
Understanding Siebel OLE DB Provider	89
What's in This Section	89
Siebel OLE DB Provider Configuration for Testing	90
Multiple Language Considerations	93
Primary and Foreign Key Relationships	93
Using the Windows Event Viewer with Siebel OLE DB Provider	94
Viewing OLE DB Information	95
Retrieving Siebel Data Using OLE DB Consumers	96
Creating and Modifying Siebel OLE DB Rowsets	97
Viewing Siebel OLE DB Rowsets in MS Office	101
Retrieving Siebel Data Using Scripts and Custom Applications	106
Writing an OLE DB Consumer	106
Retrieving Siebel Data Using VB and ASP	110
Troubleshooting	117

Chapter 6. Interfacing with Microsoft BizTalk Server

About Microsoft BizTalk Server	121
Siebel BizTalk Server Adapter	122
Where to Get More Information	123
Software Architecture	124
Schema Generation Support	125
Exchanging Integration Messages	126
Understanding Siebel BizTalk Server Adapter Through Scenarios	128
Preparing to Use the Siebel BizTalk Adapter	129
Installing and Configuring Software for Servers and Clients	130
Siebel Integration Objects	131
Connecting to BizTalk Using MSMQ	138
Siebel MSMQ Outbound Transport	138

Siebel MSMQ Inbound Transport	142
Connecting to BizTalk Using COM/AIC	148
Siebel COM Outbound Transport	148
Siebel AIC Inbound Transport	157
Connecting to BizTalk Using HTTP	174
Siebel HTTP Outbound Transport	174
Siebel HTTP Inbound Transport	180

Chapter 7. Integrating with J2EE Application Server

The Siebel JavaBean Wizard	189
Generating JavaBeans for Integration Objects	190
About the Folders and Files Created for Integration Object Java	192
Default Java Methods for Integration Objects	194
Default Java Methods for Integration Object Components	196
Generating JavaBeans for Business Services	197
About the Folders and Files Created for Business Service Java	199
Using the Java Data Bean	203
The Development Environment	203
Using Java Code Generated by Siebel eBusiness Applications	205
The Siebel Resource Adapter	211
Using the Resource Adapter	211
Using Java Code Generated by Siebel eBusiness Applications	214
Siebel Java/XML Framework	218

Index

Introduction

This guide explains the details of the transports and interfaces technology of Siebel eAI, MidMarket Edition, including MQSeries, MSMQ, OLE DB Provider, BizTalk interface, and Java Data Beans.

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.

Although job titles and duties at your company may differ from those listed in the following table, 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.
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 at an enterprise for Siebel applications.

- 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 the analysis of a business analyst to build the integration solution at an enterprise for specific applications and or to develop custom solutions.

The audience for this book also need 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 information on discreet components of the eAI transports and interfaces—such as MQSeries, BizTalk, J2EE, and so on—as individual chapters. Additional information, as applicable, can be found in the appendixes.

This book is Volume 3 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 the *Online Help*. The following are the integration related books and online help that describe all the tools required to implement integration:

- *XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition* serves as a reference if your integration involves the use of XML.
- *Siebel Tools Online Help, MidMarket Edition* serves as a reference if you plan on using COM, CORBA, or the ActiveX Plug-ins to accomplish integration and also as a reference for Siebel business objects and components.
- *Siebel Business Process Designer Administration Guide, MidMarket Edition*
- *Siebel Enterprise Integration Manager Administration Guide, MidMarket Edition* serves as a reference if you will perform bulk loading or unloading of data.

Revision History

Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition, Version 7.5

EAI Transports and Interfaces Overview

1

Siebel eAI provides three mechanisms for exchanging data between Siebel eBusiness Applications and external systems:

- **EAI Transports.** For information, see [“EAI Transports” on page 11](#).
- **Object Interfaces.** For information, see [“Object Interfaces” on page 22](#).
- **Database Level Interface and Batch Loading.** For information, see [“Database Level Interfacing” on page 22](#).

EAI Transports

Transports allow Siebel applications to exchange data with external applications using standard technologies for both synchronous and asynchronous communication protocols.

In Siebel 7.5, transports handle all data as binary data because the `IsTextData` parameter that was available in previous releases is no longer supported. If you want to use character conversion on the transport, you should use the `CharSetConversion` parameter. This functionality defers any character set conversion until needed, and avoids conversion at the transport level to prevent data corruption. For example, treating a UTF-8 XML as text when the conversion executes, will lead to an XML string in local code page while its header still describes UTF-8. It is recommended that all self-describing data including XML be treated as binary. For more information see *Overview: Siebel eBusiness Application Integration Volume I, MidMarket Edition*.

NOTE: For data validation or conversion from one encoding to another you can use the Transcode Business Service if needed. For details on the Transcode business service and its method, refer to *Global Deployment Guide, MidMarket Edition*.

Transports provide connectivity to virtually any communication protocol that can represent data as text or binary messages, including MQSeries from IBM, MSMQ from Microsoft, and HTTP. EAI Transports allow Siebel eBusiness Applications to integrate with Web-based applications as well as legacy systems that are encapsulated using middleware. Transports are interchangeable. If you change technologies at any point, you can reuse existing workflow processes and logic by switching the transport adapter.

Transports can:

- Support bidirectional exchange of messages
- Run within the Siebel Object Manager
- Invoke and be invoked by Workflow Process Manager and EAI Dispatch Service
- Be invoked within an eScript or VBScript
- Send and receive messages in XML format
- Pass messages through, or convert messages into, property sets for XML and MIME messages.

NOTE: EAI MQSeries Server/AMI Transport, EAI MSMQ Transport, and EAI File Transport business services are not re-entrant. For more information on transport re-entrance, see [“EAI MQSeries Transport Re-Entrance” on page 29](#).

Available transports include:

- EAI MQSeries Server Transport and EAI MQSeries AMI Transport. For information on these transports, see [Chapter 2, “EAI MQSeries Transports.”](#)
- EAI MSMQ Transport. For information on this transport, see [Chapter 3, “EAI MSMQ Transport.”](#)
- EAI HTTP Transport. For information on this transport, see [Chapter 4, “EAI HTTP Transport.”](#)
- EAI DLL Transport and EAI File Transport. For information on these transports, see [Chapter 5, “EAI DLL and EAI File Transports.”](#)

Transport Methods

The method on a transport adapter's business service controls the action to be performed by the transport. There are two outbound methods and three inbound methods available for EAI Transports. Not every method is available on every transport.

For each method, there are a number of common parameters, as shown on [Table 2 on page 17](#), as well as transport-specific parameters that are discussed in the respective chapter for each transport.

Outbound Methods

Available outbound methods depend on the transport business service in use, such as EAI MQSeries AMI Transport or EAI MSMQ Transport. The business service sends messages from Siebel applications using the appropriate communications protocol, such as MQSeries, MSMQ, HTTP, and so on. There are two outbound methods that you use to send requests from Siebel application to another application:

- **Send**

Sends a message from Siebel application when Siebel application does not need a response. This is an asynchronous request method, because the Siebel application does not need to wait for a response before continuing with the process.

- **Send and Receive** (SendReceive)

Sends a message from the Siebel application when Siebel application needs to receive a response before continuing. This is a synchronous request and response method, because it requires a response before the Siebel application can continue the process.

Inbound Methods

Available inbound methods depend on the transport business service in use, such as EAI MQSeries AMI Transport or EAI MSMQ Transport. The inbound methods monitor a specified queue and upon receipt of a message, dispatch it to another service.

There are three inbound methods that can be used to receive requests from another application:

- **Receive**

Receives an inbound request message and returns it to the caller of the transport.

- **Receive and Execute** (ReceiveDispatch)

Receives an inbound request message and calls another service with the inbound message as input. This called service is known as the Dispatch Service, and the method that is called is known as the Dispatch Method.

- **Receive, Execute, Send** (ReceiveDispatchSend)

This is a request/response method. Receives an inbound request message, calls another service with the inbound message as input, and then sends the output of the called service as a response.

NOTE: There are server components (called receivers) on top of the inbound methods that run as Siebel Server tasks. If when running an EAI receiver such as the SAP IDOC Receiver, MQSeries Server, MQSeries AMI Receiver, MSMQ Receiver using the methods ReceiveDispatch or ReceiveDispatchSend the dispatch service has an error, the receiver shuts down. Check the Status column on the Component Tasks for details about the cause of the error.

Using Named Subsystems for Transport Parameters

Named subsystems are groupings of defined enterprise parameters that are stored in the Siebel Gateway Server. You use named subsystems to specify methods and parameters for EAI Transports. Transport business services take two subsystem names as parameters, which you define using the Siebel Server Manager:

- **Transport Connection Subsystem** (ConnectionSubsystem)
- **Transport Data Handling Subsystem** (DataHandlingSubsystem)

Values for parameters in a named subsystem are common to every user of the subsystem across the enterprise. Subsystem names themselves are parameters for server components. You can logically group parameters into various subsystems.

NOTE: You can specify the transport-named subsystem parameters either as business service user properties or as runtime arguments. However, if you specify parameters in both locations, only the parameters found in the business service user property are actually used. For additional information on named subsystems, see *Siebel Server Administration Guide, MidMarket Edition*.

For the two EAI Transport named subsystem parameters, ConnectionSubsystem and DataHandlingSubsystem, two parameters exist for the EAI receivers—ReceiverConnectionSubsystem and ReceiverDataHandlingSubsystem. The EAI Receiver looks up these parameters from the server component parameters and copies the corresponding properties (ConnectionSubsystem and DataHandlingSubsystem) to the input property set of the transport business service.

NOTE: Parameters specified in business service user properties will no longer work “as is.” You need to create named subsystems and specify the parameters for the subsystems. Then, you need to specify the named subsystems you created as business service user properties in a workflow or through an eScript, or the other usual means. Note that business service user properties will work for the SAP Connector.

Rules of Precedence for Parameter Specification

You can specify the two named subsystem parameters, Connection Subsystem and Data Handling Subsystem, as either business service user properties or as runtime arguments. If you specify the parameters in both locations, the business service user property will take precedence over the runtime arguments.

You specify every other parameter in one of the two named subsystems or as runtime arguments. Siebel eAI looks for the parameter in the Connection Subsystem or the Data Handling Subsystem, depending on which parameter it is. If you specified the appropriate named subsystem, Siebel eAI will always look for the parameter there.

If you do not specify the parameter in this named subsystem—even if you specified it as a runtime argument—the runtime specification will be ignored. Siebel eAI will look for the parameter in a runtime specification only if no appropriate named subsystem is specified.

Common EAI Transport Parameters

To configure the EAI Transports, you will need to create named subsystems for data handling and connection parameters, as listed in [Table 1](#). The data handling parameters are listed in [Table 2 on page 17](#). These parameters are common to every Transport method. After you create the named subsystems, you will then need to specify these named subsystems as parameters in the service method argument or the business service user property.

Table 1. Named Subsystem Parameters for Transport Methods

Named Subsystem Parameter Name	Description
ConnectionSubsystem	Name of a named subsystem for transport-specific connection parameters.
DataHandlingSubsystem	Name of a named subsystem for transport data handling parameters.

Table 2. Common Data Handling Parameters for Transport Methods

Transport Method Parameter Name	Description
CharSetConversion	<ul style="list-style-type: none"> ■ Default is None and should be used for self-describing content such as XML and MIME. ■ Legal values are None, UTF-8, and UTF-16. CharSetConversion specifies if and how character set conversion needs to occur before or after sending or receiving data from the external system. ■ Depending on the value of this parameter, transport business services do implicit character set conversions if necessary. ■ Same CharSetConversion is assumed for requests and responses.
ConverterService	<ul style="list-style-type: none"> ■ ConverterService is the name of the business service to use for serializing property sets to a buffer and unserializing buffers to property sets. ■ ConverterService receives arguments through business service user properties if the converter service can accept them. ■ Note that not any arbitrary service may be designated to be a converter service.
DispatchMethod	<ul style="list-style-type: none"> ■ DispatchMethod parameter specifies the dispatch method. ■ Specification of DispatchService is mutually exclusive with specification of a DispatchRuleSet or a DispatchWorkflowProcess. ■ This parameter is only applicable for the ReceiveDispatch and ReceiveDispatchSend methods.
DispatchRuleSet	<ul style="list-style-type: none"> ■ DispatchRuleSet specifies the name of the dispatch rule set for the Dispatcher Service. ■ Specification of DispatchRuleSet is mutually exclusive with specification of DispatchWorkflowProcess or Dispatch Service. ■ This parameter is only applicable for the ReceiveDispatch and ReceiveDispatchSend methods.

Table 2. Common Data Handling Parameters for Transport Methods

Transport Method Parameter Name	Description
DispatchService	<ul style="list-style-type: none">■ DispatchService specifies the dispatch service.■ Specification of DispatchService is mutually exclusive with specification of a DispatchRuleSet or DispatchWorkflowProcess.■ This parameter is only applicable for the ReceiveDispatch and ReceiveDispatchSend methods.
DispatchWorkflow Process	<ul style="list-style-type: none">■ This parameter specifies the name of the workflow process to dispatch to.■ Specification of DispatchWorkflowProcess is mutually exclusive with specification of DispatchRuleSet or Dispatch Service.■ This parameter is only applicable for the ReceiveDispatch and ReceiveDispatch Send methods.
IgnoreCharSetConv Errors	<ul style="list-style-type: none">■ Default value is False.■ IgnoreCharSetConvErrors specifies whether character set conversion errors should be ignored. If false, with any such errors, the transport service propagates the error.
RollbackOnDispatch Error	<ul style="list-style-type: none">■ Default value is True.■ Rollback transport transaction if a Dispatch Method fails.■ This parameter is only available for the transactional transports MQSeries Server, MQSeries AMI, and MSMQ.
SiebelTransactions	<ul style="list-style-type: none">■ Default value is True.■ Nest the Siebel transaction within the transport transaction.■ This parameter is only available for the transactional transports MQSeries Server, MQSeries AMI, and MSMQ.

Configuring Named Subsystems and Receiver Server Components

This section explains how to configure named subsystems and receiver server components.

To configure named subsystems

- 1** Navigate to the Server Administration view.
- 2** Select Enterprise Configuration from the Server Administration view.
- 3** Click the Enterprise Component Groups tab.

A list of receivers appears in the Enterprise Component Groups applet, for example, MQSeries AMI Receiver.
- 4** Select Components from the Server Administration view.
- 5** Click the Component Parameters tab.
- 6** Select the MQSeries AMI Receiver in the Server Components List applet.
- 7** Query for the Default Tasks parameter in the Component Parameters applet.
- 8** Set the value for the Default Tasks parameter to 1, or any number equal to the number of tasks that will be started for the component.

To create a connection subsystem

- 1** Navigate to the Server Administration view.
- 2** Select Enterprise Configuration from the Server Administration view.
- 3** Click the Enterprise Profile Configuration tab.
- 4** In the Component Profiles List applet, create a new Named Subsystem record.
 - a** Enter MyAMISubsys in the Named Subsystem Alias column.
 - b** Select MQSeriesAMISubsys as the type.
 - c** In the Enterprise Profile Configuration applet, specify the values for each of the connection parameters for the AMI receiver, such as MqLocalHostFileName.

To create a data handling subsystem

- 1** Navigate to the Server Administration view.
- 2** Select Enterprise Configuration from the Server Administration view.
- 3** Click on the Enterprise Profile Configuration tab.
- 4** In the Component Profiles List applet, create a new Named Subsystem record.
 - a** Enter MyDispSubsys in the Named Subsystem Alias column.
 - b** Select EAITransportDataHandlingSubsys as the type.
- 5** In the Enterprise Profile Configuration applet, specify the values for each of the data handling parameters for the AMI receiver, such as Dispatch Service.

To start the Receiver Server task

- 1** Navigate to the Server Administration view.
- 2** Select Servers from the Server Administration view.
- 3** Click the Server Components tab.
- 4** In the Server Components applet, select the MQSeries AMI Receiver.
- 5** Click the Shutdown button.

Wait for the component state to change to Shutdown.

- 6** Click Startup to restart the Receiver.

Observe that the Component State now changes to Online, and the value for Running Tasks changes to the number of Default Tasks specified previously.

To specify named subsystems in a workflow

- 1** Navigate to the Workflow Process Designer.
- 2** Query and select the workflow process named MyMQOutbound.
- 3** Double-click the MQ Send step.
- 4** In the Input Arguments Applet, insert a new record to specify the Connection subsystem, and specify the value as MyAMISubsys.
- 5** Insert another record to specify the Data Handling Subsystem, and specify the value as MyDispSubsys.
- 6** Insert a third record to specify the Message Text.

Object Interfaces

Object Interfaces allow integration between Siebel applications and external applications. Object Interfaces can be called by eScripts or used within a workflow. The workflow can use other business services and transports as needed. For example, the BizTalk interface might use MSMQ Transport as part of its workflow.

Available object interface support includes:

- Siebel eBusiness Applications as a Microsoft OLE DB Provider. For information, see [Chapter 5, “Using Siebel OLE DB Provider.”](#)
- Siebel eBusiness Applications interfacing with Microsoft BizTalk. For information, see [Chapter 6, “Interfacing with Microsoft BizTalk Server.”](#)
- Siebel eBusiness Applications that produce Java Data Beans to support J2EE applications. For information, see [Chapter 7, “Integrating with J2EE Application Server.”](#)

Database Level Interfacing

In addition to Transports and Object Interfaces, Siebel application provides Enterprise Integration Manager (EIM) for high volume data exchange and batch loading. You use the set of interface tables that serve as intermediate tables between your external data source and the Siebel database tables.

NOTE: See *Siebel Enterprise Integration Manager Administration Guide, MidMarket Edition* for details.

This chapter discusses the EAI MQSeries Transports.

Siebel EAI MQSeries Transports

This section assumes that you understand the architecture and operation of IBM MQSeries and the MQSeries Application Messaging Interface (AMI). The EAI MQSeries Server transport uses the Message queuing API (MQI), and the EAI MQSeries AMI Transport interfaces with IBM MQSeries using the Application Messaging Interface (AMI). For more information, consult the IBM MQSeries documentation.

Siebel EAI provides two MQSeries transports:

- EAI MQSeries Server Transport
- EAI MQSeries AMI Transport

NOTE: The Siebel business service “EAI MQSeries Transport,” which was provided in Siebel 6.x releases is not available in the Siebel 7.x release. Customers of previous Siebel versions using this name should upgrade their workflows to use the “EAI MQSeries Server Transport” name.

EAI MQSeries Transport Named Subsystems

The EAI MQSeries Transports can read parameters from a named subsystem. For MQSeries Server Transport, the named subsystem type is MqSeriesServerSubsys. For MQSeries AMI Transport, the named subsystem type is MqSeriesAMISubsys.

- For a discussion of named subsystems for Siebel eAI, see [Chapter 1, “EAI Transports and Interfaces Overview.”](#)
- For more information on named subsystems, see *Siebel Server Administration Guide, MidMarket Edition*.

EAI MQSeries Server Transport

Prior to using the EAI MQSeries Server Transport, you will need to install and configure IBM MQSeries software. Contact your IBM sales representative for details.

Configuring the EAI MQSeries Server Transport

The Siebel EAI MQSeries Server Transport is designed to provide a messaging solution to help you integrate data between Siebel eBusiness Applications and external applications that can interface with the IBM MQSeries. The EAI MQSeries Server Transport transports messages to and from IBM MQSeries queues.

NOTE: The EAI MQSeries Server Transport can connect only to IBM MQSeries Server software. The EAI MQSeries AMI Transport can connect to IBM MQSeries server as well as the IBM MQSeries client. The IBM MQSeries server must be running on the same system with your Siebel Application Server.

The EAI MQSeries Server Transport supports all of the inbound and outbound methods described in the [“Outbound Methods” on page 13](#) and [“Inbound Methods” on page 14](#).

Using the SendReceive Method with MQSeries

The SendReceive method on the EAI MQSeries Server Transport sends a message and waits for a response from the target application on a response queue. This response message corresponds to the original message using the correlation ID in MQSeries.

NOTE: It is the responsibility of the external application to set the correlation ID of the response to the Siebel eBusiness Application to the message ID of the original message.

EAI MQSeries Server Transport Parameters

In addition to supporting the common transport parameters described in [Table 2 on page 17](#), the EAI MQSeries Server Transport takes the parameters shown in [Table 3](#). These can be specified as either service method arguments, subsystem parameters, or user properties.

NOTE: In order to send to a model queue, the model queue must have a definition type of PERMANENT and the following arguments must be supplied in the workflow process: Model Queue, Physical Queue, Queue Manager, and Message Text.

Table 3. EAI MQSeries Server Transport Specific Parameters

Argument	Display Name	Description
MqAcknowledgements	Receive Acknowledgements	Set to FALSE by default, this parameter specifies whether or not delivery and arrival acknowledgements are to be received.
MqAckPhysicalQueueName	Acknowledgement Physical Queue Name	If MqAcknowledgements is set to TRUE, this parameter contains the name of the physical queue for acknowledgements to responses.
MqAckQueueManagerName	Acknowledgement Queue Manager Name	If MqAcknowledgements is set to TRUE, this parameter contains the name of the queue manager for acknowledgements to responses. If unspecified, it defaults to MqQueueManagerName.
MqModelQueueName	Model Queue Name	Name of the MQSeries model queue.
MqPhysicalQueueName	Physical Queue Name	Name of the MQSeries physical queue.
MqQueueManagerName	Queue Manager Name	Name of the MQSeries queue manager. If this parameter is not specified, the Default Queue Manager name, as specified in the MQSeries configuration, is used.
MqRespModelQueueName	Response Model Queue Name	Name of model queue for response connection.
MqRespPhysicalQueueName	Response Physical Queue Name	Name of physical queue for response connection.

Table 3. EAI MQSeries Server Transport Specific Parameters

Argument	Display Name	Description
MqFormat	MQSeries Format	The format of the message from the Siebel application to the outbound queue.
MqSleepTime	Sleep Time	The timeout interval on receive calls, in milliseconds. The default value is 20000 milliseconds.

In addition to the EAI MQSeries Server Transport, you can run the MQSeries Server Receiver, which is a server component that periodically checks the MQSeries queues you specify, for inbound messages.

Dispatch Error Handling for the EAI MQSeries Server Transport

When using `ReceiveDispatch` and `ReceiveDispatchSend` methods, you need to be aware of specific MQSeries behavior that might affect your messages.

NOTE: The transaction does not end when the message is received from the queue because it waits for the entire dispatch process to either complete successfully for commit or fail for rollback.

If all of the following conditions are met, the message is sent to the Backout Requeue Queue of the current queue manager:

- A dispatch error has occurred.
- The `RollbackOnDispatchError` property is set to `TRUE`.
- The message has been rolled back by a count exceeding the Backout Threshold of the queue.

NOTE: If the Backout Requeue Queue has not been specified for the Queue Manager, then the message is sent to the Dead Letter Queue of the current queue manager. If there is no specified Dead Letter Queue for the current queue manager, then the queue defaults to the `SYSTEM.DEAD.LETTER.QUEUE`.

EAI MQSeries AMI Transport

When using the EAI MQSeries AMI Transport, you will need to install and configure:

- MQSeries Server
- MQSeries Client if necessary
- MQSeries Application Messaging Interface and
- MQSeries Application Messaging Interface Administration Tool (available only on Windows NT and Windows 2000)

Contact your IBM sales representative for details.

Configuring the EAI MQSeries AMI Transport

If you are using the MQSeries Application Messaging Interface (AMI), you will specify the queues, queue managers, and other parameters in the AMI Administration Tool, when you define your AMI service points and AMI policies.

NOTE: Service points define where the message needs to be sent. Policies define how the messages need to be handled. It is very important to understand what a policy is and how to set different policy elements. For more information on AMI policies, consult your IBM documentation.

EAI MQSeries AMI Transport Parameters

After defining the AMI policy and AMI service points, you must set certain enterprise parameters in the Siebel Client, as shown in [Table 4](#).

Table 4. EAI MQSeries AMI Transport Specific Parameters

Argument	Display Name	Description
MqPolicyName	Policy Name	Optional. The name of the policy you define in the AMI Administration Tool. The AMI policy defines properties such as Connection Type and Mode, Message Type, Send and Receive parameters, and many others. The policy is stored in the AMI Repository, which is an XML document. You can have multiple policies and multiple repositories. Refer to the IBM MQSeries Application Messaging Interface (AMI) documentation for more details. If you do not specify an AMI Policy, this parameter defaults to the AMT.SYSTEM.SYNCPOINT.POLICY provided by IBM.
MqReceiverServiceName	Receiver Service Name	The name of the service point you define as a receiver in the AMI Administration Tool. Service points specify queue names, queue manager names, model queue names, and several other parameters. You must create a service point that specifies the queue and queue manager that will receive messages. The Receiver Service and the Sender Service may specify the same queue and queue manager.
MqRepositoryName	Repository Name	Optional. Fully qualified name of the directory containing AMI repository files. If unspecified, the business service looks for the repository files in the default location determined by the MQSeries AMI installation.
MqSenderServiceName	Sender Service Name	The name of the service point you define as a sender in the AMI Administration Tool. Service points specify queue names, queue manager names, model queue names, and several other parameters. You must create a service point that specifies the queue and queue manager that will send messages. The Receiver Service and the Sender Service may specify the same queue and queue manager.
MqTrace	Trace	Turns on tracing for AMI (Default is False).
MqTraceLocation	Trace Location	Name of the directory for output of AMI tracing.

NOTE: Do not set the Receive attribute for the AMI policy as “Convert,” if you have set the CharacterSetConversion parameter from the data handling subsystem to TRUE. Also, do not set this attribute if receiving well-formed, self-describing documents such as XML.

Using the ReceiveDispatchSend Method with AMI

When using the ReceiveDispatchSend method of the EAI MQSeries AMI Transport, you must observe the following rule in your MQSeries configuration:

The request message to be put onto the MQSeries queue must contain values for the ReplyToQueue and ReplyToQueueManager parameters.

EAI MQSeries Transport Re-Entrance

The EAI MQSeries Server Receiver uses the “EAI MQSeries Server Transport” business service but cannot dispatch to a workflow that either uses this business service as one of its steps or dispatches directly to this business service. While in-process re-entrance is not supported, you can indirectly invoke the “EAI MQSeries Server Transport” as one of the steps out of process by calling the “Synchronous Server Requests” business service. This limitation with the suggested workaround also applies to the EAI MQSeries AMI Transport.

To use in-process nested MQSeries functionality, you can set the EAI MQSeries Server Receiver to dispatch to the “EAI MQSeries AMI Transport” business service as one of its steps as long as in the nested step the AMI policy you use has Syncpoint parameter unchecked.

This chapter discusses Siebel Systems' implementation of Microsoft MSMQ support with the MSMQ Transport.

About MSMQ

Today, many large organizations are integrating various enterprise business applications into application networks. These networks allow applications to “talk” to each other and share data, either automatically or by request. Technologies such as Microsoft Message Queuing (MSMQ) provide a messaging infrastructure for transporting data from one application to another, without the need for programming.

MSMQ allows applications running at different times to communicate across heterogeneous networks and systems, even when one or many of those systems are temporarily offline. Because applications send messages to queues and read messages from queues, the messages are always available and remain in the queue for as long as required. For example, the messages will still be there when a system that was offline comes back online to retrieve them. Optionally, messages can be sent to a “dead letter” queue after a predetermined amount of time has passed to help make sure that only timely, relevant messages are received.

About the eAI MSMQ Transport

The EAI MSMQ transport is a Siebel Business Service that can be customized using Siebel Tools MidMarket Edition. With Siebel Tools, you define Integration Objects to be transported across the EAI MSMQ Business Service. The EAI MSMQ transport is responsible for sending and receiving messages between Siebel applications and MSMQ queues. The EAI MSMQ transport allows you to:

- Send a message to an external system
- Send and receive synchronous messages between a Siebel application and an external system
- Receive a message and perform an action based on that message within a Siebel application
- Receive a message, perform an action within a Siebel application, and then send a synchronous response to the external system

Methods for Sending and Receiving Messages

The EAI MSMQ transport supports two transport modes: Sending Messages and Receiving Messages. Each supports the following methods.

- Sending Messages Methods
 - Send
 - Send and Receive Response
- Receiving Messages Methods
 - Receive
 - Receive and Execute Service
 - Receive, Execute, Send Response

Messages from a Siebel Application to an External System

You configure EAI MSMQ transport using the Siebel Workflow Process Designer, where you specify various parameters, such as the queue where Siebel outbound messages should be sent. You configure the message itself using the Integration Object feature within Siebel Tools. The message can be in any text or binary format, including XML. The default format is XML, where the Integration Object defines the XML Schema Definition (XSD) or the Document Type Definition (DTD) associated with the XML document.

You configure the EAI MSMQ transport at design time to specify the MSMQ queue machine name and the queue name. You use the eAI MSMQ transport along with the new Siebel Workflow Process Manager to model business processes for sending messages to the external system.

You can configure the EAI MSMQ transport to send messages to external systems when an event occurs in Siebel application. For example, suppose that one of your sales representatives enters a new opportunity for an account into a Siebel application. This information needs to be sent to other business units that may or may not be using a Siebel application. The message can be sent using the eAI MSMQ transport as the transport mechanism to inform these external systems.

The EAI MSMQ transport can also be used synchronously to send a message and receive a response back from an external system in a single session. For example, suppose that one of your customers calls your Call Center requesting information on an account. The sales agent initiates a process to send a request with the Account Name from a Siebel application to an external mainframe system using the eAI MSMQ transport. The sales agent then receives a list of transaction details for that customer displayed within a Siebel application form in response.

Messages to a Siebel Application from an External System

External applications can send messages to Siebel applications using the EAI MSMQ transport. These messages are received and routed by the EAI MSMQ Receiver in conjunction with the MSMQ system.

The EAI MSMQ Receiver is a Siebel Server component that waits for messages in a specified queue. If you select the Receive, Execute, Send Response method, the EAI MSMQ Receiver waits for a response from a Siebel application and places the output into a response queue.

MSMQ Transport Named Subsystems

The EAI MSMQ Transport can read parameters from a named subsystem. For this transport, the named subsystem type is MSMQSubsys.

- For a discussion of named subsystems for Siebel eAI, see [Chapter 1, “EAI Transports and Interfaces Overview.”](#)
- For more information on named subsystems, see *Siebel Server Administration Guide, MidMarket Edition*.

Configuring the EAI MSMQ Transport Servers

The instructions in this section are for configuring the EAI MSMQ transport servers. You should use a two-server setup, configured as listed in the following section. However, you can implement a single server or multiple servers.

MSMQ Primary Enterprise Controller

You configure the MSMQ Primary Enterprise Controller with the following components.

- One of the following Servers: Microsoft Windows NT Server Enterprise Edition, Windows 2000 Server, or Windows 2000 Advanced Server
- MSMQ Server
- As many MSMQ Queues as needed
- Relevant ODBC driver
- Siebel Application Server
- Siebel Gateway Server
- Siebel Web Client
- Siebel Tools

Regional Enterprise Server/MSMQ Client

You configure the Regional Enterprise Server/MSMQ Client with the following components.

- One of the following Servers: Microsoft Windows NT Server Enterprise Edition, Windows 2000 Server, or Windows 2000 Advanced Server
- MSMQ Client
- As many MSMQ Queues as needed
- Relevant ODBC driver
- Siebel Application Server

- Siebel Gateway Server
- Siebel HTML Connected Client

NOTE: MSMQ Server can reside on either machine. This functionality is independent of the underlying database platform. You can use any of the supported database platforms, including IBM UDB Oracle, and Microsoft SQL Server.

NT Server Setup

This section contains instructions on how to set up the NT server for use with MSMQ. Note that the steps below apply to Windows NT installations only. Windows 2000 Server and Windows 2000 Advanced Server both install MSMQ automatically.

NT Server Setup Prerequisites

Before using the configuration steps below, install Windows NT Option Pack for NT Server and restart the machine.

To set up the MSMQ Primary Enterprise Controller (PEC)

- 1** Choose Start > Programs > Windows NT 4.0 Option Pack Setup.
- 2** Choose Add/Remove Components.
- 3** Click Microsoft Message Queue, uncheck all other options, and click Next.
- 4** Click Primary Enterprise Controller to install MSMQ Server.
 - a** Specify a name for the Enterprise to be created, such as “Siebel.”
 - b** Specify the site to be created, such as “Sales.”
 - c** Leave all other default MSMQ parameters as is.
- 5** Click Connected Network as the machine on which MSMQ PEC is installed.

To set up the MSMQ Client

- 1** Choose Start > Programs > Windows NT 4.0 Option Pack Setup.
- 2** Choose Add/Remove Components.
- 3** Click Microsoft Message Queue, uncheck all other options, and click Next.
- 4** Click Independent Client and then click Site Controller as the machine on which MSMQ PEC is installed.

Configuring EAI MSMQ Transport for Various Send and Receive Scenarios

The EAI MSMQ transport and the Siebel Workflow Process Manager work in tandem to transfer data using MSMQ from one Siebel application to another Siebel application or to an external application. You can set up a workflow and choose attributes and values to define the transport for a particular send or receive scenario.

MSMQ Transport Prerequisites

You must be set up both Microsoft SQL Server and MSMQ prior to configuring the EAI MSMQ Transport. In addition, Siebel Workflow functionality should be available within Siebel Tools and Siebel Web Client.

MSMQ Parameters

[Table 5](#) identifies and explains the parameters used for configuring the EAI MSMQ transport.

Table 5. EAI MSMQ Transport Parameters

Parameter	Description
EndOfData	Should be set to True to indicate end of data.
MsmqPhysicalQueueName	Name of the MSMQ Queue. Can be used for both sending and receiving messages.
MsmqQueueMachineName	Machine that owns the queue specified by the physical queue name.
MsmqRespQueueMachineName	Machine that owns the queue specified by MsmqRespQueueName.
MsmqRespQueueName	Name of the response queue.
MsmqSleepTime	The amount of time that the MSMQ business service will wait to receive a message. Default value is 20000 milliseconds.
TimedOut	If no message is received in seconds specified in SleepTime, the TimedOut argument in the Output Property set will be set to True.

Table 5. EAI MSMQ Transport Parameters

Parameter	Description
IgnoreCorrelationId	Set to ignore Correlation Id value on the inbound messages. If this flag is true, the message is picked up from the queue regardless of the correlation Id on the message. This parameter is ignored for the SendReceive Method because Correlation Id is required to match the response with the original message. The default value is FALSE. This parameter must be set to TRUE to support BizTalk integration.
LargeMessageSupport	Set to enable or disable Large Message (messages over 4MB) Support. The default value is set to TRUE. IgnoreCorrelationId should be flagged FALSE for Large Message Support.

Defining Integration Objects

Before you use the EAI MSMQ transport, define Integration Objects for use with the transport. The various methods explained in the following pages assume that this Integration Object has already been defined. You define your Siebel messages as Integration Objects using Siebel Tools. These messages correspond to the information that you want to exchange between the Siebel application and an external application. An example of an integration object would be an order, an account, a quote, or a contact.

The following procedure provides you with the general flow for creating Integration Objects for use with the EAI MSMAQ transport. For more detailed information, refer to *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition*.

To define an integration object

- 1 Start Siebel Tools.
- 2 Lock the project you want in Tools (for example, the Contact Project).
- 3 Click New and choose Integration Object.

The Integration Object Wizard appears.
- 4 Select Contact as the project and EAI Siebel Wizard as source system.

- 5 Select Contact as source object and give a unique name for this integration object. Click Next, deactivate the components that are not needed, and click Finish.
- 6 Compile the `.srf` file.

After you have created an Integration Object, you can then send the message corresponding to this Integration Object through the EAI MSMQ transport, either as part of a business process flow (using Siebel Workflow Process Manager), or as a custom business service.

NOTE: For more information on Integration Objects, see *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition*. For information on Siebel Tools, see *Siebel Tools Online Help, MidMarket Edition*.

Sending Outbound Messages with MSMQ

With the Siebel application as the sender (outbound messaging), you design a Workflow process that queries for a record (such as a Contact) and then converts that record to an XML document. The XML document is then sent to an MSMQ queue.

Because MSMQ imposes a limit of four megabytes on the size of the messages it can handle, the eAI MSMQ transport separates outbound Siebel messages larger than four megabytes into smaller messages acceptable to MSMQ. The message is then reassembled after it has left MSMQ and arrived at your partner's system.

There are two methods for sending messages from Siebel application to MSMQ:

- Send
- Send and Receive Response (SendReceive)

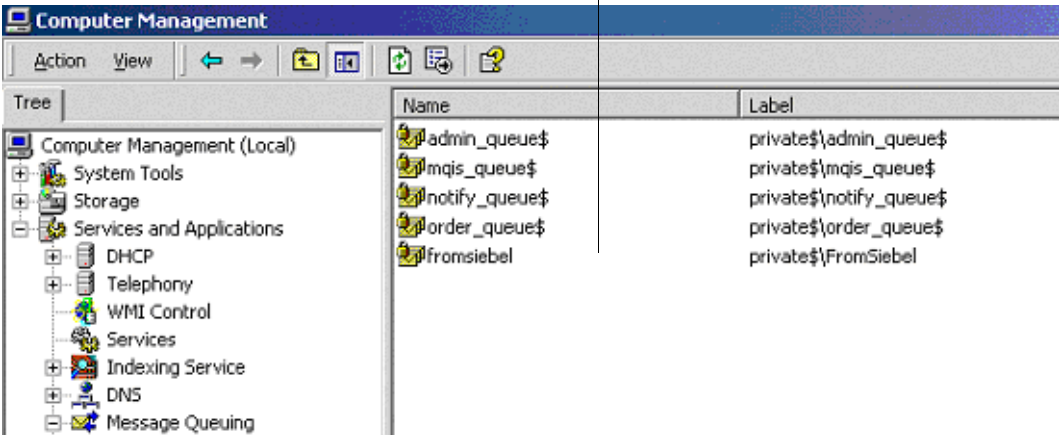
Sending Messages with MSMQ

The following procedure describes how to set up your system to send a message to an external system using MSMQ.

To send messages from a Siebel application to MSMQ

- 1 Access the Windows Computer Management tool.
 - On Windows NT, choose:
Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer
 - On Windows 2000, choose:
Start > Programs > Administrative Tools > Computer Management
- 2 Set up an MSMQ queue to receive messages from the Siebel application. Give the queue an easy to identify name, such as “fromsiebel,” as shown in the following illustration.

The MSMQ queue you create will appear in the list of queues.



- 3 Set the queue to be Transactional.

NOTE: This flag allows Siebel applications to group a number of Send or Receive messages. This is critical when large data sets are being used because it will allow a Commit or Rollback to be executed without failure.

- 4 Choose Start > Programs > Siebel HTML Connected Client > Workflow Process Administrator > Workflow Process Designer.
- 5 Set up a workflow for sending a message to MSMQ. Define the flow as shown in the following flowchart.



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 6 Double-click the eAI MSMQ Transport (Send) object. In the Service Methods applet, select EAI MSMQ Transport from the Service Name LOV and enter the Method Name, Send.
- 7 Click the Input Arguments applet to create a new record with Type of PropertySet.
- 8 Set the Value to the string (text) that you want to send to MSMQ, such as “Test Message from Siebel.”

NOTE: You can alternately open a text file or XML file. For example, you can open a file in a text editor and copy and paste the contents into the Value field.

9 Click the Input Arguments applet and set two (2) parameters as follows:

- a** Add an argument to the property set to specify the queue name, `MsmqPhysicalQueueName`.

The value of the argument should be set to the queue name. For example:

```
Name = MsmqPhysicalQueueName  
Value = FromSiebel
```

Where `value` is the name of the queue that you set up to receive messages from Siebel applications. In this example, it is `FromSiebel`.

- b** Add an argument to the property set to specify the machine that owns the queue, `MsmqQueueMachineName`.

The value of the argument should be set to the machine name. For example:

```
Name = MsmqQueueMachineName  
Value = Siebel7
```

Where `value` is the name of the machine that owns the MSMQ queue specified in the previous step. In this example, its “Siebel7.”

10 Save the workflow and run a test.

Confirm that a message was sent to the queue by using either the MSMQ Explorer or the Windows 2000 Active Directory User and Computers application. In this example, a message should be in the `FromSiebel` queue and should contain the text, “Test Message from Siebel.”

NOTE: Please review the “Export Employee (MSMQ)” workflow process in the sample database.

Sending Messages with MSMQ and Receiving Replies

The procedure below describes how to set up your system to send a message to an external system using MSMQ and receive a synchronous message back from the external system using MSMQ.

To send a message to MSMQ and receive a response

- 1 Access the Windows Computer Management tool.
 - On Windows NT, choose:
Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer
 - On Windows 2000, choose:
Start > Programs > Administrative Tools > Computer Management
- 2 Set up an MSMQ queue to receive messages from the Siebel application, and give the queue an easy to identify name, such as “fromsiebel.”
- 3 Set up another queue to send messages to the Siebel application, and give the queue an easy to identify name, such as “tosiebel.”
- 4 Choose Start > Programs > Siebel HTML Connected Client > Workflow Process Administrator > Workflow Process Designer.
- 5 Set up a workflow for sending a message out and receiving a message in response using MSMQ. Define the flow as shown in the following flowchart.



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 6 Double-click the eAI MSMQ Transport object.

In the Service Methods applet, specify the Service Name as EAI MSMQ Transport, and the Method Name as SendReceive.
- 7 Click the Input Property Set applet to create a new record with Type of “PropertySet.”

- 8 Set the Value to the string (text) that you want to send to MSMQ, such as “Test Message from Siebel.”

NOTE: You can open a text file or XML file in a text editor and copy and paste the contents into the Value field.

- 9 Click the Input Arguments applet and set four (4) parameters as follows:

- a Add an argument to the property set to specify the queue name, MsmqPhysicalQueueName.

The value of the argument should be set to the queue name. For example:

```
Name = MsmqPhysicalQueueName
Value = FromSiebel
```

Where `value` is the name of the queue that you set up to receive messages from Siebel applications. In the example above, it is “FromSiebel.”

- b Add an argument to the property set to specify the machine that owns the queue, MsmqQueueMachineName.

The value of the argument should be set to the machine name. For example:

```
Name = MsmqQueueMachineName
Value = Siebel2001
```

Where `value` is the name of the machine that owns the MSMQ queue. In this example, it is Siebel2001.

NOTE: The machine name *must* be the machine where the Siebel MSMQ Transport is running.

- c Specify the queue name for the response queue by adding an argument to the property set with a name of MsmqRespQueueMachineName.

The value of the argument should be set to the response queue name.

```
Name = MsmqRespQueueMachineName  
Value = Siebel2001A
```

Where `value` is the name of the machine that owns the MSMQ queue specified in the previous step, [Step b](#). In this example, it is Siebel2001.

- d** Specify the machine that owns the response queue by adding an argument to the property set with a name of `MsmqRespQueueName`. The value of the argument should be set to the queue name.

```
Name = MsmqRespQueueName  
Value = ToSiebel
```

Where `value` is the name of the queue that you set up to send messages to Siebel applications. In this example, it is `ToSiebel`.

- 10** Save the workflow and run a test.

The `Output Property` set should have a message in the value field. Additionally, the `EndOfData` argument in the property set should be set to `True`.

NOTE: In order to test this scenario adequately, you must have a partner application that can accept the message and return a response. The correlation ID of the response message must be set to the message ID of the message originally sent by the Siebel application.

Receiving Inbound Messages with MSMQ

With the Siebel application as the receiver (inbound messaging), you design a Workflow process that reads from the queue and converts the XML messages found there into Siebel message format. Then, the Siebel EAI Adapter updates the appropriate tables within the Siebel application as required (upsert, insert, delete, add, and so on).

NOTE: MSMQ Transport must run on the same machine where you have defined the receiving queue.

There are three methods for receiving messages for a Siebel application:

- Receive
- Receive and Execute Service (ReceiveDispatch)
- Receive, Execute, Send Response (ReceiveDispatchSend)

Receiving Messages from MSMQ

The following procedure describes how to set up your system to receive inbound messages from MSMQ.

To receive messages from MSMQ

- 1 Access the Windows Computer Management tool.
 - On Windows NT, choose:
Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer
 - On Windows 2000, choose:
Start > Programs > Administrative Tools > Computer Management
- 2 Set up an MSMQ queue to receive messages from the Siebel application.
Give the queue an easy to identify name, such as “fromsiebel.”
- 3 Set up another queue to send messages to the Siebel application.
 - a Name the queue an easy to identify name, such as “tosiebel.”
 - b Create a message in this queue: “Message from MSMQ to Siebel.”
- 4 Access the Workflow Process Designer in a Siebel application and set up a workflow for receiving a message from MSMQ. Define the flow as follows:



NOTE: For information on how to use the Workflow Designer, please refer to your *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 5 Double-click the eAI MSMQ Transport object.
- 6 In the Service Methods applet, specify the Service Name as “eAI MSMQ Transport” and the Method Name as “Receive.”
- 7 Click the Input Property Set applet and create a new record with Type of “PropertySet.”
- 8 Click the Input Arguments applet and set two (2) parameters as follows.
 - a Add an argument to the property set to specify the queue name, `MsmqPhysicalQueueName`.

The value of the argument should be set to the queue name. For example:

```
Name = MsmqPhysicalQueueName  
Value = ToSiebel
```

Where `value` is the name of the queue that you set up to receive messages from MSMQ. In this example, it is `ToSiebel`.

- b Add an argument to the property set to specify the machine that owns the queue, `MsmqQueueMachineName`.

The value of the argument should be set to the machine name. For example:

```
Name = MsmqQueueMachineName  
Value = Siebel2001
```

Where `value` is the name of the machine that owns the MSMQ queue. In this example, it is `Siebel2001`.

NOTE: MSMQ Transport must run on the same machine where you have defined the receiving queue.

- 9** Save the workflow and run a test.

The `Output Property Set` should have a message in the value field.

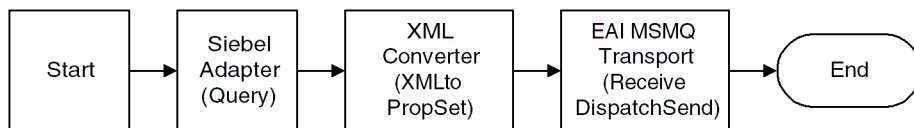
Additionally, the `EndOfData` argument in the property set should be set to `True`.

Receiving a Message from MSMQ and Acting on It

This procedure describes how to set up your system to receive an inbound message from MSMQ and perform an action based on that message within the Siebel application.

To receive and execute messages using MSMQ

- 1 Access the Windows Computer Management tool.
 - On Windows NT, choose:
Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer
 - On Windows 2000, choose:
Start > Programs > Administrative Tools > Computer Management
- 2 Set up an MSMQ queue to receive messages from the Siebel application. Give the queue an easy to identify name, such as “ToSiebel.”
- 3 Set up another queue to send messages to the Siebel application.
 - a Name the queue an easy to identify name, such as “ToSiebel.”
 - b Create a message in this queue: “Message from MSMQ to Siebel.”
- 4 Access the Workflow Process Designer in a Siebel application and set up a workflow for receiving and dispatching a message from MSMQ. Define the flow as shown in the following flowchart.



NOTE: For information on how to use the Workflow Designer, please refer to your *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 5 Double-click the eAI MSMQ Transport object. In the Service Methods applet, specify the Service Name as EAI MSMQ Transport and the Method Name as ReceiveDispatch.

- 6** Click the Input Property Set applet and create a new record with Type of PropertySet.

- 7** Click the Input Arguments applet and set four (4) parameters as follows:

- a** Add an argument to the property set to specify the queue name, MsmqPhysicalQueueName.

The value of the argument should be set to the queue name. For example:

```
Name = MsmqPhysicalQueueName  
Value = ToSiebel
```

Where `value` is the name of the queue that you set up to receive messages from Siebel application. In this example, it is “ToSiebel.”

- b** Add an argument to the property set to specify the machine that owns the queue, MsmqQueueMachineName.

The value of the argument should be set to the machine name. For example:

```
Name = MsmqQueueMachineName  
Value = Siebel2001
```

Where `value` is the name of the machine that owns the MSMQ queue. In this example, it is “Siebel2001.”

- c** Add an argument to the property set to specify the business service to receive the MSMQ message with the name, DispatchService.

```
Name = DispatchService  
Value = DispatchSend
```

Where `value` is the name of the business service to which the received message will be sent. In this example, it is “DispatchSend.”

- d** Add an argument to the property set to specify the business service method that will process the message received from MSMQ with the name, DispatchMethod.

```
Name = DispatchMethod  
Value = DispatchReceiveMethod
```

Where `Value` is the name of the business service method that will process the received MSMQ message will be sent. In this example, it is “DispatchReceiveMethod.”

8 Save the workflow and run a test.

The contents of the output property set will depend on the business service and method specified in the `DispatchService` and `DispatchMethod` arguments. Additionally, the `Output Arguments` applet should automatically populate and `EndOfData` should be set to `True`.

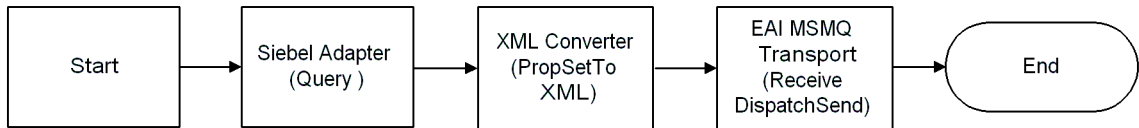
Receiving, Dispatching, and Sending MSMQ Messages

The following procedure shows you how to set up your system to receive an inbound message from MSMQ, perform an action within a Siebel application based on that message, and then send a synchronous response back to the external system.

To receive, dispatch, and send messages using MSMQ

- 1 Access the Windows Computer Management tool.
 - On Windows NT, choose:
Start > Programs > Windows NT 4.0 Option Pack >
Microsoft Message Queue > Explorer
 - On Windows 2000, choose:
Start > Programs > Administrative Tools > Computer Management
- 2 Set up an MSMQ queue to receive messages from the Siebel application.
Give the queue an easy to identify name, such as “ToSiebel.”
- 3 Set up another queue to send messages to the Siebel application.
 - a Name the queue an easy to identify name, such as “ToSiebel.”
 - b Create a message in this queue: “Message from MSMQ to Siebel.”
- 4 Choose Start > Programs > Siebel HTML Connected Client > Workflow Process Administrator > Workflow Process Designer.

- 5** Set up a workflow for receiving a message from MSMQ, dispatching data to another business service, and sending a response. Define the flow as shown in the following flowchart.



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 6** Double-click the EAI MSMQ Transport object.
- a** In the Service Methods applet, specify the Service Name as “eAI MSMQ Transport.”
 - b** Specify Method Name as “ReceiveDispatchSend.”
- 7** Click the Input Property Set applet and create a new record with Type of “PropertySet” and set the Value to the string (text) that you want to send to MSMQ, such as “Test Message from Siebel.”

NOTE: You can alternately open a text file or XML file, for example, in a text editor and copy and paste the contents into the Value field.

- 8** Click the Input Arguments applet and set six (6) parameters as follows.

- a** Add an argument to the property set to specify the queue name, MsmqPhysicalQueueName.

The value of the argument should be set to the queue name. For example:

```
Name = MsmqPhysicalQueueName
Value = ToSiebel
```

Where `value` is the name of the queue that you set up to receive messages from the Siebel application. In this example, it is “ToSiebel.”

- b** Add an argument to the property set to specify the machine that owns the queue, `MsmqQueueMachineName`.

The value of the argument should be set to the machine name. For example:

```
Name = MsmqQueueMachineName  
Value = Siebel2001
```

Where `value` is the name of the machine that owns the MSMQ queue. In this example, it is “Siebel2001.”

- c** Add an argument to the property set to specify the queue name for the response queue, `MsmqRespQueueMachineName`.

The value of the argument should be set to the response queue name.

```
Name = MsmqRespQueueMachineName  
Value = Siebel2001A
```

Where `value` is the name of the machine on which the queue that you set up to receive messages for the Siebel application is located. In this example, it is “Siebel2001A.”

- d** Add an argument to the property set to specify the machine that owns the response queue, `MsmqRespQueueName`.

The value of the argument should be set to the queue name.

```
Name = MsmqRespQueueName  
Value = FromSiebel
```

Where `value` is the name of the queue that you set up to send messages to Siebel applications. In this example, it is “FromSiebel.”

- e** Add an argument to the property set to specify the business service to receive the MSMQ message, `DispatchService`.

```
Name = DispatchService  
Value = DispatchSend
```

Where `value` is the name of the business service to which the received message will be sent. In this example, it is “DispatchSend.”

- f** Add an argument to the property set to specify the business service method that will process the message received from MSMQ, `DispatchMethod`.

```
Name = DispatchMethod  
Value = DispatchReceiveMethod
```

Where `value` is the name of the business service method that will process the received MSMQ message will be sent. In this example, it is “`DispatchReceiveMethod`.”

- 9** Save the workflow and run a test.

The contents of the output property set will depend on the business service and method specified in the `DispatchService` and `DispatchMethod` arguments. Additionally, a response message will be in the queue specified by the `MSMQRespQueueName` and `MSMQRespQueueMachineName` arguments. Use the MSMQ Explorer to check this. Finally, the `Output Arguments` applet should automatically populate and `EndOfData` should be set to “`True`.”

This chapter discusses the HTTP Transport.

About the EAI HTTP Transport

The use of the Internet protocols and technologies for business—such as HTTP, HTML, and XML—has given rise to a need to automatically send Siebel data to external sites either on the Internet, or outside the enterprise firewall to external Web sites. To meet this need, the technologies built in to Siebel eAI provide a way to send and receive messages over HTTP. Siebel EAI HTTP Transport Business Service lets you send XML messages over HTTP to a target URL (Web site). The Siebel Web Engine (SWE) serves as the transport to receive XML messages sent over the HTTP protocol to a Siebel application.

The EAI HTTP Transport business service is based on the `CSSHTTPTransService` class. You can use one of two methods with this transport. The first supports outbound messages (XML documents sent from a Siebel application to an external system) and is called simply “Send.” The second supports inbound messages (XML documents sent to a Siebel application from an external system). This method is called “Send and Receive a Response.”

Each method has its own arguments, techniques, and applications. The EAI HTTP Transports allows you to send messages across the Internet using the standard HTTP protocol. Using this transport, you can send messages to any URL. The XML document sent can then be acted upon by any Web-based application, including those written in Java, JavaScript, VBScript, or any other Web-enabled technology.

System Requirements

Using the EAI HTTP Transport requires that the following components of Siebel application be installed, configured, and operational.

- **Siebel Web Engine.** To provide the necessary HTTP listening services and invoke the requisite workflow process through a Business Service method.
- **Workflow Processes.** To accept incoming XML documents and pass them through an Integration Object into the Business Object to update Siebel data.
- **Business Services.** To execute the necessary actions.

Selecting the Appropriate Business Service for HTTP

The business service you need to initialize to process a given XML document received from an external system using the EAI HTTP Transport depends on the processing you need to do on the data. The way to approach this is to accept the output of the EAI HTTP transport and store it as a process property that you define, and process it later in the workflow based on the format of the data.

For example, you could pass the string into a custom Business Service that you build to parse the input, query some data in a Siebel application based on the data, and then update the appropriate field in the Siebel application. If the data is formatted as a SiebelMessage, you could use the EAI XML Converter business service with the XMLtoPropSet method to build a property set to pass to Siebel Adapter for further processing.

Using POST and GET

The HTTP protocol supports GET and POST methods. You might be familiar with these methods if you have ever built a Web-based CGI form.

The EAI HTTP Transport imposes certain restrictions on your use of transport features when using POST or GET method. [Table 6](#) identifies restrictions on these HTTP methods.

Table 6. Restrictions on GET and POST Methods with HTTP Transport

Method	Restriction
GET	The HTTP Body has no significance when using GET. During a GET process, only the universal resource locator (URL) is used for the request.
POST	The HTTP Body is relevant only when using POST.
	The HTTP Body is encoded with a default mechanism used to encode URLs.
	The HTTP Content-Type <code>application/xxx-form-urlencoded</code> is the default content type used for request bodies. The content is sent as it is without any special content encoding, such as Base64.

HTTP Transport Named Subsystems

The EAI HTTP Transport, like every other Siebel transport, reads required parameters from a *named subsystem* instead of the configuration file (*.cfg). The eai.cfg file entries should list the external service name and the name of the named subsystem to be used. For example:

```
SiebelQuery = SiebelQueryDispatch
```

There is no “[Properties]” section for SiebelQueryDispatch in the .cfg file. The name will be used to look up the named subsystem list and dispatch accordingly. Use named subsystems for property specification. Predefined named subsystems have been created for you already, such as:

- SiebelQueryDispatch
- SiebelExecuteDispatch
- SiebelUpsertDispatch

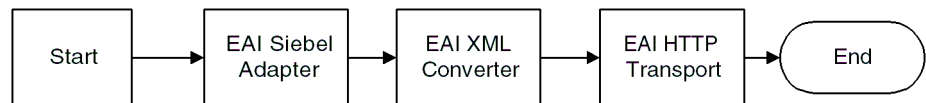
NOTE: You previously specified properties by means of *.cfg file entries. You can continue to do so, but you should switch over to using named subsystems because *.cfg file entries will not be supported in future releases. Only named subsystems will work for new functionality such as Dispatch Service and Character Set Conversions. You can create additional named subsystems as needed using Siebel Server Manager.

For a discussion of named subsystems for Siebel eAI, see [Chapter 1, “EAI Transports and Interfaces Overview.”](#) For more information on named subsystems, see *Siebel Server Administration Guide, MidMarket Edition*.

General Information on How to Send a Message

The following steps demonstrate how to send information from a Siebel application to another Web-based application using the EAI HTTP Transport.

- 1** Create an integration object in Siebel Tools based on a given business object.
- 2** Refine the integration object created in [Step 1](#) to specify just those business components and fields that you want to exchange with the external application.
- 3** Create a workflow using the Workflow Designer to send this information using the EAI HTTP Transport by including the following steps in the workflow, as shown in the following flowchart.



- 4** For the first step after Start, select EAI Siebel Adapter business service with the Query method. Enter these input arguments:
 - The Integration Object created in [Step 1](#)
 - The RowId passed from the script behind the button (or from the workflow policy in [Step 7](#))
- 5** Next, convert the output from the EAI Siebel Adapter to XML using the EAI XML Converter.
- 6** For the last step before End, take the output from [Step 5](#) and pass it to EAI HTTP Transport business service with the Send or SendReceive method.
- 7** Specify how this workflow will be invoked using one of the following methods:
 - Configure the RunTime Events to trigger the workflow.
 - Create a button on the appropriate view in the Siebel application to call this workflow process.
 - Use workflow policies on the opportunity business object to trigger the workflow process.

Using the EAI HTTP Transport for Inbound Integration

The EAI HTTP Transport uses the Siebel Web Engine (SWE) to provide inbound messaging from an application that uses HTTP. To use the EAI HTTP Transport for inbound integration, you must perform certain tasks that are not required when using the EAI HTTP Transport for outbound integration. First, you must be running the Siebel Web Engine (SWE) in order to use the EAI HTTP Transport. In turn, SWE requires that the Siebel Web Server is installed, configured, and up and running.

NOTE: Issue the `http:// < Web_Server_Name > URL` on any machine that already has connectivity to the Web Server machine to check the connectivity between the URL (for HTTP Transport) issuing machine and the SWE machine. This URL brings up the Home page of the Web Server machine confirming the connectivity between the SWE machine and the URL issuing machines.

Specifying HTTP Parameters for Inbound Integration

The EAI HTTP Transport is built into SWE. To use it, you first need to set certain configuration parameters for the virtual directory on the Web server. Your Siebel application installation includes a configuration file called `eapps.cfg` in the `\bin` subdirectory of your install directory. This file is on the Web server side of your configuration, as opposed to your Siebel application server installation. You should review the configuration file to make sure that the parameters are set properly. Use named subsystems to dispatch to a workflow.

To configure the Siebel Web Engine to run the EAI HTTP Transport for inbound Integration

- 1 Open your `eapps.cfg` file in a text editor.
- 2 Look for the section `[/eai]`.
- 3 Add the `EnableExtServiceOnly` configuration parameter or set it as follows, if it already exists, to enable the HTTP inbound transport.

```
[/eai]
ConnectString = <Connect String>
EnableExtServiceOnly = TRUE
```

You need to set the `ConnectionString` parameter for the virtual directory. The syntax for the `ConnectionString` is:

```
ConnectionString =
siebel[.transport][.encryption][.compression]://<gateway
server {host|VIP}{:port}>/<enterprise name>/<object manager
name>[/<server name>]
```

where:

- `transport` = `tcpip` | `http`.
- `Encryption` = `none` | `mscrypto`.
- `Compression` = `none` | `zlib`.
- The default port for TCP/IP is 2320, and for HTTP it is 80.

The following example shows the connect string using TCP/IP, with no encryption, no compression, and the server name and default port. In addition, you need to point to the Siebel object manager specific to your client. In the example connect string, the client is Siebel eSales MidMarket Edition, and the Siebel object manager is called *eSalesObjMgr*. Items shown in angle brackets—such as `<gatewayhost>`—should be replaced with a specific name.

```
ConnectionString = siebel.TCPIP.none.none://<gatewayhost>:2320/
siebel/eSalesObjMgr/<Siebel Application Server name>
```

- 4 Save and close the configuration file.

Setting Configuration Parameters for Siebel Application Server

You must also set certain configuration parameters for whatever Siebel application server you are using. The server component you are running must be a Client Application Manager component. Set this in the configuration file for the server component of your choice, or use named subsystems.

Calling EAI HTTP Transport Over a Network (HTTP Protocol)

The EAI HTTP transport can be used in two modes:

- Session mode
- Sessionless mode

The following sections explain the usage of these two modes.

Session Mode

This mode uses HTTP Session Cookie to retain the session information between the HTTP requests.

To view the session mode when a sequence of calls is supported from an HTTP application into the EAI HTTP Transport

- 1** Log in to the Siebel application. If successful, an HTTP session cookie gets returned in HTTP set-cookie header.
- 2** Submit one or more requests.

Each request is intended as a call to a Siebel Business Service. Requests must contain the session cookie from [Step 1](#) in the HTTP cookie header.

- 3** Log off. The request must contain the session cookie from [Step 1](#) in the HTTP Cookie header. The cookie refers to the session to be closed.

The Session cookie is passed to the caller after a successful login request as in [Step 1](#). The caller then should use that cookie for subsequent data requests in [Step 2](#) and the log off request in [Step 3](#).

Login Examples

HTTP protocol requests can be represented as URLs for HTTP GET, and as a combination of URL and request body for HTTP POST. The following sections explain in detail how each of the session mode calls is configured.

Login HTTP Request Example 1

In this example, if the call completes successfully, it will return a session cookie.

Using HTTP GET

```
URL    = http://<webserver>/<path>/
start.swe?SWEExtSource=<source>&SWEExtCmd=ExecuteLogin&UserName=
<username>&Password=<password>
```

Using HTTP POST

```
URL    = http://<webserver>/<path>/start.swe

HTTP Body =

SWEExtSource=<source>&SWEExtCmd=ExecuteLogin&UserName=<username>
&Password=<password>
```

[Table 7](#) describes each of the Login HTTP Request variables for session mode.

Table 7. Session Mode Variables

Variable	Description
webserver	URL of the Web server that has Siebel Web Engine installed, such as www.myserver.com.
path	Virtual path on the server referring to specific SWE configuration. This value should be eai for each inbound HTTP request.
source	If you are not using named subsystems, this is the name of the Business Service Source as specified in [HTTP Services] section in the .cfg file that describes the Business Service call.
username	Siebel user name for the Siebel Object Manager login.
password	Password for the login user name above.

Example Login URL

```
http://www.myserver.com/eai/
start.swe?SWEExtSource=SiebelQuery&SWEExtCmd=ExecuteLogin&UserNa
me=user1&Password=login123
```

Login HTTP Request Example 2

In this example, for the call to complete successfully, it must include the session cookie from the login.

Using HTTP GET

URL = http://<webserver>/<path>/start.swe?SWEExtData=<data text>

Using HTTP POST

URL = http://<webserver>/<path>/start.swe

HTTP Body = <data text>

Table 8 describes the HTTP GET and POST variables that are required for a request.

Table 8. HTTP GET and POST Variable Required for Request

Variable	Description
data text	Business Service Input data. Most of the time, this is the text of an XML document that on the server side will be converted to a PropertySet and passed to the Business Service.

Example Request URL

```
http://www.myserver.com/eai/start.swe?SWEExtData=<?xml
version="1.0" encoding="UTF-8"?><SiebelMessage MessageId=" "
MessageType="Integration Object" IntObjectName="Sample Account">

  <ListOfSampleAccount>
    <Account>
      <Name>A. K. Parker Distribution</Name>
      <ListOfContact>
        <Contact>
          <FirstName>Stan</FirstName>
          <LastName>Graner</LastName>
        </Contact>
      </ListOfContact>
    </Account>
  </ListOfSampleAccount>
</SiebelMessage>
```

NOTE: To use this URL you need to change the WebServer address “www.myserver.com” to the actual server URL you will be using. Data that is sent as part of the URL should be UTF-8 encoded before being URL-encoded. POST requests can send the data without URL encoding and should include the Content-Type HTTP header. The Content-Type should specify the charset of the incoming data as in `Content-Type=text/xml; charset="UTF-8"`.

Logoff HTTP Request

This request must include the session cookie from Login:

Using HTTP GET

URL = `http://<webserver>/<path>/start.swe?SWEExtCmd=Logoff`

NOTE: HTTP GET should always be used for the Logoff HTTP Request.

Example Logoff URL

`http://www.myserver.com/eai/start.swe?SWEExtCmd=Logoff`

Sessionless Mode

Using the EAI HTTP Transport in sessionless mode allows you to use one URL to perform Login, Request, and Logoff in a single HTTP request. This mode does not use session cookies since there is no login session between the HTTP requests. The disadvantage of this mode is the overhead incurred by the Siebel object manager needing to log in with every request.

Example

In this example, the URL describes the parameters for the HTTP Inbound Transport call over HTTP.

Using HTTP GET

URL = `http://<webserver>/<path>/start.swe?SWEExtSource=<source>&SWEExtCmd=Execute&UserName=<username>&Password=<password>&SWEExtData=<data text>`

NOTE: Unlike session mode, the SWEExtCmd is Execute, not ExecuteLogin.

Using HTTP POST

```
URL = http://<webserver>/<path>/start.swe
```

```
HTTP Body =  
SWEExtSource=<source>&SWEExtCmd=Execute&UserName=<username>&  
Password=<password>&SWEExtData=<data text>
```

NOTE: When using the sessionless mode with the POST method, the XML data text must be URL-encoded to prevent any errors.

Table 9 describes each of the variables for sessionless mode.

Table 9. Sessionless Mode Variables

Variable	Description
webserver	URL of the Web server that has Siebel Web Engine installed, such as www.myserver.com.
path	Virtual path on the server referring to specific SWE configuration. By default, this value should be “eai.”
source	If you are not using named subsystems, this is the name of the Business Service Source as specified in [HTTP Services] section in the .cfg file that describes the Business Service call.
username	Siebel user name for the siebel object manager login.
password	Password for the login user name.
data text	Business Service Input data. Most of the time, this is the text of an XML document that on the server side will be converted to a PropertySet and passed to the business service.

Example Sessionless Mode URL

NOTE: This sample URL should be entered as a single line of text. The URL is presented here on separate lines for clarity.

```
http://www.myserver.com/eai/start.swe?SWEEExtSource=SiebelQuery&
SWEEExtCmd=Execute&UserName=user1&Password=login123
&SWEEExtData=<?xml version="1.0" encoding="UTF-8"?><SiebelMessage
MessageId=" " MessageType="Integration Object" IntObjectName="Sample
Account">
```

```
    <ListOfSampleAccount>
      <Account>
        <Name>A. K. Parker Distribution</Name>
        <ListOfContact>
          <Contact>
            <FirstName>Stan</FirstName>
            <LastName>Graner</LastName>
          </Contact>
        </ListOfContact>
      </Account>
    </ListOfSampleAccount>
  </SiebelMessage>
```

To use this URL you need to:

- Change the WebServer address, `www.myserver.com`, to your actual Web server URL.
- Verify that the `SWEEExtSource` argument has a corresponding section in your `eai.cfg` file in the `[HTTP Services]` section.
- Change the Username and Password arguments to a valid system user, for example “SADMIN/SADMIN.”

Using the HTTP Transport for Inbound Messages

To use the EAI Transport, you complete two steps, as explained below. In the steps, the scenario assumes incoming XML. Your business requirements will dictate if and how you need to adapt these steps to fit your needs.

Checklist

- ☐ Set up the business service for use in the workflow.
For details, see [“To set up the business service.”](#)
- ☐ Create the workflow.
For details, see [“To create the new workflow process” on page 71.](#)

To set up the business service

- 1 Login to Siebel Tools connecting to the server database.
- 2 Find the Business Service named “Workflow Process Manager.”
- 3 Copy this record and rename the copy “EAITEST.”
- 4 Access the Business Service User Props window and add a new record
 - a Enter “Process Name” in the Name column.
 - b Enter “EAITEST” in the Value column, as shown in the following illustration.

	W	Name	Changed	Project	Cache	Class
>		EAITEST	✓	Account	✓	CSSwfEngine
		WI Web Proxy Service		WI - Web Integration		CSSWIService
		Web Collab Service		Web Collaboration	✓	CSSWebCollabService
		Web Engine HTTP TXN		SWE		CSSServiceSweHttpTxn
		Web Engine Interface		SWE	✓	CSSServiceSWEIface

Business Service User Props

	W	Name	Changed	Value	Inactive	Comments
>		Process Name	✓	EAITEST		

- 5 Compile a new *.srf file and copy it to the Server\Objct directory.
- 6 Re-start Siebel Server.

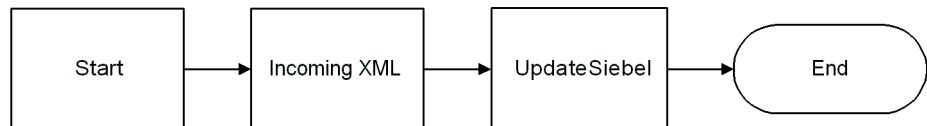
- 7** Verify that the eAI object manager has started.

To create the new workflow process

- 1** Log into Siebel client as an administrator connected to the server.
- 2** Navigate to Siebel Workflow Administration > Workflow Processes.
- 3** Create a new workflow process and give it a unique name, such as “EAITEST.”
- 4** Select the Process Properties tab and add the three properties:

Process Property	Values	Notes
IncomingXML	<ul style="list-style-type: none"> ■ Set Data Type to “Binary” ■ Set Default String to “ < Value > ” 	By creating the IncomingXML Process Property, anything that is sent as data will be placed in this variable. This allows you to then perform a given action on that data. If the POST method was used, the data sent in the Body will be stored in this property. If the GET method was used, the data sent in the URL will be stored in this property.
Account Message	<ul style="list-style-type: none"> ■ Set Data Type to “Hierarchy” 	
< Value >	<ul style="list-style-type: none"> ■ Set Data Type to “Binary” 	

- 5** Access the Workflow Process Designer.
- 6** Add two Business Services and a Start and an End, as shown in the following flowchart.



- 7** Double-click the first Business Service. Edit it as shown in the following table.

Item	Value
Name	Set to “Incoming XML”
Business Service	Set to “EAI XML Converter”
Method	Set to “XML Document to Integration Object Hierarchy”

- a** Create a new Input Argument. Edit it as shown in the following table.

Item	Value
Input Argument	Set to “XML Document”
Type	Set to “Process Property”
Property Name	Set to “Incoming XML”

- b** Create a new Output Argument. Edit it as shown in the following table.

Item	Value
Property Name	Set to “Account message”

- 8** Select the second Business Service. Edit it as shown in the following table.

Item	Value
Business Service	Set to “EAI Siebel Adapter”
Method	Set to “Insert” or “Update”

- a** Create an Input Argument. Edit it as shown in the following table.

Item	Value
Input Argument	Set to "Siebel Message"
Type	Set to "Process Property"
Property Name	Set to "Account Message"

- b** Create an Output Argument. Edit it as shown in the following table.

Item	Value
Value	Set to " < h1 > Update Completed < /h1 > "
Type	Set to "Literal"
Property Name	Set to " < Value > "

NOTE: The HTTP response for inbound requests is determined by looking at the <Value> portion of the output property set. HTTP response headers can be set by setting properties on the output property set.

- 9** Save your work.

HTTP Transport Business Service Error Handling

A business service that is called by the EAI HTTP Transport might return an error when standard HTTP headers are used to send error information back to the caller. Each of the headers has a sequence number at the end to support the return of multiple errors. The text of each error message is captured in the Siebel-Error-Message header, and the Siebel error symbol is set in the Siebel-Error-Symbol header as shown below.

```
Siebel-Error-Message-1: Error: error message text
Siebel-Error-Symbol-1: ERR_SYMBOL
...
Siebel-Error-Message-n:
Siebel-Error-Symbol-n:
```

Inbound HTTP will also return HTTP Error 500 (Internal Server Error) to indicate that there was an error from a business service. The error headers should then be examined for additional error information.

NOTE: To troubleshoot an Inbound HTTP request, run the Siebel Workflow Process Simulator or Business Service Simulator.

How to Use Messages Returned from an External System

To send and receive messages through HTTP, you need to set up two workflows.

- **Workflow 1.** This workflow sends the message using the EAI HTTP Transport Business Service with the “POST” and the “SendReceive” methods.
- **Workflow 2.** This workflow receives the message back through HTTP and places the message in a text string, as defined by a workflow step (Output argument = Message Text).

When setting up Workflow 2, first verify that Workflow 1 has correctly stored the Message Text output argument into a Process Property. Then, you use that Process Property as an Input Argument in a workflow step for Workflow 2.

Using the EAI HTTP Transport for Outbound Messages

This section explains how to use Siebel Tools and Siebel applications to set up the HTTP Transport to process and send outbound XML documents. When you want to send XML messages based on Siebel Integration Objects to an external system across Internet-support protocols, you use the EAI HTTP Transport business service.

Specifying HTTP Transport Parameters for Outbound Integration

You can specify the parameters that control the behavior of transports using four different mechanisms, in the following order of precedence:

- [Specifying Parameters as Business Service User Properties on page 75.](#)
- [Specifying Parameters as Subsystem Parameters on page 76.](#)
- [Parameters as Run-Time Properties on page 77.](#)
- [Parameters in Parameter Templates on page 77.](#)

Specifying Parameters as Business Service User Properties

You specify parameters as business service user properties in Siebel Tools. These parameters go into effect when you have compiled the `.srf` file. When using this method, keep the following in mind:

- These parameters stay in effect as long as you continue to use the same `.srf` file and do not recompile it with a newer specification for the business service parameters.
- If you define the same parameter as a subsystem parameter or as a run-time property, the subsystem parameter or run-time property will override any values you have defined in Siebel Tools and compiled into the `.srf` file.

Specifying Parameters as Subsystem Parameters

You can specify parameters as on either the client side or the server side, depending on whether you use the client or server version.

To specify the parameters on the client side

- 1 Using a text editor, open a configuration (.cfg) file, such as siebel.cfg.
- 2 Add a section in the configuration file named: [HTTPSubSys]
- 3 Add Name = Value pairs for each parameter, as follows:

```
[HTTPSubSys]
HTTPRequestURLTemplate = http://my.com/
app.exe?User=$user$&Pass=$password$
```

NOTE: The preceding entry sets the HTTPRequestURLTemplate to a specific location and passes parameters \$user\$ and \$password\$. The user and password parameters must be set as either user properties or as run-time parameters.

- 4 Save the file and exit the text editor.

To specify the parameters on the server side

- 1 Start the Siebel client and navigate to Server Administration > Enterprise Configuration.
- 2 Navigate to the Enterprise Parameters form.
- 3 Add a new key and value pair that specifies the parameter and its corresponding value.

If the parameter already exists, select it to make it the active parameter, then type the value you want to use in the Value field.

- 4 Restart the Siebel Server.

Any parameter values you set using a configuration file overrides parameters with the same name you might have specified as user properties in your Siebel .srf file. The name and password you specify should already exist in your database.

Parameters as Run-Time Properties

You specify HTTP parameters as run-time properties by passing them as values in an input property set to the HTTP Transport business service. You can pass the values to the business service by way of a workflow or through a program that calls the HTTP Transport business service directly.

NOTE: Any parameters specified in an input property set will override parameters with the same name that have been specified in either the `.srf` file or in your subsystem parameters.

Parameters in Parameter Templates

Parameter templates allow you more flexibility in specifying parameters. You can use variables to specify certain elements of a given parameter value. The following example shows how to specify a variable for a login password, rather than hard-coding a password into the parameter.

```
HTTPLoginURLTemplate = http://www.srvr.com/  
login?Username=ronw&Password=$PWD$  
  
PWD = 421ax7
```

The business service, EAI HTTP Transport in this case, receives the parameter template. The token, shown above as “\$PWD\$,” indicates that the business service should look for a parameter called “PWD” from a user property or run-time parameter. Dollar signs (\$) delimit the token in the template definition. The token specifies the actual password variable. The token is case-sensitive—PwD is different from PWD or pwd.

The token must be defined as either a business service user property or as a run-time parameter in the input property set. For example, you could specify the HTTPLoginURLTemplate as a user property of the business service, and *username* and *password* as run-time properties. Any logins that specify the template will always use the same template, but different users can specify unique user names and passwords at run time.

Examples Using HTTP Request

This section provides you with a couple of examples of using the HTTP Transport in two modes: Session and Sessionless. This is to help in understanding how to use the HTTP Transport in your business.

HTTP Request in Session Mode

The session mode allows control over login sessions. In this mode you log in first and open a session. Any message can be exchanged without having to log in again until you explicitly log off.

The following example shows parameters for Login, Request, and Logoff in a session mode HTTP request. Session cookies are required in a case such as this.

NOTE: You enter each of the following URLs as a continuous line of code.

The following URL logs in to a server with passed parameters for username and password:

```
HTTPLoginURLTemplate = "http://$ServerPath$/  
start.swe?SWEExtSource=$Source$&SWEExtCmd=ExecuteLogin&User  
Name=$Username$&Password=$Password$"
```

- The following URL passes a query string as the SWEExtData value along with the GET request:

```
HTTPRequestURLTemplate = "http://$ServerPath$/  
start.swe?SWEExtData=<Prop>somedata</Prop>  
HTTPRequestMethod='GET' "
```

- The following URL logs off from the server:

```
HTTPLogoffURLTemplate = "http://$ServerPath$/  
start.swe?SWEExtCmd=Logoff"
```

```
ServerPath = "siebell/eai"
```

```
Username = "pdavis"
```

```
Password = "1234abcd"
```

```
Source = "testdoc"
```

In the preceding example, the `ServerPath` variable value of “siebel1/eai” is substituted for the token `$ServerPath$`. The `Source` variable value of “testdoc” is substituted for the `$Source$` token, the `Username` variable value of “pdavis” for the token `$Username$`, and the `Password` variable value of “1234abcd” for the `$Password$` token.

Any XML document represented by the entry for `SWEEExtData` can be put into the body. This would change the sample code so that the `HTTPRequestURLTemplate` would read as:

```
HTTPRequestURLTemplate = "http://$ServerPath$/start.swe?"
```

Secure Request in Sessionless Mode

The following example includes a `RequestMethod`, a `Request`, and a `Login` for a sessionless mode request. In this example, the request is simply passed to the secure server using the `POST` command. Unlike the `Session Mode` example, this request sends data in the body of the request. This request does not require cookies.

```
RequestMethod = "POST"
```

```
HTTPRequestURLTemplate = "https://accounts.mypartner.com/server/login.asp"
```

```
HTTPRequestBodyTemplate =  
"Acct=ABCIntl&User=$Username$&pwd=$Password$"
```

```
Username = "acctuser"
```

```
Password = "123456789abcdefg"
```

EAI HTTP Transport Method Arguments

EAI HTTP Transport methods take the arguments described in [Table 10](#).

- A box (■) in the “S” column means the parameter is required for session mode.
- A box in the “SL” column means the parameter is required for sessionless mode.

Table 10. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
<Value>	User-Defined Message Text			Input and Output data passed as a string. This is the value stored in the Value field of the property set, either input or output. If you specify the HTTPRequestBodyTemplate, the <Value> parameter is ignored and the HTTPRequestBodyTemplate parameter is used instead.
HTTPRequestURLTemplate	Request URL Template	■	■	Template for the request URL, which is the address to which the data is sent or from which a response is requested.
HTTPRequestMethod	Request Method	■	■	HTTP method to use with the data request.
HTTPRequestBodyTemplate	Request Body Template			HTTP Body to use with the POST method. This overrides any body specified in the Value field of the input property set.
HTTPLoginURLTemplate	Login URL Template	■		Template for the URL used for the login operation. This operation is separate from the request operation and assumes communication mode is session mode. If there is a separate login, one or more request and response messages are expected.
HTTPLoginMethod	Login Method	■		HTTP method to be used for logging in. If no Login Method is specified, this parameter defaults to the HTTPRequestMethod value.

Table 10. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPLoginBodyTemplate	Login Body Template	■		Specifies the HTTP request body that should be used when HTTPLoginURLMethod is POST. By putting login information into the HTTP body (as opposed to putting it into the URL) for sending, this method provides stronger security than sending the login information in the URL. Normally, the login parameters in a login query are specified in the body of the request that uses the POST method. Required for session mode only if the HTTPLoginMethod parameter is set to POST.
HTTPLogoffURLTemplate	Log Off URL Template	■		Template for the URL that is used for the logoff operation. This operation is separate from the request operation and assumes that the mode of communication is session mode. If set, the logoff operation will be completed. Otherwise, logoff is skipped. The purpose of the logoff operation is to end a session that was started with the corresponding login.
HTTPLogoffMethod	Log Off Method			HTTP method to be used for logging off. Defaults to the HTTPLoginMethod.
HTTPAccept	HTTP Accept			The explicit value for the Accept: header to override the default. Specifies the MIME types accepted by the sender. The default value is <i>text/*</i> .
HTTPContentType	HTTP Content Type			The explicit value for the Content-Type: header to override the default. Specifies the type of data sent in the body of the request. The default value is <i>application/xxx-form-urlencoded</i> .
HTTPUserAgent	HTTP User Agent			The explicit value for the User-Agent: header to override the default. Specifies the name/version of the client program. The default value is <i>Mozilla/4.0</i> .
HTTPMaxIdleSeconds	Max Idle Seconds			Maximum number of seconds to allow connections to be idle. After the elapsed max idle time, the connection is invalidated and restarted.

Table 10. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPAllowCaching	Allow Caching			<p>Set to N by default. By default, the responses for specific URL addresses are not cached by the HTTP transport. Set this flag to Y to enable caching.</p> <p>Note that this can lead to undesirable side effects, as old data from prior requests can be exposed from the cache buffer.</p>
HTTPAllowPersistentCookies	Allow Persistent Cookies			<p>Set to N by default.</p> <p>A session cookie is used to tie requests and logoff operations to the user session started at the login, when communicating with any session-cookie-based system. Leaving this flag set to N leaves the persistence of cookies under the control of the HTTP transport, which is the default behavior.</p> <p>All session cookies persist in memory only as long as the current session. Session cookies are not written to disk.</p> <p>If you want to use persistent cookies—that is, if persistence between logins is required and you want cookies written to disk and then set the parameter to Y.</p>
HTTPIsSecureConn	Is Secure Connection			<p>This parameter is set to N by default. Its behavior is that the security mode defaults to whatever the URL specifies, either HTTP or HTTPS.</p> <p>Setting this parameter to Y enables the Secure flag for SSL communications; thus, it forces the use of secure mode.</p> <p>If you choose to use SSL encryption, you must establish valid certificates and SSL capabilities on both the client and server.</p> <p>Using the HTTPS: designation in an URL, by default, enables the Secure flag. Using the HTTP: designation in an URL, by default, specifies “clear text” unencrypted communications.</p>

Table 10. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPNoAutoRedirect	No Auto Redirect			This parameter is set to N by default. By default, auto-redirect is enabled. Setting this parameter to Y disables auto-redirection of messages to other URLs.
HTTPSleepTime	Sleep Time			The timeout interval on login, send, and logoff requests in milliseconds. The default for the EAI HTTP Transport is 120000 milliseconds.
HTTPImplicitCharsetDetection	Implicit Character Set Detection			Implicit Character Set Detection for incoming data. The default value for this is FALSE, and should not be set to TRUE for self-describing documents like XML. If set to TRUE, this overrides the CharSetConversion parameter.

The Siebel OLE DB Provider conforms to Microsoft's OLE DB data access specifications and provides a unidirectional method for retrieving data from the Siebel database and viewing it in any supported OLE DB-enabled application.

Microsoft OLE DB

Microsoft's OLE DB provides applications, compilers, and other database components access to Microsoft and third-party data stores. OLE DB defines interfaces for accessing and manipulating every type of data. These interfaces are used both by data consuming applications and data providing applications.

An OLE DB Consumer is any application that can access OLE DB Providers and display the data as an embedded objects, which retain their original format and links to the application that created them. OLE DB consumers are also known as *external OLE DB-enabled applications*.

Siebel OLE DB Provider

Siebel OLE DB Provider is a set of interfaces that allow you to access the Siebel business object layer. Using Siebel OLE DB Provider technology, you can build *ad hoc* queries, use third-party business analysis tools, and build Web applications that access business-critical information from Siebel Systems. Access to this information is essential for providing excellent customer service and making intelligent business decisions.

OLE DB consumers can access data stored in the Siebel database by referring to Siebel objects such as Contact or Account, without having to perform mapping tasks between the Siebel Data Model and the external application. Siebel OLE DB Provider is integrated with Siebel Tools, allowing management and configuration of the Siebel Business Components that are exposed to the client application as OLE DB tables.

Most third-party business intelligence tools provide powerful Web-based *ad hoc* query tools that let you access, navigate, and explore relational data to make key business decisions in real time. This insight helps companies improve target marketing efforts and forge closer, more responsive, relationships with customers.

For example, your job might require forecasting sales opportunities. Using a third-party query and reporting tool such as Seagate's Crystal Reports, you could retrieve opportunities from Siebel Systems' operational data store using Siebel OLE DB Provider. You could even create a heterogeneous query across multiple data stores to get the level of detail required to make better eBusiness decisions.

As another example, you could determine the success of your Web marketing campaign by evaluating the number of hits your Web site received last month, last week—even today—and contrast that information with the number of products (or services) purchased.

You might also have a portal where customers can look up outstanding orders, and service requests. Using Siebel OLE DB Provider invoked from an ASP file on a Windows Server, the Siebel System Administrator could expose the orders and service requests, and the Web Developer could create a Web-based query which would:

- Gather information on orders and service requests from the Siebel application
- Populate a customized view of outstanding information related to the customer

Software Architecture

The Siebel OLE DB Provider is a read-only object that exposes Siebel business components as virtual OLE DB tables. You can connect to the Siebel OLE DB Provider by way of external OLE DB-enabled applications—for example, from OLE DB Consumers including Microsoft Excel and Microsoft Access—and view data dynamically, as it is queried from your Siebel database, within pivot tables, charts, or other appropriate data controls.

Using Siebel Tools, you define Siebel OLE DB rowsets to be queried against. These rowsets are an extension to the integration objects available within Siebel Tools. Siebel OLE DB Provider must be installed on the system that executes the queries. This does not mean that a Siebel client or the Enterprise Server must be running on this same system, but that they be accessible on the network. You can use Siebel OLE DB Provider to use your Siebel data in two ways: either through the use of existing OLE DB consumers, such as Microsoft Excel and Access, or through applications and scripts you write. With either method, Siebel OLE DB Provider works in the following scenarios:

- **Windows Client.** You can query Siebel OLE DB rowsets using third party business intelligence tools such as Microsoft Office or Cognos.

NOTE: Siebel OLE DB Provider must be installed on the Windows Client to gain access to the defined OLE DB rowsets.

- **Windows Server.** Using a third-party business intelligence tool such as Seagate's Crystal Reports or Cognos allows for the distribution of Siebel OLE DB rowsets through their query and reporting interface. You can add predefined queries or reports to the Siebel client Reports Menu using the Siebel Tools Reports Administrator.

NOTE: Siebel OLE DB Provider must be installed on the IIS system in order to gain access to the defined OLE DB rowsets.

- **Siebel Web Client, Mobile Web Client, and Dedicated Web Client.** As noted previously, using a third-party business intelligence tool installed on Windows Server is used to output to these clients.

Figure 1 illustrates the architecture of the Siebel OLE DB Provider.

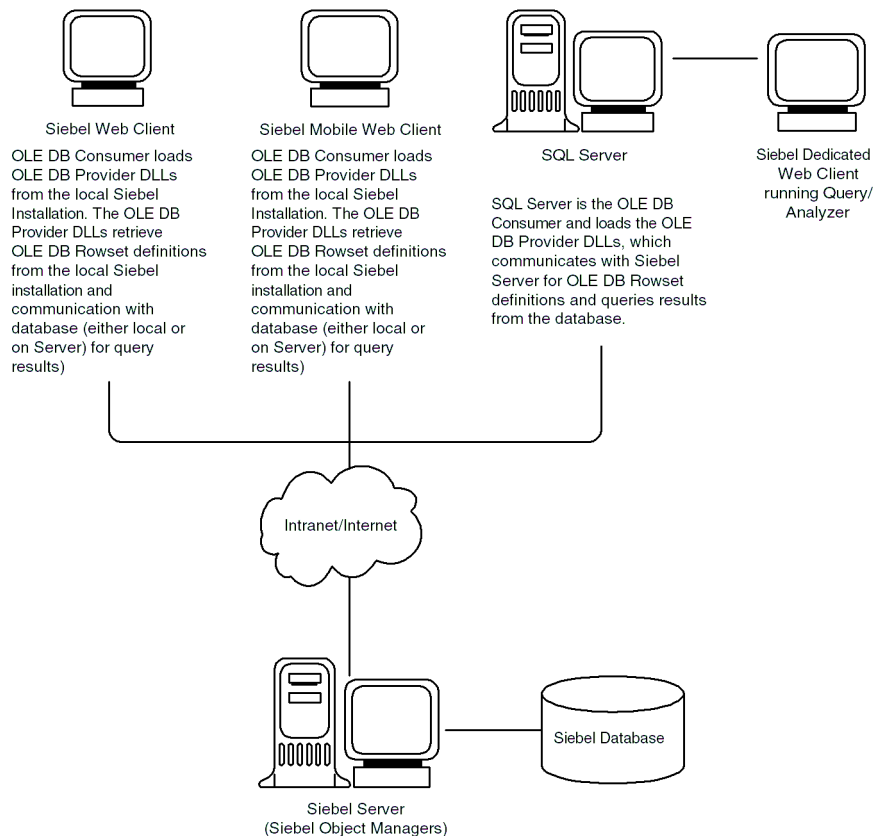


Figure 1. OLE DB Providers and Consumers of Siebel Data

Understanding Siebel OLE DB Provider

You can install the Siebel OLE DB Provider standalone from the Siebel Server CD ROM. By selecting the custom EAI installation, you can install the OLE DB Provider as a standalone client. Siebel OLE DB Provider must be installed on the same system where the OLE DB Consumer has been installed.

The Siebel OLE DB Provider can be queried from a number of sources. To use Siebel OLE DB Provider with Microsoft SQL, Microsoft SQL and the full version of Microsoft SQL Server must be installed and operational. To create an Active Server Page (ASP) application, the Siebel OLE DB Provider and Microsoft IIS must be installed and operational. The required base-level operating systems validated for the use of Siebel Systems products contain the necessary foundation versions of OLE DB and ADO to support the Siebel OLE DB Provider. See the *Siebel System Requirements and Supported Platforms* for further information.

- When accessing the Siebel OLE DB Provider from ASP pages (this includes access from the client application Internet Explorer or other Web browser), IIS is the OLE DB Consumer.
- When accessing the Siebel OLE DB Provider from SQL Server (this includes access from the client application Query Analyzer), SQL Server is the OLE DB Consumer.
- When accessing the Siebel OLE DB Provider from Excel or Access, Excel or Access is the OLE DB Consumer.

The Siebel OLE DB Provider is being installed as a default interface within the Siebel Mobile Client and the Siebel Server applications. You can also install this on different systems using the custom EAI install from Siebel Server. The Siebel OLE DB Provider supports the following OLE DB and ActiveX Data Object (ADO) foundation versions:

- Microsoft OLE DB
- Microsoft ADO

What's in This Section

This section covers the following topics:

- [Siebel OLE DB Provider Configuration for Testing on page 90](#)
- [Multiple Language Considerations on page 93](#)
- [Primary and Foreign Key Relationships on page 93](#)
- [Using the Windows Event Viewer with Siebel OLE DB Provider on page 94](#)
- [Viewing OLE DB Information on page 95](#)

Siebel OLE DB Provider Configuration for Testing

You configure and test Siebel OLE DB Provider using the siebel.udl file that is installed in the <install_directory>\bin\<language> directory. The siebel.udl file appears this way:

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=SiebelOLEDB.Provider.1;Persist Security Info=False;
```

[Table 11](#) lists the necessary connection properties.

Table 11. Siebel OLE DB Provider Connection Properties

Property	Description
Data Source Name	The connection string as explained in <i>Siebel Tools Online Help, MidMarket Edition</i> . In the .cfg file, make sure the DataSource property in the [Siebel] section is set for the local client using Local, Sample, or Server. For example, set to “Sample” for the sample database.
User ID	Your Siebel user ID.
Password	Your Siebel password.

When you have entered these values, test the connection to the data source by clicking the Test Connection button on the Microsoft Data Link Properties dialog box. If the connection is valid, you will receive a confirmation. If you receive a successful connection confirmation, you can use the same data source name with any OLE DB-enabled application to connect to the Siebel OLE DB Provider.

Server Connected Mode

In the server connected mode Siebel OLE DB Provider uses the Siebel Web client to communicate with the rest of the Siebel environment. In this mode, the Data Source name defines the connect string that is used by the Siebel Web client to connect with the Siebel Gateway Server. A sample Excel query that uses Siebel OLE DB provider in Server mode follows:

```
QueryType=OLEDB
Version=1

Connection=Provider=SiebelOLEDB.Provider.1;Password=SADMIN;
Persist Security Info=True;User ID=SADMIN;
DataSource=<Gateway machine name>,<name of the Enterprise>,<name
of the Object Manager>,<name of the Siebel Server>;

CommandType=Default
CommandText=select * from Contact where 'Job Title'=Manager
```

NOTE: There should be no blank lines between commands such as QueryType, Version, and CommandText. Connection information should be in one continuous line.

Local WIN32 Siebel Client Mode

In the local client mode, the Siebel eBusiness Object Manager allows the Siebel OLE DB Provider to connect to the Siebel application residing on the same machine. In the local client mode, the Data Source name defines a path to the Siebel configuration (*.cfg) file. Table 12 below defines the syntax.

Table 12. Data Source Syntax

String	Database
c:\...\siebel.cfg	Uses the default database
c:\...\siebel.cfg, Local	Uses the local database regardless of the default setting
c:\...\siebel.cfg, ServerDataSrc	Uses the server database regardless of the default setting

CAUTION: When you install using the Siebel EAI Connectors installation option, the Siebel OLE DB Provider can only be used in the Server Connected Mode. The Local Client Mode is not available.

A sample Excel query that uses Siebel OLE DB Provider in the local client mode follows:

```
QueryType=OLEDB
Version=1

Connection=Provider=SiebelOLEDB.Provider.1;Password=SADMIN;
User ID=SADMIN;Data Source=c:\Program
Files\SiebelApp\bin\siebel.cfg,lang=ENU

CommandType=Default
CommandText=select * from Contact where 'Job Title'=Manager
```

Multiple Language Considerations

Siebel OLE DB Provider supports multilingual operation. During installation, Siebel OLE DB Provider sets the language in which it was installed as the default language. However, you can install additional languages later, with each newly installed language becoming the new default. In order to use a nondefault language, you need to pass the Siebel OLE DB Provider a language parameter.

NOTE: The language you choose pertains only to the messages displayed. The text that is retrieved will be in the language in which it was stored.

Primary and Foreign Key Relationships

Siebel OLE DB Provider supports Primary and Foreign Keys for creating links between Siebel Business Components in a Siebel Business Object. With Siebel OLE DB Provider, you will be able to determine the relationships between business objects. Different OLE DB consumers may use this feature in different ways. For example, Microsoft Access shows Primary and Foreign Keys in a tree pane. You can use this information to build queries that use relationships between business objects.

NOTE: A Primary Key is a set of fields where there is more than one available key field that can establish a relationship, but only one of the keys is chosen by the DBA to be the Primary Key. A Foreign Key is a field whose values are keys in another relationship.

To enable primary and foreign key support

- 1** Access the Windows Registry and expand HKEY_CLASSES_ROOT.
- 2** Expand the key “CLSID.”
- 3** Expand and highlight the key {84C9F452-1ECC-11d3-9D36-0080C7AAC8A7}.
The default value should be “Siebel OLEDB Provider.”
- 4** Highlight the key “Parameters.”

- 5** Right click to select New > DWORD Value.
 - a** Set the Name of the new value to “EnablePkFk.”
 - b** Set the value in the Value Data field to 1.
- 6** Select File > Exit to close the Windows Registry and save your changes.

Using the Windows Event Viewer with Siebel OLE DB Provider

After you have initiated an OLE DB client to connect to Siebel OLE DB Provider, you can use the Windows Event Viewer to review the events associated with Siebel OLE DB Provider and troubleshoot as needed.

To view OLE DB Provider events

- 1** Start the Event Viewer.
- 2** Select the Application Log.
- 3** Look for events with a source of “Siebel OLE DB Provider.”

There should be two information events:

- The first event identifies Siebel OLE DB Provider DLL, ssceolpr.dll and the executable that loaded the DLL.
- The second event identifies the connection string used.

Viewing OLE DB Information

You can verify proper installation and troubleshoot problems with connecting to, or running queries with, Siebel OLE DB Provider and a Siebel data source.

To view OLE DB information

- 1 Using Windows Explorer, right click on each of the following OLE DB Provider DLLs.
 - bin\ssceolpr.dll
 - bin\ssceolwr.dll
 - bin\language\ssceolrs.dll

Where: *language* = the Siebel code for the Language Pack you are installing for this server; for example, enu for U.S. English.
- 2 Select Properties from the pop-up menu.
- 3 Select the Version tab in the Properties dialog.
- 4 Verify that the fields have the appropriate values.

NOTE: You can also view this information within Siebel Tools and any COM-related development tool, such as the Object Browser in Microsoft Visual Studio.

Siebel OLE DB and Multivalue Fields

Siebel allows you to create rowsets that contain multivalue fields (MVFs). However, when the rowset is issued in an OLE DB consumer such as MS Excel, only the first child record in the Siebel MVF field will be returned.

For instance, if you create a rowset based on the Opportunity BC and you add the Product MVF to the rowset, when you use the rowset in MS Excel only one record will display for each opportunity. The field/column “Product” will contain only the first product in the Multivalue fields list of products for the opportunity.

Retrieving Siebel Data Using OLE DB Consumers

You can use existing OLE DB Consumers to connect to a Siebel OLE DB rowset that you create using Siebel OLE DB Rowset wizard. This wizard helps you create an integration object that interfaces to Siebel business objects. For example, if you want to display data about accounts in Microsoft Access to take advantage of its graphical reporting tools, you specify the Account business object when creating Siebel OLE DB rowset. The wizard creates an integration object that, in effect, translates data from the Siebel business object format into data in the OLE DB rowset format.

Siebel OLE DB Provider for Siebel eAI supports four preconfigured applications as OLE DB Consumers:

- Microsoft Excel
- Microsoft SQL Server
- Microsoft Access

NOTE: Microsoft Access XP is not supported as a Siebel OLE DB Provider.

This section explains how to use the Siebel OLE DB Provider with these applications. After you have installed Siebel OLE DB provider library files and created your OLE DB rowsets within Siebel Tools, you are ready to use Siebel data within the framework.

NOTE: Siebel OLE DB Provider allows you to retrieve information from the Siebel database on a “read-only” basis. You can review the data and incorporate it into spreadsheets, databases, and Web pages, as needed. You cannot make changes to the data or affect the Siebel database in any way with Siebel OLE DB Provider.

Creating and Modifying Siebel OLE DB Rowsets

This section covers the following topics:

- [To create Siebel OLE DB rowset object in Siebel Tools on page 97](#)
- [To modify a Siebel OLE DB rowset object in Siebel Tools on page 100](#)

To create Siebel OLE DB rowset object in Siebel Tools

The Siebel OLE DB Rowset object is exposed as a Siebel integration object in Siebel Tools. You create the OLE DB integration object using the OLE DB Rowset wizard.

- 1** Start Siebel Tools.
- 2** Lock the project from which you will be creating the OLE DB rowset.
- 3** Choose File > New Object to display the New Object Wizards dialog box.
 - a** Select the EAI tab.
 - b** Select the OLE DB Rowset icon and click OK.

The New OLE DB Rowset wizard appears.

- 4** Select the items from the drop-down lists to define your rowset. When finished, click Next.
 - Choose the locked project from which you will be creating the OLE DB rowset.
 - Choose the Siebel Business Object that represents the data you want as the basis of your OLE DB rowset.

The wizard displays the business components that are used by this Business Object.

- Choose the Business Component you want to use to populate your OLE DB rowset with data.

- The system automatically generates a unique name for your OLE DB rowset by concatenating the Business Object name and the Business Component name and adding a unique number (starting with 1); for example, Account_Contact_1.” You can change this name as needed.

NOTE: Siebel Tools limits this field to a maximum of 75 characters. If it should happen that the combination of the Business Object and the Business Component names come to more than 75 characters, the name will be truncated, starting from the right. Again, you can change this name as needed. It is recommended that you use the following: < Business Object > _ < Business Component > _ < Unique Identifier > .

- Choose the Visibility Type for the rowset.

You can limit the data users are allowed to see based on their Siebel-defined responsibility and visibility privileges. Visibility levels can be set for each rowset.

5 Click Next.

The OLE DB Rowset Fields page appears.

6 From the list of available field names, select the fields you want to include in your OLE DB table. You can select one or more fields by using the following standard techniques:

- To select a single field, click on a field name, then click the right arrow button to move the field name to the scrolling field on the right.
- To select multiple fields, click on a field name, hold down the Ctrl key on your keyboard and click on another field name.

Repeat this process to select any number of field names. Alternatively, you can click on a field name, then hold down the Shift key on your keyboard and click on another field name farther down the list. This selects all of the field names between and including your two selections.

- 7** Rearrange the order of the field names, if necessary, by clicking the up and down arrow buttons to move a selected field name in one direction or another.

NOTE: Rearranging the field names at this point in the process can make it easier to view the data in a more meaningful order when you access the OLE DB table from another application.

- 8** Click Next.

The OLE DB Wizard displays the OLE DB Rowset - Finish page to allow you to review and confirm your selections.

- To change one or more of your selections, click Back to return to the previous page of the wizard.
- If you are satisfied with your choices, click Finish.

The OLE DB wizard creates the OLE DB integration object.

- 9** Recompile the `.srf` file. Siebel OLE DB Provider retrieves Integration Object information from the `.srf` file.

- Select Tools > Compile (or press F7).

You can now access the Siebel OLE DB integration object using any external OLE DB-enabled applications.

NOTE: Integration objects are created slightly differently by the OLE DB Rowset wizard than by any other integration object wizard. In other chapters of this guide, you will read that integration objects provide the interface between external data objects and the Siebel property set format. The OLE DB Provider integration objects you create transform data between Siebel Business Object Interfaces and OLE DB rowsets.

To modify a Siebel OLE DB rowset object in Siebel Tools

- 1** Lock the project from which the OLE DB rowset was originally created.
- 2** Access the list of Integration Objects and highlight the OLE DB rowset you want on the Integration Objects list.
- 3** Right click on the OLE DB rowset you want to modify.
- 4** Select Edit OLE DB Rowset from the pop up menu.

The Edit OLE DB Rowset wizard appears, displaying the details on the rowset. This first page is read only, but it can be changed using visibility rules.

- Click Next to proceed to the Field selection page.
 - The OLE DB Rowset Fields page appears.
- 5** From the list of available field names, select the fields you want to include in your OLE DB table and deselect others that you want to remove.
 - 6** Rearrange the order of the field names, if necessary, by clicking on the up and down arrow buttons to move a selected field name in one direction or another.
 - 7** Click Next.

The OLE DB Wizard displays the OLE DB Rowset - Finish page to allow you to review and confirm your selections.

- To change one or more of your selections, click Back to return to the previous page of the wizard.
- If you are satisfied with your choices, click Finish.

The OLE DB wizard modifies and saves the OLE DB integration object.

- 8** Recompile the `.srf` file. Siebel OLE DB Provider retrieves Integration Object information from the `.srf` file.

You can now access the Siebel OLE DB integration object using any external OLE DB-enabled applications.

Viewing Siebel OLE DB Rowsets in MS Office

This section discusses a variety of ways to view information using Microsoft Office applications.

To view Siebel data in Microsoft Excel

Business analysts will find the Siebel OLE DB Provider support useful for analyzing account data and other information stored in the Siebel database and incorporating that data in an Excel spreadsheet. To use Siebel OLE DB Provider from Excel, you create an external query that connects to Siebel OLE DB Provider.

- 1 Open Microsoft Excel.
- 2 Choose File > New to open a new spreadsheet.
- 3 Create a query using Notepad or any text editor. If you want to use a different data source then the one defined in the `.cfg` file you can do so here.
- 4 Enter the connect string, and any other information.
- 5 Save your work.

The following example connects to Siebel OLE DB Provider and sends a command to retrieve all records from the Contact virtual table where the position is equal to Manager. You can store any query in a `*.qry` file and execute the query at a later time.

The properties identify the contents of the file as an OLE DB type query and provide the connection parameters and query text. The properties QueryType, Version, Connection, CommandType and CommandText are required. The structure defined is mandatory and cannot be changed.

[Table 13](#) shows the required properties for an Excel query file.

Table 13. Excel Query Properties

Property	Value/Description
QueryType	OLEDDB.
Version	1.
CommandType	Default.

Table 13. Excel Query Properties

Property	Value/Description
QueryText	Set to the text of the query to be executed by the Siebel OLE DB Provider.
Connection	Several parameters, each separated by a semicolon: <ul style="list-style-type: none">■ Provider. Set to the Siebel OLE DB Provider COM component, SiebelOLEDB.Provider.1.■ Data Source. Set to the Siebel OLE DB Provider connection string.■ User ID and Password are optional. If not set, OLE DB Provider will prompt for them.

For example:

```
QueryType=OLEDB
Version=1

Connection=Provider=SiebelOLEDB.Provider.1;Password=db2;
User ID=SADMIN;Data Source=siebel://10.24.20.5/siebel/sseobjmgr;

CommandType=Default

CommandText=select "City" from Contact_Contact_1 where "Bill To
City"="Menlo Park"
```

NOTE: There should no blank lines between each commands such as QueryType and CommandText. Connection information should be on one continuous line.

To view Siebel data in Microsoft Access

You can use Microsoft Access to create ad hoc reports. Using the Access Data Access Page Designer's drag-and-drop capabilities, you can create Web pages by selecting, dragging, and dropping Siebel OLE DB tables onto the Access form. The underlying OLE DB infrastructure writes the necessary information, and the newly created Web page accesses the Siebel virtual table data transparently.

- 1 Open Microsoft Access.
- 2 Choose File > New.

- 3** On the New dialog box, select the Data Access Page and click OK.

The new Data Access Page dialog box displays.

- 4** Select Design View and click OK.

The Data Link Properties dialog box displays.

- 5** Select the Provider tab.

- 6** Select Siebel OLE DB Provider from the picklist and click Next.

- 7** On the next page, fill in the parameters, including the Data Source and User name properties.

- 8** Click OK to save the changes.

The Siebel OLE DB Provider login dialog appears.

- 9** Provide the password and click OK.

You are presented with the designer and the Field List dialog box displaying the available Integration Objects.

- 10** Design and save this Web page.

Access this Page View to review the results of the query from the OLE DB Provider.

Viewing Siebel Data Using Microsoft SQL Server Distributed Queries

In Microsoft SQL Server version 7.0, distributed queries enable SQL Server users to access data outside a SQL Server-based server, within either other servers running SQL Server or other data sources that expose an OLE DB interface. OLE DB provides a way to uniformly access tabular data from heterogeneous data sources.

A distributed query for the purpose of this document is any SELECT, INSERT, UPDATE, or DELETE statement that references tables and rowsets from one or more external OLE DB data sources.

A remote table is a table that is stored in an OLE DB data source and is external to the server running SQL Server executing the query. A distributed query accesses one or more remote tables.

Siebel OLE DB provider may be used as one of the data sources in the SQL 7.0 distributed query.

```
SELECT * FROM
OPENROWSET('SiebelOLEDB.Provider.1','<ConnectionString>'; '<UserId>'
; '<Password>', <Siebel OLE DB Provider query text>)
```

For example:

```
SELECT * from OPENROWSET('SiebelOLE DB.Provider.1',
'somelhost,siebel,objmgr,w_name';

'SADMIN'; 'SADMIN' ,

SELECT "First Name", "Last Name" from Contact_Contact_1 where ("Job
Title" = "Manager")
```

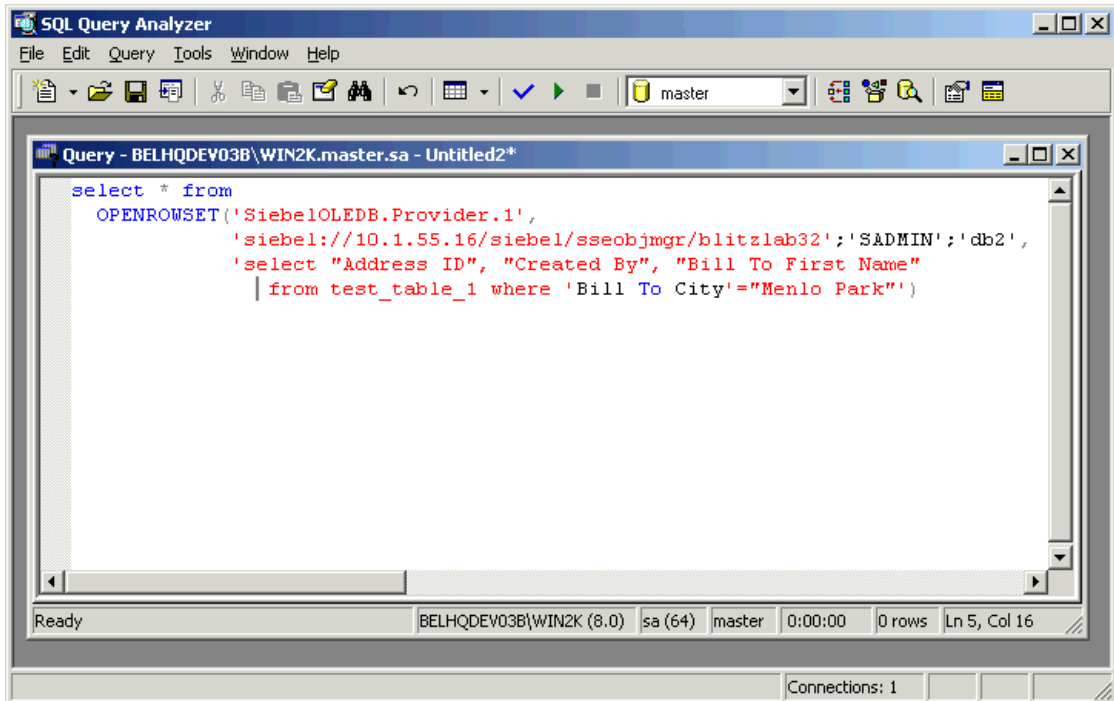
NOTE: For more information about Microsoft SQL Server distributed queries, please refer to Microsoft SQL Server 7.0 documentation.

Database administrators (DBAs) will find the OLE DB support in Siebel eAI useful for checking data integrity in Siebel applications and other database-related tasks. With Siebel OLE DB Provider, table, row, and field information are displayed in the Microsoft SQL Analyzer for review and action.

To view Siebel data in SQL Analyzer

- 1** Start the Microsoft SQL Server Query Analyzer tool.
- 2** Connect to an SQL Server on which the Siebel OLE DB Provider has been installed.

- 3 Enter the SQL Server query text in the query window, as shown in the following illustration.



- 4 Enter a connect string similar to this (providing your actual ID, password, selection criteria, and so on):

```
SELECT * from OPENROWSET('SiebelOLEDB.Provider.1','siebel://
10.1.55.16/siebel/sseobjmgr/blitzlab32';

'SADMIN'; 'db2',

'select "Address ID","Created By","Bill To First Name" from
test_table_1 where 'Bill To City'="Menlo Park"')
```

Retrieving Siebel Data Using Scripts and Custom Applications

Siebel OLE DB Provider for Siebel eAI supports:

- Visual Basic (VB), C++ , VBScript, and Javascript
- Active Server Pages (ASP)

This section explains how to use the Siebel OLE DB Provider with these technologies. After you have installed Siebel OLE DB provider library files and created your OLE DB rowsets as integration objects within Siebel Tools, you are ready to use Siebel data within the support framework.

NOTE: Siebel OLE DB Provider allows you to retrieve information as needed from the Siebel data repository on a “read-only” basis. You can review the data and incorporate it into spreadsheets, databases, and Web pages, as needed, but you cannot make changes to the data and affect the Siebel data repository in anyway using Siebel OLE DB Provider connection.

This section covers these topics:

- [Writing an OLE DB Consumer on page 106](#)
- [Retrieving Siebel Data Using VB and ASP on page 110](#)

Writing an OLE DB Consumer

You can write your own OLE DB consumer or access Siebel OLE DB Provider programmatically using standard programming languages, such as C++ and Visual Basic, or using scripting languages, such as VBScript and JavaScript.

This section describes the objects and object interfaces provided by OLE DB to facilitate access to Siebel OLE DB Provider and other OLE DB providers. This is a brief summary of the OLE DB interfaces. It is intended to identify the support available from the Siebel OLE DB Provider. You can find more information on the OLE DB interfaces from reference documentation published by Microsoft and others.

OLE DB Object Support in Siebel OLE DB Provider

The following list summarizes the OLE DB version 2.1 objects that are supported by the current version of Siebel OLE DB Provider.

- DataSource
- Session
- Command
- Rowset

Siebel OLE DB DataSource Object

You can use any OLE DB-compliant products to access Siebel objects. These products include Microsoft Excel, Microsoft Access, and others. These applications are referred to as *consumers*. You must create the DataSource object by defining the object in an OLE DB consumer. During the creation process, you provide the properties and parameters required for the DataSource object to connect to the Siebel environment. The consumer then uses the DataSource object to create one or more Session objects.

The following DataSource object OLE DB interfaces are supported in the current version of Siebel OLE DB Provider:

- IDBCreateSession
- IDBInitialize
- IDBProperties
- IPersist
- ISupportErrorInfo

You must specify DataSource properties to successfully initialize and authorize the connection to the Siebel environment, as shown in [Table 14](#).

Table 14. OLE DB DataSource Properties

Property ID	Description
DBPROP_AUTH_USERID	A Siebel user name.
DBPROP_AUTH_PASSWORD	The password assigned to the Siebel user.
DBPROP_INIT_DATASOURCE	The connect string or the path to the local configuration file.
DBPROP_INIT_PROMPT	Specifies the prompt mode supported for data source initialization. This provider supports every prompting mode. The default value is DBPROMPT_NOPROMPT.

Siebel OLE DB Session Object

The DataSource object creates and uses the Session object to create one or more Rowset objects. The following Session object OLE DB interfaces are supported in the current version of Siebel OLE DB Provider:

- IDBCreateCommand
- IGetDataSource
- IOpenRowset
- ISessionProperties
- IDBSchemaRowset
- ISupportErrorInfo

Siebel OLE DB Command Object

The OLE DB Command object supports and provides a subset of SQL commands that you can use to query the Siebel business objects supported by Siebel OLE DB Provider. The OLE DB consumer creates the Command object by executing `IDBCreateCommand.CreateCommand`. Multiple commands can be created and executed during a single session. Siebel OLE DB Provider supports the following OLE DB interfaces on the Command object:

- `IAccessor`
- `ICommand`
- `ICommandWithParameters`
- `ICommandProperties`
- `ICommandText`
- `IColumnsInfo`
- `IConvertType`
- `ISupportErrorInfo`

Siebel OLE DB Provider Command Syntax

Siebel OLE DB Provider Command object supports a subset of SQL, which allows OLE DB consumers to issue simple query statements against one virtual table.

The following query is an example of the type of statement you can execute:

```
SELECT 'First Name', 'Last Name'
FROM Contact
WHERE 'Job Title' = Manager;
```

The general syntax of Siebel OLE DB Provider Command language is as follows:

- Required terms are delimited by square brackets ([]).
- Optional terms are delimited by angle brackets (< >).

```
SELECT [ column/list of columns/* ]
FROM [ table_name ]
<WHERE> [ column = value ] <AND> [ column=value ]
<ORDER BY> [ column ];
```

NOTE: The current Command language does not support the JOIN construct. A command can be issued against only one virtual table.

Siebel OLE DB Rowset Object

The Session object creates the Rowset. The consumer can also call ICommand:Execute to create a Rowset. The data in the Rowset object is displayed in tabular format. The following Rowset object OLE DB interfaces are supported in the current version of Siebel OLE DB Provider:

- IAccessor
- IColumnsInfo
- IConvertType
- IRowset
- IRowsetInfo
- ISupportErrorInfo

Retrieving Siebel Data Using VB and ASP

You can view Siebel data using Visual Basic. Programmers writing custom VB programs and scripts that need to access data in the Siebel repository will find the Siebel OLE DB Provider support useful for such tasks.

To view Siebel data using Visual Basic

- 1** Start Microsoft Visual Basic.
- 2** Enter code similar to the following example (providing your actual ID, password, selection criteria, and so on):

```
' This program will connect to the Siebel OLE DB Provider, retrieve data
' and save the records in a file with tab separated fields. Other tools
' can then be used to further process the data.

Dim Fso, File

' Setup program parameters.
ProviderString = "SiebelOLEDB.Provider.1"
DataSourceString = "siebel://MyGateway/MyEnterprise/MyObjMgr/MyServer"
UserIdString = "MyUserId"
PasswordString = "MyPassword"
OutFileString = "output.txt"

' Build the connection string.
ConnectString = "Provider=" & ProviderString & ";User Id=" & UserIdString & _
";Password=" & PasswordString & ";Data Source=" & DataSourceString & ";"
```

```

' Ask the user if they are ready to establish a connection
' and retrieve the data.
Message = "Ready to connect using" & Chr(13) & Chr(10) & _
ConnectString & Chr(13) & Chr(10) & _
"Do you want to continue?"
Continue = MsgBox(Message, vbYesNo, "Ready to Connect")

If Continue = vbYes Then
' Create the output file for storing the data.
Set Fso = CreateObject("Scripting.FileSystemObject")
Set File = Fso.OpenTextFile(OutFileString, 2, True)

' Establish a connection.
Set Connection = CreateObject("ADODB.Connection")
Connection.Open ConnectString

' Execute a query to create a record set.
' Retrieve all accounts involved in any electrical related business.
QueryString = "Select * from Account_Account_1 where 'Line of Business' =
'Electrical*'"
Set RecordSet = Connection.Execute(QueryString)

' If we have any data then write a header record with column names.
If Not RecordSet.EOF Then
First = True
For Each Field in RecordSet.Fields
' Write each field within double quotes and a tab separator between them.
If First Then
File.Write """"
First = False
Else
File.Write """" & Chr(9) & """"
End If
File.Write Field.Name
Next
File.WriteLine """"
End If

' Keep track of the number of records.
RecordCount = 0
Do While Not RecordSet.EOF
First = True
For Each Field in RecordSet.Fields
' Write each field within double quotes and a tab separator between them.
If First Then
File.Write """"
First = False
Else
File.Write """" & Chr(9) & """"
End If
File.Write Field.Value
Next
File.WriteLine """"
RecordCount = RecordCount + 1
RecordSet.MoveNext
Loop

```

```
' Clean up local variables.
RecordSet.Close
Connection.Close
Set RecordSet = nothing
Set Connection = nothing
File.Close
Set File = nothing
Set Fso = nothing

' Notify the user of the number of records retrieved and stored.
Message = "Successfully retrieved and stored " & RecordCount & _
" records in " & OutFileString
MsgBox Message, vbOkOnly, "Data Retrieved"
End If
```

Programmers, Webmasters, and others who need to display data from the Siebel database in a Web page or portal will find Siebel Systems' OLE DB Provider support useful for such tasks.

To view Siebel data using ASP

- 1 Create an HTML page to display a form for gathering user input with the following HTML:

```
<html>
<head>
<TITLE>Request Account Information</TITLE>
</head>
<body>
<FONT FACE="Verdana, Arial, Helvetica">
<br><P ALIGN=left><FONT SIZE=4>Request Account Information.</
FONT></P>
<HR ALIGN=center NOSHADE SIZE=4>
<form action="test04.asp" method="POST">
<P ALIGN=left>Please enter your user id, password and account id
to access your account information.</P>
<TABLE CELLPADDING=4>
<TR>
<TD>User Id:</TD>
<TD><input type="TEXT" name="UserId"></TD>
</TR>
<TR>
<TD>Password:</TD>
<TD><input type="PASSWORD" name="Password"></TD>
</TR>
<TR>
<TD>Account Id:</TD>
<TD><input type="TEXT" name="AccountId"><br></TD>
</TR>
</TABLE>
```

```

<br>
<input type="submit" value="Submit">
</form>
</FONT>
</BODY>
</html>

```

- 2 Create an Active Server Page (ASP) file that retrieves the input parameters, connects to the Siebel OLE DB Provider and builds the output to be sent back and displayed in the browser.

Use the following HTML code:

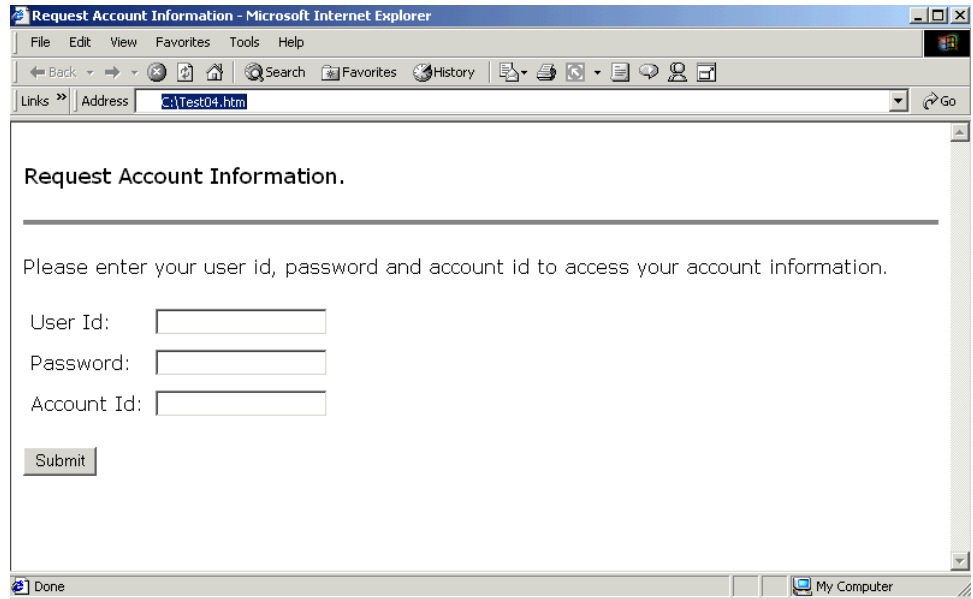
```

<HTML>
<HEAD>
<TITLE>Account Information</TITLE>
</HEAD>
<BODY>
<FONT FACE="Verdana, Arial, Helvetica" SIZE=2>
<!-- Display the time the request was processed -->
Your request has been processed at
<%Response.Write time%>.<br><br>
<!-- Get the form input data -->
<%
  UserId = Request.Form("UserId")
  Password = Request.Form("Password")
  AccountId = Request.Form("AccountId")
  'Connect to Siebel and retrieve the account information as an OLE
  DB Rowset.
  Set Connection = Server.CreateObject("ADODB.Connection")
  ConnectString = "Provider=SiebelOLEDB.Provider.1;User Id=" +
  UserId + ";Password=" + Password + ";Data Source=siebel://
  MyGateway/MyEnterprise/MyObjMgr/MyServer;"
  Connection.Open ConnectString
  If Len(AccountId) = 0 Then
    Query = "Select * from Account"
  Else
    Query = "Select * from Account where Id = '" + AccountId + "'"
  End If
  Set RecordSet = Connection.Execute(Query)
  If Not RecordSet.EOF Then
    %>
    This is your current account information.
    <br><br>
    <!-- Build a table to display the data -->
    <TABLE CELLPADDING=4>

```

```
<!-- BEGIN column header row -->
<TR>
<%For Each Field in RecordSet.Fields%>
<TH><FONT SIZE=2><%Response.Write Field.Name%></FONT></TH>
<%Next%>
</TR>
<%Do While Not RecordSet.EOF%>
  <TR>
    <%For Each Field in RecordSet.Fields%>
      <TD><FONT SIZE=2><%Response.Write Field.Value%></FONT></
TD>
    <%Next%>
  </TR>
  <%
    RecordSet.MoveNext
  Loop
  %>
</TABLE>
<%
End If
' Clean up variables.
RecordSet.Close
Connection.Close
Set RecordSet = nothing
Set Connection = nothing
%>
</FONT>
</BODY>
</HTML>
```

The form created by the HTML page prompts the user for input, as shown in [Figure 2](#).



The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads 'Request Account Information - Microsoft Internet Explorer'. The menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains buttons for Back, Forward, Stop, Home, Search, Favorites, History, Print, and Go. The address bar shows 'Links >> Address C:\Test04.htm'. The main content area displays the following text:

Request Account Information.

Please enter your user id, password and account id to access your account information.

User Id:

Password:

Account Id:

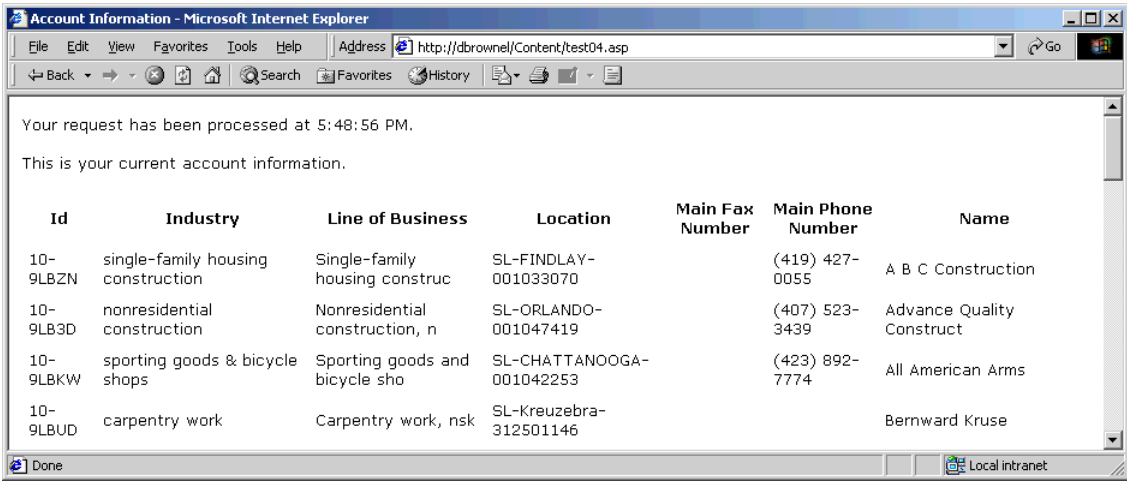
The status bar at the bottom shows 'Done' and 'My Computer'.

Figure 2. Sample HTML User Input Page

The output is shown in [Figure 3 on page 116](#).

Using Siebel OLE DB Provider

Retrieving Siebel Data Using Scripts and Custom Applications



Your request has been processed at 5:48:56 PM.

This is your current account information.

ID	Industry	Line of Business	Location	Main Fax Number	Main Phone Number	Name
10-9LBZN	single-family housing construction	Single-family housing construc	SL-FINDLAY-001033070		(419) 427-0055	A B C Construction
10-9LB3D	nonresidential construction	Nonresidential construction, n	SL-ORLANDO-001047419		(407) 523-3439	Advance Quality Construct
10-9LBKW	sporting goods & bicycle shops	Sporting goods and bicycle sho	SL-CHATTANOOGA-001042253		(423) 892-7774	All American Arms
10-9LBUD	carpentry work	Carpentry work, nsk	SL-Kreuzebra-312501146			Bernward Kruse

Done Local intranet

Figure 3. Sample HTML Output

Troubleshooting

This section describes common connection problems when using OLE DB. Please consult the Windows Event Log to view the details on other OLE DB and Siebel OLE DB Provider errors.

Problem	Initialization of the data source failed.
	Either:
Description	<ul style="list-style-type: none"> ■ You have selected an invalid .cfg file in your Connection string. ■ The .cfg file you have selected has not been updated to support OLE DB.
Sample Problem Code	<pre> QueryType=OLEDB Version=1 Connection=Provider=SiebelOLEDB.Provider.1;Password=SADMIN;Persist Security Info;User ID=SADMIN;Data Source=c:\sea601\client\bin\siebel.cfg; CommandType=Default CommandText= select * from LabRowset </pre>
Solution	Check the database server or contact your database administrator. Make sure the external database is available, and then try the operation again. If you see this message again, create a new data source to connect to the database.
Problem	The query did not run or the database table could not be opened.
Description	<p>Either:</p> <ul style="list-style-type: none"> ■ The database name in your .cfg file is not set correctly. ■ The name of the table in your query is incorrect.
Sample Problem Code	<pre> QueryType=OLEDB Version=1 Connection=Provider=Siebel OLEDB Provider;Password=SADMIN;Persist Security Info=False;User ID=SADMIN;Data Source=c:\sea601\client\BIN\siebel.cfg CommandType=Default CommandText= select * from LabRowset </pre>
Solution	Check the database server or contact your database administrator. Make sure the external database is available and has not been moved or reorganized, then try the operation again.

Problem SQL Query Analyzer error message.

Description The following error is generated in the SQL Query Analyzer when the Object Manager on the Server is not initialized:

```
Server: Msg 7399, Level 16, State 1, Line 1  
OLE DB provider 'SiebelOLEDB.Provider.1' reported an error.
```

(The OLE DB provider 'SiebelOLEDB.Provider.1' indicates that the object has no columns.)

This error typically occurs whenever the .srf is recompiled, the Gateway Server and the Siebel Server are stopped, and then the query is rerun.

This error may be related to SQL Server caching of the OLEDB datasource or to the servers not being successfully restarted.

**Sample
Problem Code** n/a

Solution Reboot. After rebooting the servers, all queries should work.

Problem Test connection failed because of an error in initializing provider. 0x80040e73

Description Failure to set the DataSource property in the [Siebel] section correctly will generate this error when testing the connection. (You will also get an entry in Windows Event Log regarding this failure.)

**Sample
Problem Code** n/a

Solution Set the connect string to the Siebel Thin Client environment within the double quotes for the host parameter in the `tclient.htm` file, or in the path to the `.cfg` file for your local client. In the `.cfg` file, make sure the `DataSource` property in the [Siebel] section is set for the local client using Local, Sample, or Server. For example, set to “Sample” for Sample database.

Problem	Receiving “Could not process object select * from GPTest2” error message
Description	<p>If you did not make the custom OLE DB available to the Siebel Server Object Manager, you will see the following error when you run the query:</p> <pre>Server: Msg 7357, Level 16, State 2, Line 1. Could not process object 'select * from GPTest2'</pre>
Sample Problem Code	n/a
Solution	Make sure that you have copied the latest <code>.srf</code> to the <code>Server\objects</code> directory and have restarted the server. These actions will make the custom OLE DB available to the Siebel Server Object Manager.
Problem	Provider "SiebelOLEDB.Provider.1 supplied inconsistent metadata for a column. Metadata information was changed at execution time"
Description	The length of the column in the two applications do not match.
Sample Problem Code	n/a
Solution	When SQL Server reports “inconsistent metadata,” the user can modify the field length in the rowset definition in Tools.
Problem	Unable to query SODP using MSSQL Query Analyzer
Description	<p>When a connection is established with one type of connection string and then another connection is attempted with a different type of connection string, it will fail on the second connection attempt, generating following error message:</p> <pre>Error message with MSSQL: OLE DB provider "siebelOLEDB.Provider.1"reported an error. Provider caused a server fault in an external process.</pre>
Sample Problem Code	n/a
Solution	Restart the MS SQL Server.

This chapter discusses the Microsoft BizTalk Server.

About Microsoft BizTalk Server

Siebel eBusiness Applications provide technology for data integration between Siebel applications and Microsoft BizTalk Server. This allows diverse external and internal applications to communicate with Siebel applications using proven transports, regardless of the original data format. This section discusses the Siebel BizTalk Server interface, presenting details on architecture and related components, including transports and message formats.

Microsoft BizTalk Server provides a translation gateway that can read and write XML, positional or delimited flat files, and both formats of Electronic Data Exchange (EDI): United Nations/Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) and ANSI X12. BizTalk Server provides a resource for the secure and reliable delivery and transformation of business documents regardless of source, data format, or communication protocol in use.

Siebel BizTalk Server Adapter

With Siebel eBusiness Applications support for Microsoft BizTalk Server, you can share Siebel data with external and internal contemporary and legacy systems by generating and exchanging XML documents through BizTalk Server, without having to create a custom solution or write custom code. This allows for interaction between business processes within a single organization and automated procurement arrangements. The Siebel interface for BizTalk Server provides the following features and functionality.

- **Generation and Transformation of XML Documents.** To transform your schema into a trading partner's schema (and their schema to yours) using Microsoft's BizTalk Mapper GUI tool. BizTalk Mapper generates W3C-standard Extensible Stylesheet Language Transformation-based maps to perform the translation between trading partners' schema. This allows you to create and edit DTD schema and XDR and transform to a partner's schema format.
- **XML Support.** To simplify B2B and internal systems data exchange, document exchanges made through the Siebel BizTalk interface are in W3C-standard XML. Document transformation is in W3C-standard XSLT.

XML messages are the key substrate of application integration. Siebel applications can exchange XML messages with other application systems through Microsoft's BizTalk Server over standard protocols and transports. The interface between Siebel application and BizTalk Server supports both the Siebel application as the message sender and the Siebel application as the message receiver.

The Siebel BizTalk interface provides a comprehensive mechanism for interacting with Siebel APIs in order to both "get" (extract) and "put" (insert or update) information. On an outbound request from the Siebel client, the Siebel application transforms the external interface object into XML and sends it to its destination. On an inbound transaction from an external system, the Siebel application receives an XML message, which is then validated against the appropriate Siebel Integration Object.

NOTE: For more information on Siebel applications and XML, see *XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition*.

- **Support for Multiple Transports and Protocols.** To allow for different options when sending and receiving data through BizTalk Server. The Siebel BizTalk adapter supports Hypertext Transfer Protocol (HTTP), Microsoft Message Queuing (MSMQ), Application Integration Component (AIC), Component Object Model (COM), and File.

Where to Get More Information

Table 15 lists other sources of information on associated technologies.

Table 15. Other Information Sources

Technology	Reference
EAI HTTP Transport	Chapter 4, “EAI HTTP Transport” in this book
eScripts	<i>Siebel Tools Online Help, MidMarket Edition</i>
ActiveX	<i>Siebel Tools Online Help, MidMarket Edition</i>
Microsoft BizTalk Server (BTS)	Microsoft BizTalk Server documentation. (Available with BizTalk or download from http://www.microsoft.com .)
Siebel Installation	<i>Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition</i> <i>Siebel Web Client Administration Guide, MidMarket Edition</i>
Siebel EAI Integration Objects	<i>Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition</i>
Siebel Workflow	<i>Siebel Business Process Designer Administration Guide, MidMarket Edition</i>
XML (Siebel-specific)	<i>XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition</i>

Software Architecture

Communication between Siebel applications and Microsoft's BizTalk Server is based on the Siebel eBusiness Application Integration (eAI) framework. The BizTalk Server provides the technology for application integration and data transformation. When you use Microsoft's BizTalk Server to exchange documents, the Siebel application constructs an XML document that is sent to BizTalk for data transformation and subsequently sent on to your trading partners. Your trading partners then communicate back to you in a similar manner. [Figure 4](#) illustrates this process.

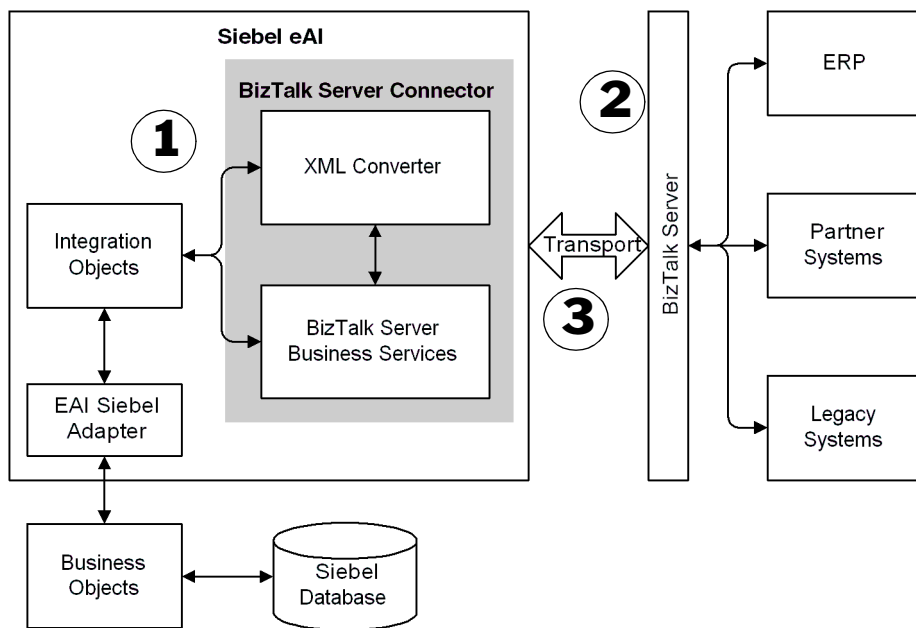


Figure 4. Siebel BizTalk Interface Architecture

As shown in [Figure 4](#), interfacing between a Siebel application and BizTalk Server is a three-step process:

- 1** First, expose any Siebel Integration Object to BizTalk Server using the Schema Generator Wizard in Siebel Tools.
- 2** Next, import the schema into BizTalk to create document specifications.
- 3** Finally, exchange integration messages over any O/S, using the appropriate choice from the supported transports, using BizTalk Server for mapping and message transfer.

Schema Generation Support

Siebel Tools provides the functionality to generate schema in the Document Type Definition (DTD) format and Microsoft's XML-Data Reduced (XDR) format. You import the generated schema into BizTalk for further processing. Here is what's involved:

- **Schema Wizard.** You use the Schema Generator Wizard in Siebel Tools to generate DTD or XDR Siebel Integration Objects. The exposed integration objects are imported into BizTalk and stored on WebDAV as BizTalk *document specifications*.
- **Data Mapping.** The document specifications created in BizTalk are used to map between the Siebel-published schema and partner applications' schema using the Microsoft GUI tool, BizTalk Mapper. Maps are stored on WebDAV in XSLT format.

Exchanging Integration Messages

Siebel's BizTalk interface supports the following message formats and transport protocols.

Message Data Format. Both inbound and outbound messages use XML. BizTalk Server performs any data translation, if required, using XSLT.

Transport Protocol. Siebel applications and Microsoft BizTalk Server exchange inbound and outbound messages using the following transports:

- For Heterogeneous environments, the Siebel application and BizTalk communicate using:
 - File (read and write)
 - HTTP (Hypertext Transfer Protocol)
- For Windows-only environments, the Siebel application and BizTalk communicate using:
 - COM (Component Object Model)
 - AIC (Application Integration Component)
 - File (read and write)
 - HTTP (Hypertext Transfer Protocol)
 - Message Queuing (MSMQ)

Inbound and Outbound Interfacing

As the receiver, the Siebel application can receive inbound messages using MSMQ, File, Siebel's AIC component, or HTTP.

As the sender, the Siebel application sends outbound XML documents by means of MSMQ, File, HTTP, or the BizTalk Server COM Interchange Interface.

Figure 5 illustrates the inbound and outbound processes.

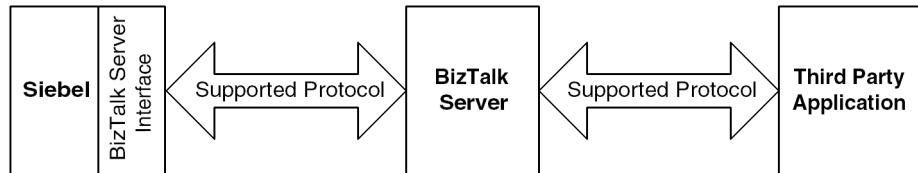


Figure 5. Siebel Application and BizTalk Inbound and Outbound Interfacing

- **Asynchronous Messaging.** The Siebel client sends an outbound message without waiting for an acknowledgment. It is free to process other events.
- **Synchronous Messaging.** The Siebel client sends an outbound message and expects a reply within a given timeframe. After the return message has been received, the Siebel application resumes processing of any events that were waiting to proceed.

Understanding Siebel BizTalk Server Adapter Through Scenarios

Siebel BizTalk adapter allows administrating, managing, and executing document exchange with various applications, without programming. There are many business scenarios where BizTalk Server can be used to integrate a Siebel application with other applications—including another Siebel application implementation—both internally and externally from an organization. In this communication process, you configure a Siebel application to be the sender, the receiver, or both.

In another scenario, say you need to get the same set information to a number of different trading partners. In order to accomplish such a Siebel application-to-multiple applications connection, you would first create a Distribution List using BizTalk Server's Management Desk. This allows for single document to be sent to multiple recipients, with personalized processing for each recipient. Each recipient can have its own channel, which defines recipient specific business rules, format, and transport protocols.

Here's how this would work. The Siebel application submits an XML document to BizTalk Server over HTTPS. BizTalk sends the business document to multiple recipients by invoking the channel associated with the distribution list. BizTalk server then uses the properties of each of the messaging ports within that distribution list to send the message to the destinations over the protocol specified in the corresponding ports.

There are also times when two Siebel applications need to exchange data. Since no external system is involved, typically no data transformation is required. Typically, this might be used for distributing information from a central repository to other global systems for exchanging of sales opportunities.

The first Siebel application sends a message to the second Siebel application to pass along sales opportunities. A new opportunity is created in the first Siebel application and is assigned to a partner. This new opportunity is sent using Siebel eAI to the second Siebel application. The second Siebel application receives the message and adds the new opportunity. Finally, the Product Catalog is updated in real-time. The product information is added or changed on the Siebel application and the changed product information would then be sent to the Siebel application.

Preparing to Use the Siebel BizTalk Adapter

This section explains how to set up and configure your servers for Siebel eBusiness Applications to communicate with BizTalk. The set up here is recommended, but your setup will reflect decisions made at your organization for using BizTalk.

Additionally, this section explains how to use Integration Objects with BizTalk so that you can communicate bidirectionally (send and receive messages) with your trading partners. A key step is to create the schema for the integration objects you will be using. Much of the actual communication, however, happens outside of Siebel applications, through the various BizTalk applications.

If you need specific information on how to use the applications that are included in Microsoft's BizTalk Server—which include BizTalk Mapper, BizTalk Editor, BizTalk Messaging Manager, and others—refer to the BizTalk Server documentation and help system. (The help system for BizTalk is also available from Microsoft's Web site.)

Installing Software and Creating BizTalk Doc Specs for Siebel Integration Objects

Checklist

- ☐ Set up BizTalk Server and each Siebel Client that will be communicating with BizTalk (a one-time operation).

For details, see [“Installing and Configuring Software for Servers and Clients” on page 130](#).

- ☐ Generate the DTD or XDR schema for the integration objects you want to use and import them to BizTalk Server, where they become BizTalk “document specifications,” BizTalk's proprietary XDR hybrid format (one time for each integration object).

For details, see [“Siebel Integration Objects” on page 131](#) and [“Data Types” on page 137](#).

Installing and Configuring Software for Servers and Clients

Before you can establish a relationship with your trading partners through BizTalk, or set up an application-to-application connection, you need to set up and configure servers and clients.

The components required for BizTalk integration are installed when you install Siebel EAI Connectors (as part of a Custom Installation). The default configuration is the Siebel database and Siebel Gateway Server installed on the Siebel Server machine. Your deployment, however, might very well have a different configuration.

CAUTION: The Siebel BizTalk interface files are not installed automatically when you do a Typical or Compact installation. You must install BizTalk files as a Custom installation by selecting Custom Installation, then EAI Connectors, and then the Siebel Connector for Microsoft BizTalk Server option.

NOTE: The following multiple-machine configuration is recommended: Siebel Server, BizTalk Server, and Siebel Clients. However, your setup may be different and your configuration may vary from the suggested setup accordingly.

■ Siebel Server

- Install on your preferred platform:
 - Microsoft Windows NT, Windows 2000 Server, or Windows 2000 Advanced Server
- Microsoft Internet Information Server
- Siebel Server
- Siebel Web Engine (SWE)
- Siebel Gateway Server
- SQL Server (Siebel DB)
- Siebel EAI Transports

■ BizTalk Server**■ Microsoft BizTalk Server 2000 and 2002 requirements:**

- BizTalk Server
- Microsoft Windows 2000 Server, or Windows 2000 Advanced Server
- Windows 2000
- SQL Server or SQL Server 2000
- Microsoft Visio 2000
- Microsoft Internet Explorer

■ Siebel Clients

- Microsoft Windows NT or Windows 2000
- Microsoft Internet Explorer
- Siebel Mobile Web Client
- Siebel Tools

Siebel Integration Objects

Before you can exchange documents with any of your trading partners or with any other application using BizTalk Server, you need to expose the appropriate Siebel integration objects to BizTalk; this is done by creating integration object *schema* which you then import into BizTalk.

Siebel integration objects are published using the Schema Generator Wizard—a component of Siebel Tools. You need to create a new document definition using the Schema Wizard to expose any of the Siebel integration object formats for use with BizTalk.

You export either DTD or XDR schema. These exported files are imported into BizTalk where they are transformed into *document specifications*. BizTalk Server adds BizTalk-specific XML tags that convert the standard XDR format to BizTalk's document specification format. [Figure 6](#) depicts how the Siebel integration objects are exposed to BizTalk Server.

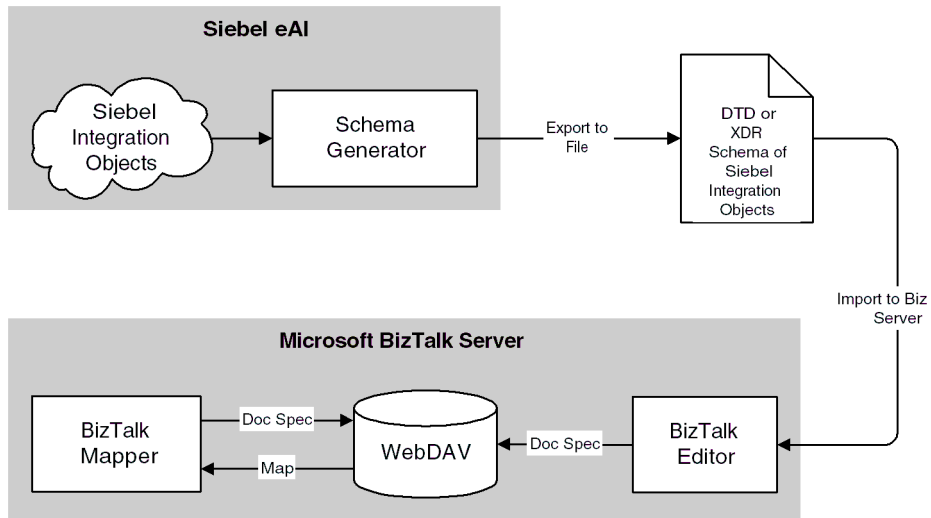


Figure 6. Exposing Siebel Integration Objects to BizTalk Server

Document specifications are stored in the WebDAV BizTalk Repository and are available to any client that can connect to WebDAV. You can retrieve stored document specifications from WebDAV and create maps to perform data transformations from within BizTalk Mapper. BizTalk maps, which are XSLT (Extensible Stylesheet Language)-based, are also stored in WebDAV.

NOTE: To use the Siebel BizTalk Server interface, you need to have integration objects defined for each trading partner. You can either use existing integration objects for this purpose, or you can create new ones depending upon your business requirements.

NOTE: For instructions on how to create and modify Siebel integration objects, see *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition*.

Exposing Integration Objects to BizTalk

Checklist

- ☐ Generate an DTD or XDR schema for each integration object you want to expose to BizTalk (repeat for each integration object you will need).

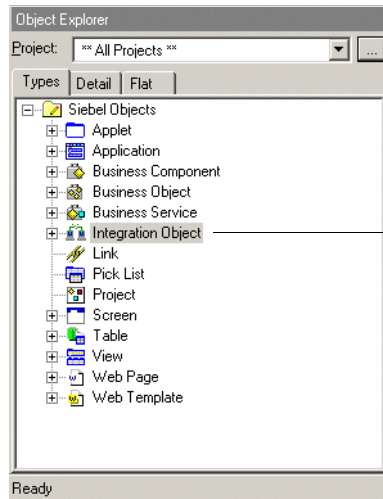
For details, see [“To generate a new DTD or XDR schema” on page 134](#).

- ☐ Import the generated schema for the integration object into BizTalk Server, where they are converted to document specifications.

For details, see [“To import schema into BizTalk Server to save it as a BizTalk document specification” on page 136](#).

To generate a new DTD or XDR schema

- 1 Open Siebel Tools on the Siebel client.
- 2 Select Integration Object in the Objects Explorer window as shown in the following figure.



In the Types tab, click here to display the list of Integration Objects

- 3 Select the integration object that you want to expose and click the Generate Schema button at the top of the window as shown in the following figure.

Integration Objects

Synchronize Generate Schema

Click the Generate Schema button after you select the Integration Object

Name	Changed	Project	Base Object Type	Business Object	External
SAP IDOC Wizard - Get IDOC List (BA)		SAP Design	SAP BAPI Input	IDOC_I	IDOC_I
SAP IDOC Wizard - Get IDOC List (BA)		SAP Design	SAP BAPI Output	IDOC_O	IDOC_O
SAP IDOC Wizard - Get IDOC Metada		SAP Design	SAP BAPI Input	EDI_IDI	EDI_IDI
SAP IDOC Wizard - Get IDOC Metada		SAP Design	SAP BAPI Output	EDI_IDO	EDI_IDO
SAP IDOC Wizard - Get IDOC Segmei		SAP Design	SAP BAPI Input	EDI_SEI	EDI_SEI
SAP IDOC Wizard - Get IDOC Segmei		SAP Design	SAP BAPI Output	EDI_SEO	EDI_SEO
SAP IDOC Wizard - Get IDOC Segmei		SAP Design	SAP BAPI Input	EDI_SEI	EDI_SEI
SAP IDOC Wizard - Get IDOC Segmei		SAP Design	SAP BAPI Output	EDI_SEO	EDI_SEO
SAP IDOC Wizard - Get IDOC Struct.		SAP Design	SAP BAPI Input	IDOC_I	IDOC_I
SAP IDOC Wizard - Get IDOC Struct.		SAP Design	SAP BAPI Output	IDOC_O	IDOC_O
SAP IDOC Wizard - Get RFC Table En		SAP Design	SAP BAPI Input	TABLE_I	TABLE_I
SAP IDOC Wizard - Get RFC Table En		SAP Design	SAP BAPI Output	TABLE_O	TABLE_O
SAP IDOC Wizard - Get Record Strud		SAP Design	SAP BAPI Input	IDOC_F	IDOC_F
SAP IDOC Wizard - Get Record Strud		SAP Design	SAP BAPI Output	IDOC_F	IDOC_F
SAP RFC - Execute ABAP (BAPI Inpu		SAP Design	SAP BAPI Input	RFC_AE	RFC_AE
SAP RFC - Execute ABAP (BAPI Outp		SAP Design	SAP BAPI Output	RFC_AE	RFC_AE
SAP Wizards - Get Field Info (BAPI In		SAP Design	SAP BAPI Input	DDIF_F	DDIF_F
SAP Wizards - Get Field Info (BAPI In		SAP Design	SAP BAPI Input	DDIF_F	DDIF_F
SAP Wizards - Get Field Info (BAPI O		SAP Design	SAP BAPI Output	DDIF_F	DDIF_F
SAP Wizards - Get Field Info (BAPI O		SAP Design	SAP BAPI Output	DDIF_F	DDIF_F
Sample Account		EAI Test	Siebel Business Object	Account	Account
Sample Account LIV		EAI Test	Siebel Business Object	Account	Account
Sample Contact		EAI Test	Siebel Business Object	Contact	Contact
Sample Employee		EAI Test	Siebel Business Object	Employee	Employee
Sample Order		EAI Test	Siebel Business Object	Order	Order
Sample Quote		EAI Test	Siebel Business Object	Quote	Quote

Select an Integration Object; here "SampleOrder" is selected

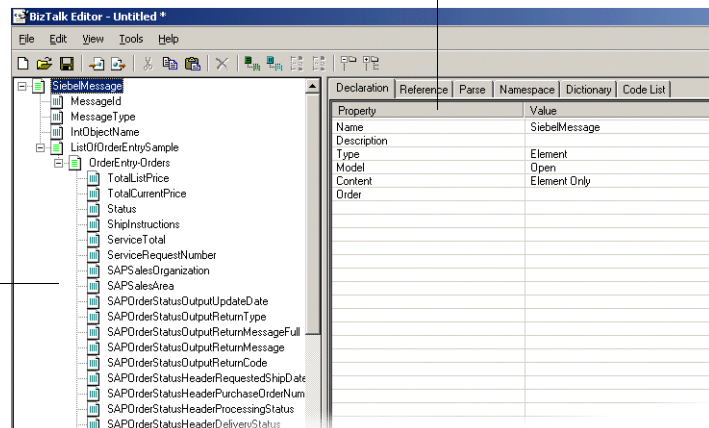
- 4 The Generate Schema Wizard appears.
 - a Select "EAI XML XDR Generator" from the Business Service drop-down list.
 - b Select "EAI Siebel Message Envelope Service" from the Envelope drop down list.
 - c Browse to a file location and type a file name to generate the schema, for example, "ListOfSiebelOrder.xml," and click Save.
- 5 Click Finish.

A file (named for example, "ListOfSiebelOrder.xml.") has now been created in the file location you specified.

- 6 Copy the newly generated schema from the Siebel Client machine to the BizTalk Server machine.

- 1** Open the BizTalk Server Schema Editor on the BizTalk Server machine.
- 2** Choose Tools > Import.
- 3** Select XDR Schema and click OK.
- 4** Browse to the location where you stored the XDR Schema and click Open.

Properties, values, and other
information appears here.



- NOTE:** The XDR schema is now saved in the BizTalk Documentation Specification format on WebDay.

Data Types

Standard Siebel applications come with predefined data types, which are mapped/converted to XDR-specific data types.

[Table 16](#) lists data types for the Siebel Field object type and the current corresponding mapping to XDR data types.

Table 16. Siebel Data Type Mappings in BizTalk

Siebel Data Type	XDR Data Types
DTYPE_TEXT	string
DTYPE_BOOL	char (1)
DTYPE_CURRENCY	number
DTYPE_DATE	string
DTYPE_DATETIME	string
DTYPE_TIME	string
DTYPE_ID	string
DTYPE_INTEGER	int
DTYPE_NOTE	string
DTYPE_NUMBER	number
DTYPE_PHONE	number

Connecting to BizTalk Using MSMQ

This section describes how to set up and use the MSMQ transport to send and receive messages to and from the BizTalk Server.

The BizTalk Server Adapter uses the Siebel EAI MSMQ Transport to exchange messages with the BizTalk Server over MSMQ. See [Chapter 3, “EAI MSMQ Transport,”](#) for specific details about that transport.

The following sections show examples of how MSMQ can be used with BizTalk.

Siebel MSMQ Outbound Transport

This section describes how to set up and use the MSMQ Transport to send messages to BizTalk Server.

Sending Messages to BizTalk Using the EAI MSMQ Transport

You can use the Siebel BizTalk interface to send an XML document to BizTalk Server over MSMQ transport.

Checklist

- ☐ Set up the Siebel EAI MSMQ Transport and setup queue to send messages, if you have not already done so.

For details, see [“Configuring EAI MSMQ Transport for Various Send and Receive Scenarios” on page 38.](#)

- ☐ Set up the organization, port, and channel (collectively known as BizTalk configuration objects) for use with the MSMQ Transport.

For details, see [“To set up BizTalk configuration objects for MSMQ Outbound” on page 139.](#)

- ☐ Create the workflow to process outbound documents from Siebel using MSMQ transport.

For details, see [“Sending Outbound Messages with MSMQ” on page 40.](#)

Setting Up an MSMQ Queue for Sending Messages

Set up an MSMQ transactional queue to receive messages from the Siebel application. You should name the queue an easy to identify name, such as "FromSiebel." See the EAI MSMQ Transport chapter and the Microsoft MSMQ documentation for details.

To set up BizTalk configuration objects for MSMQ Outbound

- 1** On the BizTalk Server machine, access the BizTalk Messaging Manager.
- 2** Set up your home organization as Siebel and set up applications for the home organization.

NOTE: For each BizTalk-specific step, see Microsoft BizTalk Server documentation for the details.

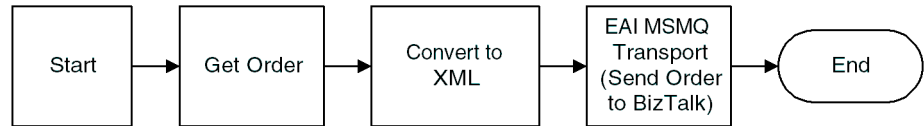
- 3** Create a new organization and name it appropriately. For example, if your trading partner is Microsoft, enter "Microsoft."
- 4** Create a document definition for the previously-created document specification from WebDAV and name it, for example, Siebel Order. See [To import schema into BizTalk Server to save it as a BizTalk document specification on page 136](#).
- 5** Create a messaging port to your trading partner.
 - a** In the Destination Organization window, select "Microsoft," for example, as the Organization.
 - b** Click Browse under Primary Transport.
 - c** Select the appropriate transport for your trading partner.
 - d** Complete the remaining pages as needed and click Finish to complete.
- 6** Create a new channel for the port created [Step 5](#).
 - a** Create a new channel "From an Application," named, for example, "Siebel To Microsoft Channel."
 - b** Click Next.

- c** Select the appropriate inbound document definition, for example, “Siebel Order.”
 - d** Select appropriate outbound document definition for your trading partner, for example, “Microsoft Order.”
 - e** Select an appropriate map, if necessary.
 - f** Complete other pages as required and select Finish to complete.
- 7** Create a Message Queuing Receive Function in the BizTalk Server Administration (for example, “Siebel MSMQ Receive”) to specify the queue (for example, “FromSiebel”) that the receive function polls. This queue is the same queue that Siebel uses to send the message using EAI MSMQ Transport.

To create the Siebel workflow for MSMQ Outbound

- 1 On the Siebel machine, access the Workflow Process Designer.

Set up a workflow for sending a message to BizTalk so that it follows the flow illustrated in this flowchart.



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 2 Define the XML conversion. An example is provided in the following table.

Business Service

Business Service	XML Converter
Method	Integration Object Hierarchy to XML Document

Input Argument

Input Argument	XML Character Encoding
Value	UTF-16

- 3 For the EAI MSMQ Transport (Send to BizTalk) business service (Step 4 in the workflow), double-click the object icon to open the Business Service form.

- 4 In the Business Service form, enter the information from the following table:

Field Name	Value
MsmqPhysicalQueueName	Physical Queue Name
MsmqQueueMachineName	Machine Name that owns the MSMQ queue
LargeMessageSupport	False
Value	XML Data that you to send

NOTE: XML documents must be sent in UTF-16 format for BizTalk Server to parse. In order for message queuing to work properly with BizTalk, you should disable large message support as shown above.

- 5 Save your workflow.

NOTE: See [Chapter 3, “EAI MSMQ Transport”](#) for more information on setting up and using the Siebel EAI MSMQ Transport.

Siebel MSMQ Inbound Transport

This section describes how to use the Siebel BizTalk interface to receive an XML document from BizTalk Server sent over the MSMQ Transport.

Receiving Messages to BizTalk Using the EAI MSMQ Transport

This section explains how to use the Siebel BizTalk interface to receive an XML document from BizTalk Server over the MSMQ transport.

Checklist

- ☐ Set up the Siebel EAI MSMQ Transport and setup queue to receive messages, if you have not already done so.

For details, see [“Configuring EAI MSMQ Transport for Various Send and Receive Scenarios”](#) on page 38.

- ❑ Set up the organization, port, and channel (collectively known as BizTalk configuration objects) for use with the MSMQ Transport.

For details, see [“To set up BizTalk configuration objects for MSMQ Inbound.”](#)
- ❑ Create the workflow to process inbound documents from Siebel using MSMQ transport.

For details, see [“Receiving Messages from MSMQ” on page 47.](#)

Setting Up an MSMQ Queue for Receiving Messages

Set up an MSMQ transactional queue to receive messages from the Siebel application. You should name the queue an easy to identify name, such as “ToSiebel.” See the EAI MSMQ Transport chapter and the Microsoft MSMQ documentation for details.

To set up BizTalk configuration objects for MSMQ Inbound

- 1** On the BizTalk Server machine, access the BizTalk Messaging Manager.
- 2** Set up your home organization as Siebel and set up applications for the home organization.

NOTE: For each BizTalk-specific step, see Microsoft BizTalk Server documentation for the details.

- 3** Create a new organization and name it as is appropriate. For example, if your trading partner is Microsoft, enter “Microsoft.”
- 4** Create a document definition for previously-created document specification from WebDAV and name it, for example, Siebel Order. See [To import schema into BizTalk Server to save it as a BizTalk document specification on page 136.](#)
- 5** Set up a new BizTalk Server port to send the document to the Siebel application (for example, Siebel Sales) with the Primary Transport of Message Queuing.
 - a** Using BizTalk Messaging Manager, create a new port named Siebel MSMQ BTS.
 - b** Specify the primary transport type to be Message Queuing.

- c** Specify the address to point to the queue on which you will be sending the message to. For example, DIRECT = OS: < machine > \ToSiebel. Refer to the BizTalk documentation for queue supported format names.
 - d** Save your work.
- 6** Create a new channel for the port created [Step 5](#).
 - a** Create a new channel “From an Organization,” named, for example, “Oracle To Siebel MSMQ Channel.”
 - b** Click Next.
 - c** Select the appropriate inbound document definition, for example, “Oracle Order.”
 - d** Select appropriate outbound document definition for your trading partner, for example, “Siebel Order.”
 - e** Select an appropriate map, if necessary.
 - f** If required, configure the Advanced properties in the final Channel Configuration page to provide authentication information.
 - g** Complete other pages as required and select Finish to complete.
- 7** Configure a File Receive Function from BizTalk Server Administration to poll a file location and deliver to the channel configured in [Step 6](#).

NOTE: This example uses File transport to receive documents from trading partners for the sake of illustration. In your actual business situations, a trading partner can deliver messages to a Siebel application over *any* supported transport.

- a** Open BizTalk Server Administration, expand Microsoft BizTalk Server, and expand the server group to which you want to add the File receive function.
- b** Select Receive Functions.
- c** Choose Action > New > File Receive Function.
- d** The Add a File Receive Function dialog box appears.

- e** In the Name box, type the name of the File receive function, for example, “Oracle to Siebel MSMQ.”
- f** In the Comment box, add a brief description (optional).
- g** In the Server on which the receive function will run list, click the name of a server in the group.
- h** For the Polling Location, enter `c:\MSMQ`.

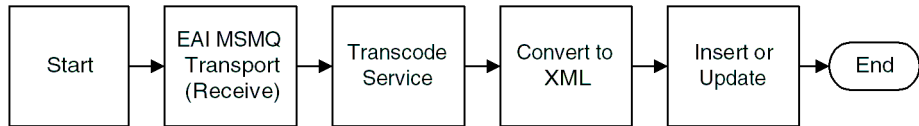
NOTE: This is the location where your trading partner delivers the XML documents for Siebel so the channel can pick it up from there and send it to Siebel application for processing.

- i** In the File types to poll for box, type the extension of the files that BizTalk Server receives. In this case, type `*.xml`.
- j** Click the Advanced tab and enter as the Channel Name as “Oracle to Siebel MSMQ Channel,” for example.
- k** Click OK to finish.

To create the Siebel workflow for MSMQ Inbound

- 1 On the Siebel machine, access the Workflow Process Designer.

Set up a workflow for BizTalk to receive a message so that it follows the flow shown in this flowchart.



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 2 For the EAI MSMQ Transport (Receive) business service (Step 2 in the workflow), double-click the object icon to open the Business Service form.

- 3** In the Business Service form, enter the values for each field, as shown on the following table.

Field Name	Value
MsmqPhysicalQueueName	Physical Queue Name
MsmqQueueMachineName	Machine Name that owns the MSMQ queue
LargeMessageSupport	False
IgnoreCorrelation	True

NOTE: XML documents are sent in UTF-16 format from BizTalk Server to Siebel. In order for the XML Converter to correctly process, the incoming document must be converted to UTF-8 format through the Transcode Service (Step 3 in the workflow). In addition, Large Message Support must be disabled and Ignore Correlation Id must be set to TRUE in the EAI MSMQ Transport step for Siebel to pick up messages from BizTalk. Correlation Id must be ignored because BizTalk always populates the information on the sent messages. The messages with Correlation Id are considered response messages by Siebel applications.

- 4** Save your workflow.

NOTE: See [Chapter 3, “EAI MSMQ Transport”](#) for more information on setting up and using the Siebel EAI MSMQ Transport.

Connecting to BizTalk Using COM/AIC

This section describes how to set up and use the COM Outbound Transport to send messages to BizTalk Server, and how to use the AIC Inbound Transport to receive messages from BizTalk.

Siebel COM Outbound Transport

This section describes how to set up and use the COM Outbound Transport to send messages to BizTalk Server.

Setting Up the COM Outbound Transport

To use the Siebel application to send documents to BizTalk over COM, you first need to install the BizTalk Server Interchange Application COM + package from the BizTalk machine to the Siebel Client. This is a one-time operation.

Checklist

- ☐ Complete prerequisites, if you have not already done so (for setting up BizTalk Server and Siebel Clients and generating schema from integration objects).

For details, see [“Preparing to Use the Siebel BizTalk Adapter” on page 129](#).

- ☐ Install the Interchange Application COM + package onto the Siebel Client machine from the BizTalk Server machine (one time operation).

For details, see [“To establish a COM + remote communication link with BizTalk” on page 149](#).

To establish a COM+ remote communication link with BizTalk

- 1** On the BizTalk machine, choose Start > Programs > Administrative Tools > Component Services.
- 2** In the tree pane, click Component Services, expand Computers and then under Computers, expand My Computer.
- 3** Expand COM+ Applications and select BizTalk Server Interchange Application.
- 4** Right click and, from the pop-up menu that appears, select Export.
- 5** On the Welcome to COM Application Export Wizard, click Next.
- 6** Enter the name of the export application to be created, for example, "BiztalkInterchange." Use the Browse button as needed.
- 7** In the Export As area, select the Application proxy radio button and click Next.
- 8** On the final page, click Finish.

You should now have two files, named as you specified in [Step 6](#), above. One should have the extension ".MSI," and one should have the extension ".cab," for example, BiztalkInterchange.MSI and BizTalkInterchange.MSI.cab.

- 9** Copy these files to the Siebel Client and then, on the Siebel Client, run the *.MSI file to install the remote client for BizTalk Server.

Repeat the steps above for each Siebel Client that will need to communicate with BizTalk Server.

Using the COM Outbound Transport to Send Messages to BizTalk

This section explains how to use the COM outbound transport to send messages from Siebel application to a trading partner or external application through BizTalk Server. COM Outbound Transport is implemented as a Business Service, so you could potentially use this service as you would use any other Business Service, such as calling it from Workflow for example.

You would run the workflow as fits your needs, using any of the usual mechanisms for running workflows in Siebel application: Workflow Process, eScripting, and so on.

Checklist

- ☐ Set up the Siebel COM Outbound Transport, if you have not done so already.

For details, see [“Setting Up the COM Outbound Transport” on page 148](#).

- ☐ Create configuration objects in BizTalk, by setting up new organizations, ports, and channels (once for each trading partner).

For details, see [“To set up BizTalk configuration objects” on page 152](#), and [“COM Outbound Transport Parameters” on page 151](#).

- ☐ Create a new workflow in Siebel application to transform data in a Siebel application and send as messages to BizTalk.

For details, see [“To create the Siebel workflow for COM Outbound” on page 154](#).

COM Outbound Transport Parameters

When defining the workflow for COM Outbound transport to BizTalk, you can optionally choose from a number of parameters. These parameters correspond with the BizTalk Server `IInterchange.Submit()` and `SubmitSync()` parameters.

Table 17 lists the outbound parameters.

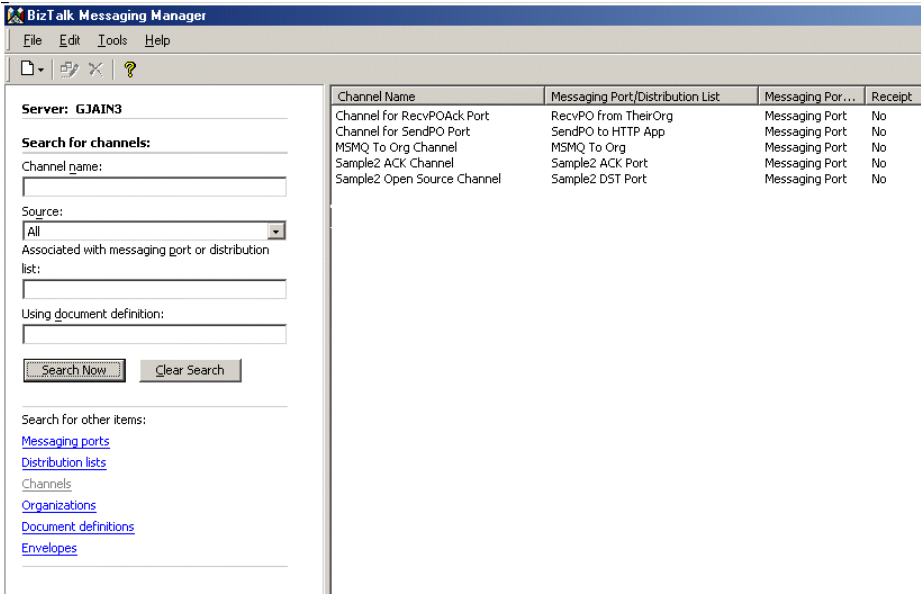
Table 17. COM Outbound Parameters

Parameter	Type	Description
< Value >	String	The document instance submitted.
DocumentName	String	The name of the BizTalkDocument object associated with the instance of the document being submitted.
SourceQualifier	String	The qualifier of the source organization and indicates how the <i>Source</i> is to be interpreted.
Source	String	The value of the qualifier of the source organization.
DestinationQualifier	String	The qualifier of the destination organization, it indicates how the DestinationID parameter is to be interpreted.
DestinationID	String	The value of the qualifier of the destination organization.
Channel	String	Contains the name of the BizTalkChannel object that is executed for this document.
FileName	String	Specifies a fully qualified path that contains the document to be submitted, rather than submitting the document directly as a string.
Envelope	String	Contains the name of the envelope specification to use to break the interchange into documents.

NOTE: For more details on these parameters, see your Microsoft BizTalk Server documentation.

To set up BizTalk configuration objects

- 1** On the BizTalk Server machine, access the BizTalk Messaging Manager.



NOTE: For details on each BizTalk-specific step, see the Microsoft BizTalk Server documentation and help system.

- 2** Set up your home organization as Siebel and then set up applications for the home organization.

- 3** Create a new organization and name it appropriately, such as “Microsoft.”

NOTE: Throughout this chapter, many steps are illustrated with the use of an example. This example uses “Microsoft,” but your actual business requirements will dictate the name you give to the actual destination organizations that you set up for *your* trading partners.

- 4** Create a document *definition* for the previously-created document *specification* in WebDAV and name it, for example, Siebel Order. See [“To import schema into BizTalk Server to save it as a BizTalk document specification” on page 136.](#)
- 5** Create a messaging port to your trading partner.
 - a** In the Destination Organization window, select “Microsoft,” for example, as the Organization.
 - b** Click Browse under Primary Transport.
 - c** Select the appropriate transport for your trading partner.
 - d** Complete the remaining pages as needed and click Finish to complete.
- 6** Create a new channel for the port created [Step 5.](#)
 - a** Create a new channel “From an Application,” named, for example, “Siebel To Microsoft Channel.”
 - b** Click Next.
 - c** Select the appropriate inbound document definition, for example, “Siebel Order.”
 - d** Select appropriate outbound document definition for your trading partner, for example, “Microsoft Order.”
 - e** Select an appropriate map, if necessary.
 - f** Complete other pages as required and select Finish to complete.

To create the Siebel workflow for COM Outbound

- 1 On the Siebel client, access the Workflow Process Designer to set up a workflow.
- 2 Set up a workflow for sending a message to BizTalk. Define the flow according to the following flowchart.



NOTE: For information on how to use the Siebel Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 3 Define the XML conversion. For example:
 - Business Service: XML Converter
 - Method: Integration Object Hierarchy to XML Document
 - Input Argument: XML Character Encoding
 - Value: UTF-16

- 4 To define the BizTalk COM object (the fourth object in the preceding workflow), double-click the object icon to open the Business Service form. In the Business Service form, enter the following:

Field Name	Value
Name	Send Order to BizTalk Over COM
Business Service	EAI BTS COM Transport
Method	Send SendReceive

NOTE: You may require asynchronous or synchronous communication, using Send and SendReceive respectively, depending on your unique situation.

- a Save your work so far by clicking Save.
- b Click the Input Arguments form and set two (2) parameters according to the values provided in the following table.

Field Name	Value	Description	Example
Input Argument	< Value >	Data string that you want to send to BizTalk. Typically, this corresponds to the name of the workflow process property that stores the actual data.	“Order XML”
Input Argument	Channel	Specify the channel you created in “To set up BizTalk configuration objects” on page 152 .	“Siebel to Microsoft Channel”

NOTE: Optionally, you can set any one of the other available input parameters in the pick list, such as “Destination,” “DocumentName,” and so on. For the full list of these parameters and their meanings, refer to [Table 17 on page 151](#).

The Input Arguments form will look similar to the sample shown in the following illustration.

The Input Arguments you enter appear here

All Processes Process Designer Process Properties Process Simulator			
New Query Last 2 Records			
	Input Argument	Type	Value
◆	<input type="text" value="<Value>"/> <input checked="" type="checkbox"/>	<input type="text" value="Literal"/> ...	<input type="text" value="ORDER.XML"/>
◇	Channel	Literal	Siebel To Oracle Channel

- 5 Make other modifications to your workflow as your situation requires and save the workflow.

Siebel AIC Inbound Transport

This section describes how to set up and use the AIC Inbound Transport to receive messages from BizTalk.

Setting up the AIC Inbound Transport

AIC can be run either on the Siebel Server machine or on the BizTalk Server machine. Generally, AIC is run on the BizTalk Server machine.

NOTE: AIC (Application Integration Components) and ActiveX Data Controls must both be installed on the *same* machine, regardless of your configuration setup.

This section explains how to set up the AIC Inbound Transport on either BizTalk Server with IIS 5.0 installed or a Siebel Server machine and covers these topic areas:

- [“Setting Up AIC Inbound Transport to Run Locally on BizTalk Server” on page 157](#)
- [“Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server” on page 161](#)

Setting Up AIC Inbound Transport to Run Locally on BizTalk Server

This section explains how to set up the required components for running the AIC Inbound Transport on the BizTalk Server machine.

Checklist

- ☐ Install the necessary Siebel EAI ActiveX Data Controls and Siebel Application Integration Components on the BizTalk Server machine (one time operation).

For details, see [“To install Siebel ActiveX Data Controls and AIC on the BizTalk Server machine” on page 158](#).

- ☐ Register the Siebel AIC on the BizTalk Server machine (one time operation).

For details, see [“To register AIC as a COM + Server Application on the BizTalk Server machine” on page 158](#).

- ❑ Copy the required Siebel Active Server Pages (ASP) to the BizTalk Server machine.

For details, see [“To configure BizTalk Server Messaging Manager with new ASP files” on page 164.](#)

- ❑ Make sure that the Siebel machine has the capability for running workflows—BizTalk Server will be sending named workflows to Siebel expecting them to run.

For details, see [“To configure Siebel Server to run Workflow” on page 164.](#)

To install Siebel ActiveX Data Controls and AIC on the BizTalk Server machine

- 1** On the BizTalk machine, insert the Siebel Enterprise Server Installation CD and choose Custom Install.
- 2** Deselect other options so that only the EAI Connectors will be installed.
- 3** Follow online prompts to complete the remaining pages and click Finish to install the Siebel EAI ActiveX Data Controls for BizTalk on the BizTalk Server machine.

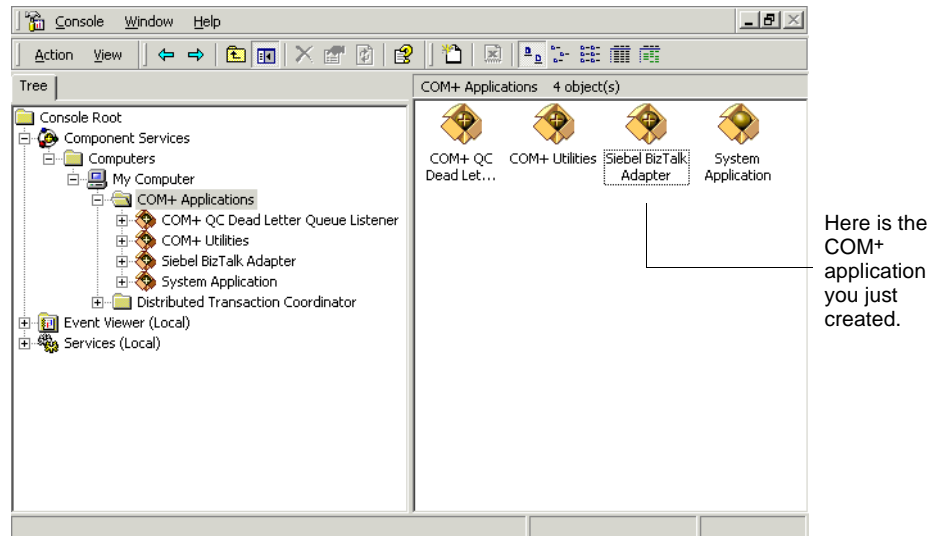
To register AIC as a COM+ Server Application on the BizTalk Server machine

NOTE: COM + application functionality is available only with Windows 2000.

- 1** From the Start menu, select Programs > Administrative Tools > Component Services.
- 2** In the tree pane, click Component Services, and expand Computers. Under Computers, expand My Computer, and then click on COM + Applications.
- 3** On the Action menu, click New > Application.
- 4** On the Welcome to COM Application Install Wizard dialog box, click Next.
- 5** On the next page, select Create an Empty Application.
 - Enter the name for the new application to be created, for example, “Siebel BizTalk Interface” for AIC Transport or “Siebel BizTalk HTTP Adapter” for the HTTP Inbound Transport.
 - Select the Activation Type of “Server application” and click Next.

- 6 On the next screen, select the Application Identity of “Interactive user - the current logged on user.” Click Next and then click Finish.

You should now have a new COM + application (which you named in [Step 5](#)), as shown in the following sample figure.



- 7 Expand this newly created COM + Server Application by double-clicking on it. You will see folders appear where the COM + applications were.
- 8 Click the Components folder and choose Action > New > Component. The Welcome to COM Component Install Wizard appears.
- 9 On the Welcome to COM Component Install Wizard, click Next.
 - a Select Install new component or components and click Next.
 - b On the next page, click Add.

- 10** Browse to the *< Siebel Installation directory > \eaiconn\bts\aic* directory and select the AIC component, *sscaeiba.dll*.

You should see BizTalk.SiebelBizTalkAIC component registered and you should be able to see the AIC interfaces and methods.

Or, for the HTTP Inbound Transport, browse to the *c:\AIC* directory (the directory where you originally copied your dll from the Siebel Server machine) and select the AIC component, *sscaeibh.dll*.

You should see Siebel.BizTalkHTTPAIC component registered and you should be able to see the AIC interfaces and methods.

Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server

This section explains how to set up the required components for running the AIC Inbound Transport remotely on the Siebel Server machine.

Checklist

- ☐ Copy required BizTalk Dynamic Link Library (DLL) and Type Library (TLB) files from the BizTalk Server to the Siebel Server.

For details, see [“To copy BizTalk Server DLL and Type Library files to the Siebel Server” on page 162.](#)

- ☐ Register the Siebel BizTalk AIC as a COM + Server Application on the Siebel Server machine.

For details, see [“To register AIC as a COM + package remote communication” on page 163.](#)

- ☐ Create a COM + communications link between the Siebel machine and the BizTalk Server machine.

For details, see [“To create a COM + package remote communication on the Siebel machine” on page 163.](#)

- ☐ Copy the required Siebel Active Server Pages (ASP) to the BizTalk Server machine.

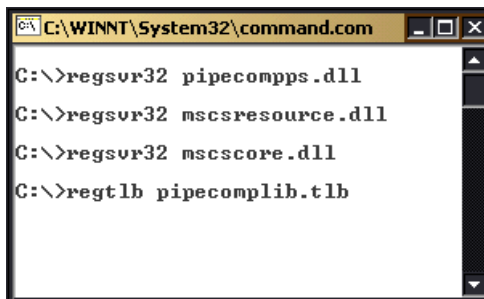
For details, see [“To configure BizTalk Server Messaging Manager with new ASP files” on page 164.](#)

- ☐ Make sure that the Siebel machine has the capability for running workflows—BizTalk Server will be sending named workflows to Siebel application expecting them to run.

For details, see [“To configure Siebel Server to run Workflow” on page 164.](#)

To copy BizTalk Server DLL and Type Library files to the Siebel Server

- 1** Copy dependency shared DLLs stored in “%Program Files%\Common Files\Microsoft Shared\Enterprise Servers\Commerce” from BizTalk (Commerce server shared DLL) to Siebel Server Machine:
 - pipecomlib.tlb
 - pipecompps.dll
 - mscsresource.dll
 - mscscore.dll
- 2** Open a DOS command prompt window. On the Siebel Server machine:
 - Register the DLLs.
 - Register the type library by using the regsvr32 and regtlb utilities, respectively, as shown in the following illustration.



```
C:\>regsvr32 pipecompps.dll
C:\>regsvr32 mscsresource.dll
C:\>regsvr32 mscscore.dll
C:\>regtlb pipecomplib.tlb
```

Enter the commands one at a time, as shown, and press Enter. As each file is registered, additional information displays in the DOS window.

NOTE: The regtlb utility ships with BizTalk Server and is available in the <Microsoft BizTalk Server>\Setup directory.

To register AIC as a COM+ package remote communication

- Register AIC as a COM+ Server Application on the Siebel Server machine.

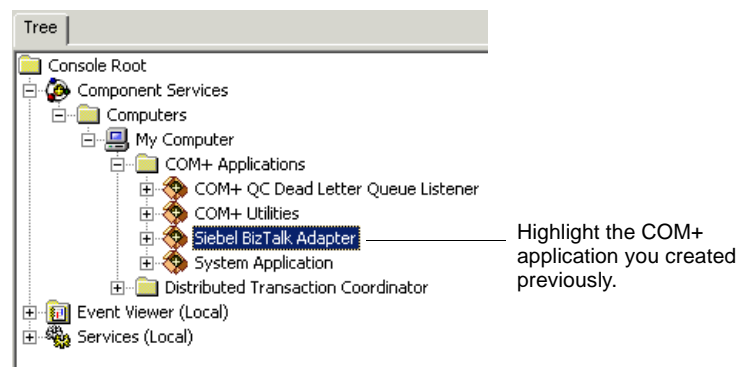
Steps to register AIC are the same as described above in the previous section, [“To register AIC as a COM+ Server Application on the BizTalk Server machine” on page 158](#).

NOTE: Siebel AIC (Application Integration Components) and ADC (ActiveX Data Controls) are installed as a part of the Siebel Server or the EAI Connectors installation.

To create a COM+ package remote communication on the Siebel machine

Here you create a COM+ package for the remote communication on the Siebel machine and run on the BizTalk Server machine as Application Proxy.

- 1 On the Siebel machine, go to the Start menu. Select Programs > Administrative Tools > Component Services.
- 2 In the tree pane, click Component Services, and expand Computers. Under Computers, expand My Computer, and then expand COM+ Applications. Click “Siebel BizTalk Adapter” (the COM+ Application you created previously—you may have given it a different name). A sample tree pane is shown in the following illustration.



- 3 On the Action menu, click Export.

- 4** On the Welcome to COM Application Export Wizard dialog, click Next.
- 5** Enter the name of the export installation package to be created.
- 6** In the Export As area, select the Application proxy radio button and click Next.
- 7** Click Next to finish the wizard. You should see a “Thank you” page. Click Finish.

You should now have two files, named as you specified in [Step 5](#). The files will have two extensions: .MSI and .cab.

- 8** Copy these files to the BizTalk Server machine.
- 9** Run the .MSI file to install the remote client for the Siebel AIC that is on the BizTalk Server machine.

To configure BizTalk Server Messaging Manager with new ASP files

- Copy the BizTalk_SiebelBizTalkAIC_1_post.asp page and the BizTalk_SiebelBizTalkAIC_1.asp page,

from Siebel Machine:

```
<Siebel installation dir>\eaiconn\bts\scripts
```

to BizTalk Machine:

```
<BizTalk installation directory>\MessagingManager\pipeline
```

To configure Siebel Server to run Workflow

The Siebel Application Integration Components receive documents from BizTalk Server and deliver them to the Siebel Workflow specified in the BizTalk Server channel configuration.

AIC also invokes the Siebel Workflow Engine to run the workflow on either the Siebel Client or the Siebel Server using ActiveX Data Controls.

- *AppObjMgr_Lang* is the Application Object Manager to which AIC connects through ADC to run Workflows on Siebel Server. Therefore, make sure that the Siebel Server Workflow component group and the appropriate Siebel Applications *AppObjMgr_Lang* (for example, *SCCObjMgr_enu*, for Siebel Call Center MidMarket Edition) components are installed and correctly configured.

NOTE: See *Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition* and *Siebel Tools Online Help, MidMarket Edition* for details on enabling server components and connecting to Application Object Manager using ADC.

Using the AIC Inbound Transport to Receive Messages from BizTalk

This section explains how to receive messages from BizTalk using the AIC Inbound transport.

NOTE: Microsoft BizTalk Server must be installed. In addition, the appropriate Siebel Applications AppObjMgr_Lang (for example, SCCObjMgr_enu) component, Siebel Workflow, and Siebel eBusiness Application Integration (EAI) must be enabled prior to following the procedure below. See *Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition* for more information.

Checklist

- ☐ Set up the Siebel AIC Inbound Transport, if you have not already done so.
For details, see [“Setting up the AIC Inbound Transport” on page 157](#).
- ☐ Create the workflow to process inbound documents from BizTalk using AIC.
For details, see [“To create a workflow to receive inbound XML documents over AIC from BizTalk.”](#)
- ☐ After you have created the workflow, you need to supply the workflow information to the Siebel AIC Inbound transport—a pipeline component for BizTalk.
For details, see [“To supply the workflow information for AIC Inbound” on page 170](#).

To create a workflow to receive inbound XML documents over AIC from BizTalk

For AIC inbound connections, you define a workflow to process the inbound documents passed by the BizTalk Server using the Siebel Application Integration Component (AIC). This workflow performs the necessary steps to process the document, depending on the business requirements. For example, it could read the document, perform conversions, update the database, call other business services (such as the Siebel Adapter), and so on. This process gives you the immense flexibility in the way an inbound XML document is processed.

- 1** Run Siebel Mobile Client to set up a new workflow to receive and process XML document from AIC.

For information on defining and maintaining Siebel workflows, refer to *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

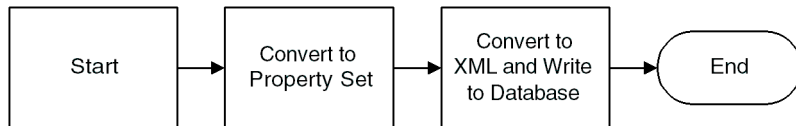
- 2** Create a workflow using Siebel Workflow Process Designer.

- a** Give the workflow a name, for example, “Write XML Doc.”

You might want to write this name down, as this is the name you enter later in [Step 3 on page 171](#) in “[To supply the workflow information for AIC Inbound](#)” when you create the AIC connection to the workflow process in BizTalk Server.

- b** Set up the workflow so that it follows this flow (use Start, two business objects, and End).

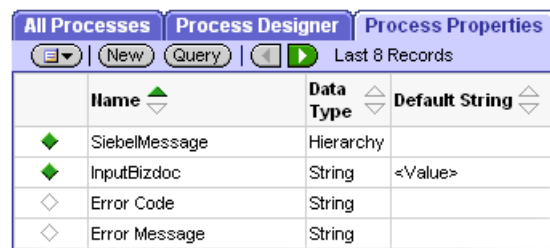
The following example of the workflow is for illustration purposes. In reality, you would define a workflow that actually meets your business needs.



- 3 Create two (2) new Process Properties that you will need to set up the workflow, according to the information in the following table:

Name	Data Type	Default String
InputBizdoc	String	< Value >
SiebelMessage	Hierarchy	

NOTE: AIC passes the XML Document received from BizTalk Server to this process property and it can be used to retrieve and process the XML document. For example, write this process property to a file or update the Siebel database by calling Siebel Adapter.



	Name	Data Type	Default String
◆	SiebelMessage	Hierarchy	
◆	InputBizdoc	String	<Value>
◇	Error Code	String	
◇	Error Message	String	

The Process Properties you just created now appear in the list of process properties

- 4 For the “Convert to Property Set” business service (Step 2 in the workflow), double-click the object icon to open the Business Service form.

- a In the Business Service form, enter the information from the following table:

Field Name	Value
Business Service	EAI XML Converter
Method	XML Document to Integration Object Hierarchy

- b Click the Input Arguments form and create a new argument called MessageText, using the information from the following table:

Input Argument	Type	Value	Property
MessageText	Process Property		InputBizdoc

- c Click the Output Arguments form and set the new argument, according to the information in the following table:

Property Name	Type	Output Argument
SiebelMessage	Output Argument	Siebel Message

- 5 For the “Convert to XML and Write to Database” business service (Step 3 in the workflow), double-click on the object icon to open the Business Service form.

- a** In the Business Service form, enter the information from the following table:

Field Name	Value
Business Service	Siebel Adapter
Method	Upsert

- b** Click the Input Arguments form and set a new argument according to the information in the following table:

Name	Type	Property
Siebel Message	Process Property	Siebel Message

NOTE: Now that you have created the workflow, the workflow information must be supplied to the Siebel Application Integration Component. The Siebel AIC, which is a *pipeline* component, is implemented as a special kind of COM object that the BizTalk Server state engine calls to deliver data to an application. Therefore, you need to configure a BizTalk Server port and channel in BizTalk to include use of this Siebel AIC pipeline component for application integration.

To supply the workflow information for AIC Inbound

- 1** Run the BizTalk Server Messaging Manager on the BizTalk Server machine.
 - a** Set up an Organization for Siebel applications (such as “Siebel Sales”). Set it to be the Home Organization.
 - b** Set up a Siebel Trading Partner, for example, Microsoft, to be the Destination Organization.
 - c** Create new document specification for the XDR document definition stored in WebDav.
- 2** Set up a new BizTalk Server port to send the document to the Siebel application (for example, Siebel Sales) with the Primary Transport of Application Integration Component, BizTalk SiebelBizTalkAIC.

- a** Using BizTalk Messaging Manager, create a new port named Siebel AIC BTS.
- b** Specify the primary transport type to be Application Integration Component.
- c** Specify the Component Name to be the AIC component name, BizTalk.AppIntegrationSiebel.
- d** Save your work.

When you have finished setting up this port, BizTalk is configured to deliver messages to Siebel BizTalk AIC.

3 Set up a new channel for the port created in [Step 2](#).

After you have specified AIC as the primary transport for document interchange with Siebel information, you need to supply BizTalk with configuration data specific to Siebel AIC. You do so by creating a new channel. For information on setting up ports and channels, see Microsoft BizTalk Server documentation.

This data is passed to the Siebel AIC component through the BizTalk Management Desk. One of the configuration items that is passed to the Siebel AIC component is the name of the workflow you defined within Siebel application. AIC dynamically invokes the Siebel Workflow engine and passes the XML document to the workflow specified in the channel AIC properties.

After Siebel AIC is implemented, the property sheet contains the properties specific to the Siebel AIC implementation, one of which is "Siebel Workflow Name."

You can optionally create maps using BizTalk Mapper if data transformation is required for the documents received by Siebel application from the external application. For information on the mapping capabilities of BizTalk, see Microsoft BizTalk Server online documentation.

- a** Configure the Advanced properties in the final Channel Configuration page.
- b** Provide the Siebel Workflow name to be invoked as previously defined in [Step 2](#) of ["To create a workflow to receive inbound XML documents over AIC from BizTalk"](#) on page 166.

- c** Enter the user name, password, and connect string for the Active X Data Control to connect to Siebel Server. For example (notice the quotes):

```
host="siebel.tcpip.none.none://HOST/EnterpriseServer/  
SCCObjMgr_enu/SiebelServer" lang="ENU"
```

Where “HOST” is the machine name where the gateway server is installed and lang is the language of the Siebel Server. The default lang is ENU but for other languages you have to provide this value.

CAUTION: Make sure the connect string you type is exactly correct, including the use of quotes. The correct connect string format is very important in order for ADC to correctly connect to Siebel Server.

- 4** Configure a File Receive Function from BizTalk Server Administration to poll a file location and deliver to the channel configured in [Step 3](#).

NOTE: This example uses File transport to receive documents from trading partners for the sake of illustration. In your actual business situations, a trading partner can deliver messages to a Siebel application over *any* supported transport.

- a** Open BizTalk Server Administration, expand Microsoft BizTalk Server, and expand the server group to which you want to add the File receive function.
- b** Select Receive Functions.
- c** Choose Action > New > File Receive Function.
- d** The Add a File Receive Function dialog box appears.
- e** In the Name box, type the name of the File receive function, for example, “Microsoft to Siebel AIC.”
- f** In the Comment box, add a brief description (optional).
- g** In the Server on which the receive function will run list, click the name of a server in the group.

- h** For the Polling Location, enter `c:\AIC`.

NOTE: This is the location where you will copy the XML Instance you created earlier so the channel can pick it up from there and send it to Siebel application.

- i** In the File types to poll for box, type the extension of the files that BizTalk Server receives. In this case, type `*.xml`.
- j** Click the Advanced tab and enter as the Channel Name as “AIC Microsoft To Siebel Channel,” for example.
- k** Click OK to finish.

Connecting to BizTalk Using HTTP

This section describes how to set up and use the HTTP Outbound Transport to send messages to BizTalk Server and how to use HTTP to receive messages from BizTalk. Topics covered include:

- [Siebel HTTP Outbound Transport](#)
- [Siebel HTTP Inbound Transport on page 180](#)

Siebel HTTP Outbound Transport

BizTalk Server uses the Microsoft Internet Information Server (IIS) and Active Server Pages (ASP) to receive documents over HTTP. The ASP file uses BizTalk `IIInterchangeSubmit()` and `SubmitSync()` methods to send documents to BizTalk Server.

Checklist

- ❑ HTTP Outbound Transport requires a virtual directory. This directory needs to reside on the IIS to which Siebel outbound XML documents will be sent. Typically, this is on the same machine as BizTalk Server.

For details, see [“To create a virtual directory for HTTP Outbound” on page 175](#).

- ❑ Copy the sample ASP file to the virtual directory. You can modify the Active Server Page to meet your business requirements. This section provides you with a sample ASP file to serve as a general guideline.

For details, see [“To copy the ASP files to the new HTTP Outbound directory” on page 175](#).

To create a virtual directory for HTTP Outbound

- 1 Create a new directory for the ASP file.

The directory should be on the Web Server where you would be sending your outbound XML documents from Siebel application. This is usually the BizTalk machine.

- 2 Create a new virtual directory on IIS using the new directory you created. Name the directory, for example, BizTalkReceive.

NOTE: The virtual directory can be on either the Siebel Server or the BizTalk Server. If the Internet Information Server is on the Siebel Server machine, you need to register the BizTalk Server Interchange application to the Siebel Server. If IIS is on the BizTalk Server, you *do not* need to register Interchange application, because it is already a part of the BizTalk Server installation. See your Microsoft Internet Information Server (IIS) documentation for details on setting up virtual directories.

To copy the ASP files to the new HTTP Outbound directory

- Copy the ASP file SiebelBizTalkOutHTTP.asp from:

`<Siebel Server installation directory>\eaiconn\BTS\SCRIPTS`

to the new directory you just created, for example:

`c:\BizTalkReceive\`

Sending Messages to BizTalk Using the Siebel HTTP Outbound Transport

This section explains how to use the Siebel BizTalk interface to send an XML document to BizTalk Server over HTTP Outbound transport.

Checklist

- ☐ Set up the Siebel HTTP Outbound Transport, if you have not already done so.
For details, see [“Siebel HTTP Outbound Transport” on page 174.](#)
- ☐ Set up the organization, port, and channel (collectively known as BizTalk configuration objects) for use with the HTTP Outbound Transport.
For details, see [“To set up BizTalk configuration objects for HTTP Outbound,”](#) below.
- ☐ Create the workflow to process outbound documents from Siebel using HTTP transport.
For details, see [“To create the Siebel workflow for HTTP Outbound” on page 178.](#)

To set up BizTalk configuration objects for HTTP Outbound

- 1** On the BizTalk Server machine, access the BizTalk Messaging Manager.
- 2** Set up your home organization as Siebel and set up applications for the home organization.

NOTE: For each BizTalk-specific step, see Microsoft BizTalk Server documentation for the details.

- 3** Create a new organization and name it as is appropriate. For example, if your trading partner is Microsoft, enter “Microsoft.”
- 4** Create a document definition for previously-created document specification from WebDAV and name it, for example, Siebel Order. See [To import schema into BizTalk Server to save it as a BizTalk document specification on page 136.](#)

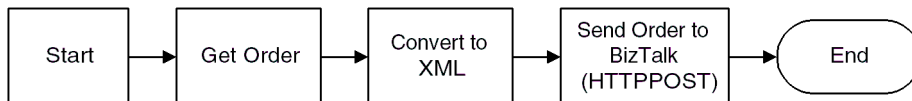
- 5** Create a messaging port to your trading partner.
 - a** In the Destination Organization window, select “Microsoft,” for example, as the Organization.
 - b** Click Browse under Primary Transport.
 - c** Select the appropriate transport for your trading partner.
 - d** Complete the remaining pages as needed and click Finish to complete.
- 6** Create a new channel for the port created in [Step 5](#).
 - a** Create a new channel “From an Application,” named, for example, “Siebel To Microsoft Channel.”
 - b** Click Next.
 - c** Select the appropriate inbound document definition, for example, “Siebel Order.”
 - d** Select appropriate outbound document definition for your trading partner, for example, “Microsoft Order.”
 - e** Select an appropriate map, if necessary.
 - f** Complete other pages as required and select Finish to complete.
- 7** Modify `SiebelBizTalkOutHTTP.asp` to specify the Channel name you configured in [Step 6](#).
 - a** Search for “sChannel” in the file.
 - b** Remove the comment designator from this statement:

`sChannel = <Channel Name you just created>`

To create the Siebel workflow for HTTP Outbound

- 1 On the Siebel machine, access the Workflow Process Designer.

Set up a workflow for sending a message to BizTalk so that it follows the flow shown in the following illustration (use Start, three business objects, and End):



NOTE: For information on how to use the Workflow Designer, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

- 2 Define the XML conversion. For an example, refer to the following table.

Business Service	Value
Business Service	XML Converter
Method	Integration Object Hierarchy to XML Document

Input Argument	Value
Input Argument	XML Character Encoding
Value	UTF-16

- 3 For the Send Order to BizTalk (HTTP Post) business service (Step 4 in the workflow), double-click on the object icon to open the Business Service form.

- In the Business Service form, enter the information from the following table:

Field Name	Value
Business Service	EAI BTS COM Transport
Method	<ul style="list-style-type: none"> ■ Send ■ Send and Receive Response (These require a value for Output, as well)

- Click the Input Arguments form and create two (2) new arguments according to the information in the following table:

Input Argument	Value
MessageText	The XML data string that you want to send to BizTalk. Typically, this corresponds to the name of the workflow process property that stores the actual data, for example, "Order XML."
Request Method	POST.
Request URL Template	The URL to which the document will be posted.

- Enter the Request URL Template literal:

```
http://<WebServer>/<virtual directory name>/
SiebelBizTalkOutHTTP.asp
```

Where <virtual directory name> is the virtual directory you set up in [“To create a virtual directory for HTTP Outbound” on page 175](#), for example, BizTalkReceive.

- 4 Save your workflow.

NOTE: Refer to [Chapter 4, “EAI HTTP Transport”](#) for more information on setting up and using the Siebel EAI HTTP Transport.

Siebel HTTP Inbound Transport

This section describes how to set up and use HTTP Transport for receiving documents from BizTalk over HTTP.

NOTE: BizTalk Siebel HTTP Inbound Transport has been redesigned and implemented as an HTTP-based Application Integration Component (AIC) for the Siebel 7.5 release. This new HTTP-based AIC combines the functionality of the 7.0.x ASP application and the HTTP COM utility.

There are two high-level steps to this procedure:

- [Configuring Siebel Server](#)
- [Setting Up HTTP-Based AIC to Run on BizTalk Server](#)

Configuring Siebel Server

You need to set up EAI Inbound HTTP Transport in order to receive documents from BizTalk over HTTP. There are three tasks involved with configuring the Siebel Server for Inbound HTTP communication with BizTalk. These three steps include:

- Enabling the EAI Component group on the Siebel Server
- Setting up Siebel eAI
- Setting up the Siebel Web Engine

For details on each of these steps, see [Chapter 4, “EAI HTTP Transport.”](#)

Setting Up HTTP-Based AIC to Run on BizTalk Server

This section explains how to set up the required components for running the HTTP-based AIC Inbound Transport on the BizTalk Server machine. Although the AIC can run either on the Siebel Server machine or on the BizTalk Server machine, it is recommended that you run the AIC on the BizTalk Server machine locally for ease of configuration. If you would like to run the AIC on the Siebel Server, see [“Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server”](#) on page 161.

NOTE: The HTTP-based AIC does *not* require ActiveX Data Controls.

Checklist

- ☐ Copy the HTTP-based AIC Dynamic Link Library (DLL) from the Siebel Server machine to the BizTalk Server machine.

For details, see [“To copy HTTP-based AIC DLL from Siebel Server to BizTalk Server Machine.”](#)
- ☐ Register the Siebel AIC on the BizTalk Server machine (one-time operation).

For details, see [“To register AIC as a COM + Server Application on the BizTalk Server machine.”](#)
- ☐ Copy the required Siebel HTTP AIC Active Server Pages (ASP) to the BizTalk Server machine.

For details, see [“To copy the required Siebel HTTP AIC Active Server Pages \(ASP\) to the BizTalk Server machine”](#) on page 182.

To copy HTTP-based AIC DLL from Siebel Server to BizTalk Server Machine

- Copy the following file from Siebel Server to BizTalk Machine:

```
<Siebel Server installation  
directory>\eaiconn\bts\http\sscaeibh.dll
```

to a newly created directory, for example:

```
c:\AIC
```

To copy the required Siebel HTTP AIC Active Server Pages (ASP) to the BizTalk Server machine

- Copy the Siebel_BizTalkHTTPAIC_1_post.asp page and the Siebel_BizTalkHTTPAIC_1.asp page from Siebel Machine:

<Siebel installation dir>\eaiconn\bts\scripts

to BizTalk Machine:

<BizTalk installation directory>\MessagingManager\pipeline

Receiving Messages from BizTalk Using the HTTP Inbound Transport

This section explains how you use the BizTalk interface to receive XML documents using the Siebel HTTP Inbound transport. The activities associated with the inbound communication from BizTalk occur in the BizTalk Messaging Manager. For information on using this BizTalk application, consult the Microsoft BizTalk Server documentation.

Checklist

- ☐ Set up the Siebel HTTP Inbound Transport.
For details, see [“Siebel HTTP Inbound Transport” on page 180](#).
- ☐ Set up the HTTP Transport Named Subsystem and the HTTP Receive function to receive and process incoming XML documents.
For details, see [“HTTP Transport Named Subsystems” on page 60](#).
- ☐ Set up the organization, port, and channel (collectively known as BizTalk configuration objects) for use with the HTTP Inbound Transport.
For details, see [“To set up BizTalk configuration objects for HTTP inbound” below](#).

To set up BizTalk configuration objects for HTTP inbound

- 1** Run the BizTalk Server Messaging Manager on the BizTalk Server machine.
 - a** Set up an Organization for Siebel applications (such as “Siebel Sales”). Set it to be the Home Organization.
 - b** Set up a Siebel Trading Partner, for example Microsoft, to be the Destination Organization.
 - c** Create new document specification for the XDR document definition stored in WebDav.
- 2** Set up a new BizTalk Server port to send the document to the Siebel application (for example, Siebel Sales) with the Primary Transport of Application Integration Component, Siebel.BizTalkHTTPAIC.
 - a** Using BizTalk Messaging Manager, create a new port named “Siebel HTTP AIC Port.”
 - b** Specify the primary transport type to be Application Integration Component.
 - c** Specify the Component Name to be the AIC component name, Siebel.BizTalkHTTPAIC.
 - d** Save your work.

When you have finished setting up this port, BizTalk is configured to deliver messages to Siebel BizTalk AIC.

- 3** Set up a new channel, for example “Siebel HTTP AIC Channel,” for the port created in [Step 2](#).

After you have specified HTTP AIC as the primary transport for document interchange with Siebel information, you need to supply BizTalk with configuration data specific to Siebel HTTP AIC. You do so by creating a new channel. For more information on setting up ports and channels, see Microsoft BizTalk Server documentation.

This configuration data is passed to the Siebel HTTP AIC component through the BizTalk Server. One of the configuration items that is passed to the Siebel HTTP AIC component is the name of the HTTP Service you defined within Siebel application. HTTP AIC dynamically invokes the HTTP Service within the Siebel application and passes the XML document to the HTTP Service through the Siebel Web Engine.

You can optionally create maps using BizTalk Mapper if data transformation is required for the documents received by Siebel application from the external application. For information on the mapping capabilities of BizTalk, see Microsoft BizTalk Server online documentation.

- a** Configure the Advanced properties in the final Channel Configuration page.
- b** Provide the input parameters as required by the EAI HTTP Transport:

Web Server is the server where SWE is running (format is defined as `webserver:Port`), Source is the new HTTP Service you previously created, and the user name and password to connect to Siebel Server.

- 4** Configure a File Receive Function from BizTalk Server Administration to poll a file location and deliver to the channel configured in [Step 3](#).

NOTE: This example uses File transport to receive documents from trading partners for the sake of illustration. In your actual business situations, a trading partner can deliver messages to a Siebel application over *any* supported transport.

- a** Open BizTalk Server Administration, expand Microsoft BizTalk Server, and expand the server group to which you want to add the File receive function.
- b** Select Receive Functions.

- c** Choose Action > New > File Receive Function.
- d** The Add a File Receive Function dialog box appears.
- e** In the Name box, type the name of the File receive function, for example, “Siebel HTTP AIC” (no quotes).
- f** In the Comment box, add a brief description (optional).
- g** In the Server on which the receive function will run list, click the name of a server in the group.
- h** For the Polling Location, enter `c:\http`.

NOTE: This is the location where your trading partner delivers the XML documents for Siebel so the channel can pick it up from there and send it to the Siebel application for processing.

- i** In the File types to poll for box, type the extension of the files that BizTalk Server receives. In this case, type `*.xml`.
- j** Click the Advanced tab and enter as the Channel Name as “Siebel HTTP AIC Channel” (no quotes), for example.
- k** Click OK to finish.

If there are any errors while sending the document to Siebel Server, they are written to the Application Event log with details.

Many enterprises, especially those involved in eBusiness, develop and implement Java applications to meet a variety of business requirements. Typically, these applications combine existing enterprise information systems with new business functions to deliver services to a broad range of users.

Siebel applications generate JavaBeans for existing Integration Objects and Business Services by means of the Siebel JavaBean Wizard. By generating JavaBeans that represent Siebel Integration Objects or Business Services, Siebel applications help facilitate the creation of J2EE applications that need to interface with the Siebel eBusiness Applications. The Siebel Code Generator produces JavaBeans for Integration Objects, Business Services, and their related components, allowing you to identify and use the Java code you need for your application.

In addition to the generation of Java Beans, Siebel applications support the J2EE Connector Architecture with the Siebel Resource Adapter and also provide a range of tools for integrating with J2EE environments, supporting both loosely-coupled and tightly-coupled integration architectures using HTTP and IBM MQSeries, as illustrated in [Figure 7](#).

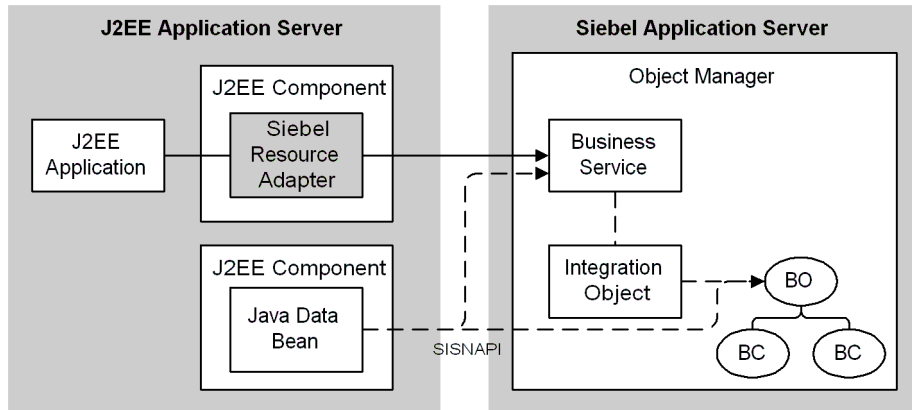


Figure 7. J2EE Application Server Integration Architecture

The Siebel JavaBean Wizard

To ease integration between Siebel applications and J2EE Application Servers, Siebel applications provides a wizard that generates JavaBeans for Business Services and Integration Objects. In Siebel Tools, you generate JavaBeans for the Siebel objects you want to interface with and then include the generated JavaBeans in the J2EE application or component you are constructing. The generated JavaBeans act as client-side proxy stubs to the corresponding objects on the Siebel Server and are not the actual Business Service logic written in Java.

Siebel Tools ships with a wizard for creating JavaBeans based on Siebel Business Services or Siebel Integration Objects. Creating JavaBeans based on Business Services or Integration Objects provides a uniform mechanism to interact with these objects from within a J2EE component or application.

Using these beans, you can develop J2EE components that retrieve data and content from within Siebel applications and trigger events within Siebel applications depending upon the nature of the interaction. The generation of these beans reduces the complexity of the integration, while providing a standard, component-based interface to Siebel applications. JavaBeans generated by the Wizard may be used in conjunction with either the Siebel Java Data Bean, or the Siebel Resource Adapter.

For more information about the Siebel Java Data Bean, see *Siebel Tools Online Help, MidMarket Edition*. For more information about the Siebel Resource Adapter, see [“The Siebel Resource Adapter” on page 211](#).

Generating JavaBeans for Integration Objects

This section describes how to generate JavaBeans for Siebel Integration Objects, as well as the methods that are generated for each JavaBean. For information on how to generate JavaBeans for Siebel Business Services, see [“Generating JavaBeans for Business Services” on page 197](#).

You use the JavaBean Wizard to produce the JavaBean based upon Siebel integration objects. JavaBeans based upon Integration Objects are designed to be used with the EAI Siebel Adapter and may be used to query, delete, upsert, and synchronize information maintained within Siebel’s database. The structure of the JavaBeans generated for an integration object and its components is shown in [Figure 8 on page 190](#). The integration object and its components will each have their own Java class, stored in the `com.siebel.local. <Integration Object Name>` package.

The Siebel Java Data Bean provides a Java-based programming interface to the Siebel eBusiness Applications. Using the Siebel Java Data Bean, you can build Java or J2EE components that interact with the Siebel Application Object, Business Objects, Business Components, Business Services and Property Sets.

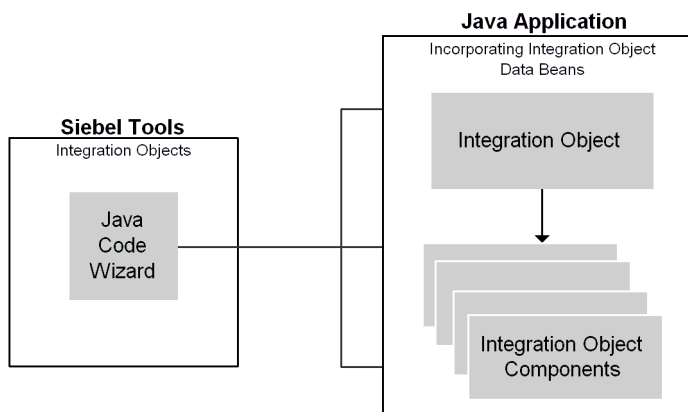


Figure 8. Generate Code Wizard Creates Java for Siebel Integration Objects

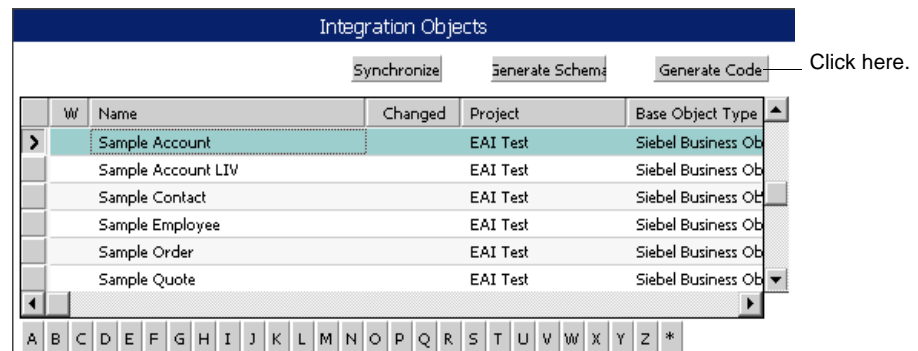
To generate Java code for a Siebel Integration Object

- 1 Login to Siebel Tools and lock the project you are going to work on.

NOTE: For information on how to use Siebel Tools and how to lock projects and so on, consult *Siebel Tools Online Help, MidMarket Edition*.

- 2 Select Integration Objects from the Types tab on the Object Explorer.
- 3 Select the Integration Object for which you want to create Java code from the Integration Objects list (as shown in the following illustration).

On the right top corner of the Integration Object list there is a set of three buttons. The following illustration shows the Sample Account Integration Object highlighted.



- 4 Click the Generate Code button.
- 5 Complete the wizard:
 - a Leave the Business Service as is (there is only one applicable: the Siebel Code Generator).
 - b Select either JDB (Java Data Bean) or JCA (J2EE Connector Architecture/ Siebel Resource Adapter) for the Supported Language.

- c** Browse to select an existing folder as the output folder. Your Java code for the selected integration objects will be stored in subdirectories there, as explained next.
- d** Click Finish.

The code is generated and the wizard closes, returning you to the Integration Objects form.

About the Folders and Files Created for Integration Object Java

A new folder named “com” is created as the first subfolder under the folder you specified in the wizard as your root folder. Beneath it is a “siebel” directory, and below that a “local” directory. Here is where you will find separate folders for the Java code generated for any Integration Object for which you have generated Java code, as shown in [Figure 9](#).

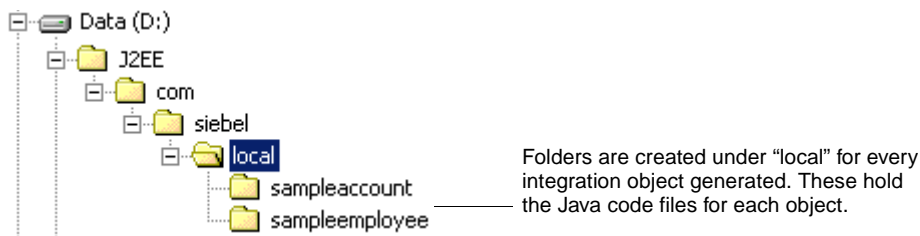


Figure 9. Directory Structure Created to Contain Java Code for Integration Objects

The files will be created under the last leaf of the directory as shown in [Figure 10](#).

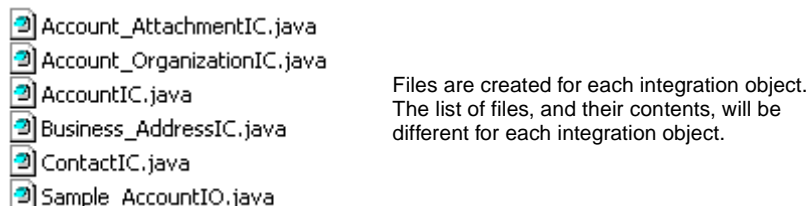


Figure 10. Java Bean Files for Sample Account Integration Object

Generated JavaBean for the Sample Account Integration Object

Table 18 provides a description of the Java classes generated for the Sample Account Integration Object. The files the JavaBean Wizard generates will vary for each Integration Object definition. The table presented here is designed to provide you with an idea of the structure of the Java classes that are generated for a typical Integration Object, as well an explanation of what each file represents.

Table 18. Java File List for Sample Account Integration Object

Java File	Description
Account_AttachmentIC.java	Java class that corresponds to the Account Attachment Integration Object Component. Provides a set of methods that allow Java developers to construct the Account Attachment Integration Object Component, convert to and from a Property Set, and to define relationships between Integration Object Components.
Account_OrganizationIC.java	Java class that corresponds to the Account Organization Integration Object Component. Provides a set of methods that allow Java developers to construct the Account Organization Integration Object Component, convert to and from a Property Set, and to define relationships between Integration Object Components.
AccountIC.java	Java class that corresponds to the Account Integration Object Component. Provides a set of methods that allow Java developers to construct the Account Integration Object Component, convert to/from a Property Set, and to define relationships between Integration Object Components.
Business_AddressIC.java	Java class that corresponds to the Business Address Integration Object Component. Provides a set of methods that allow Java developers to construct the Business Address Integration Object Component, convert to/from a Property Set, and to define relationships between Integration Object Components.

Table 18. Java File List for Sample Account Integration Object

Java File	Description
ContactIC.java	Java class that corresponds to the Contact Integration Object Component. Provides a set of methods that allow Java developers to construct the Contact Integration Object Component, convert to/from a Property Set, and to define relationships between Integration Object Components.
Sample_AccountIO.java	Java class that corresponds to the Sample Account Integration Object. Provides a set of methods that allow Java developers to construct an Integration Object, add Integration Object Components to the Integration Object and convert to/from a Property Set.

Default Java Methods for Integration Objects

[Table 19](#) lists and describes are default methods generated by Siebel Tools for Integration Objects for Java code.

Table 19. Default Java Methods for Integration Objects

Object	Description
addfIntObject(IntObjComp ioc)	Adds an integration object component object to the integration object.
clone	Makes a copy of the Integration Object.
equals(Object)	Helper method to determine if two Integration Object objects are equal.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an Integration Object from a Property Set.
getfIntObjectFormat	Access method that returns a String containing the format of the Integration Object.
getfIntObjectName	Access method that returns the integration object name property.
getfIntObjInst	Access method that returns a Vector representation of the Integration Object.

Table 19. Default Java Methods for Integration Objects

Object	Description
getfMessageId	Access method that returns the MessageId property of the integration object.
getfMessageType	Access method that returns the MessageType property of the integration object.
getfOutputIntObjectName	Access method that returns the OutputIntObjectName property of the integration object.
Integration_ObjectIO()	Default constructor.
Integration_ObjectIO(SiebelPropertySet ps)	This method will create an integration object (and its hierarchy) based from a property set.
setfIntObjectFormat	Access method that sets the IntObjectFormat property of the integration object.
setfIntObjectName	Access method that sets the IntObjectName property of the integration object.
setfMessageId	Access method that sets the MessageId property of the integration object.
setfMessageType	Access method that sets the MessageType property of the integration object.
setfOutputIntObjectName	Access method that sets the OutputIntObjectName property of the integration object.
toPropertySet	Method that converts the integration object to a property set.

Default Java Methods for Integration Object Components

[Table 20](#) lists and describes are default methods generated by Siebel Tools for Integration Object Components for Java code.

Table 20. Default Java Methods for Integration Object Components

Object Component	Description
addfChildIntObjComp(ChildIntObjComp)	This method will create a child integration object component of type ChildIntObjComp and associate with the parent integration object component.
clone	Makes a copy of the Integration Object.
equals(Object)	Helper method to determine if two Integration Object objects are equal.
fromPropertySet(SiebelPropertySet)	This method will populate the Integration Object Component based upon the contents of a property set.
getfChildIntObjComp	Returns a Vector containing all child integration object components of type ChildIntObjComp associated with the parent integration object component.
getFieldName()	Access method that returns the value of FieldName.
IntObjCompIC()	Default constructor.
IntObjCompIC(SiebelPropertySet)	This method will create an integration object component from a property set.
setfFieldName(val)	Access method that sets class property FieldName to “val.”
toPropertySet	Method that converts an Integration Object Component to a Property Set.

Generating JavaBeans for Business Services

This section describes how to generate JavaBeans for Siebel Business Services. For information on how to generate JavaBeans for Siebel Integration Objects, see [“Generating JavaBeans for Integration Objects” on page 190](#).

You use the JavaBean Wizard to generate JavaBeans based on Siebel Business Services. The wizard produces JavaBeans for the Business Service and the Input and Output parameters for each Business Service Method defined in Siebel Tools, as shown in [Figure 11](#). If methods of the Business Service take an Integration Object as a parameter, the JavaBeans for the Integration Objects is not generated. In order to generate JavaBeans for Integration Objects, see [“Generating JavaBeans for Integration Objects” on page 190](#). The Business Service, Input and Output parameters for each method will each have their own Java class, stored in the `com.siebel.jdb.service.< Business Service Name >` or `com.siebel.jca.service.< Business Service Name >` package.

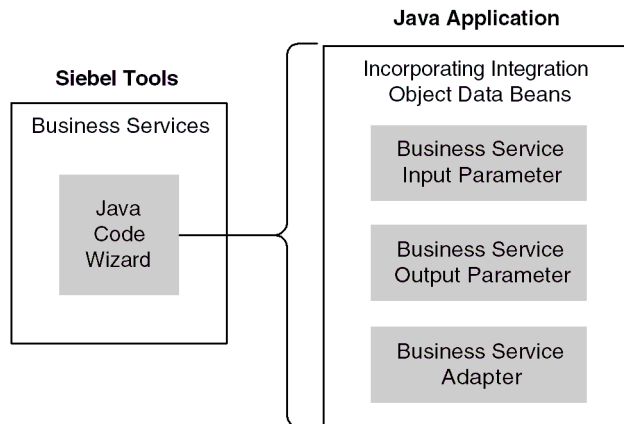


Figure 11. Generate Code Wizard Creates Java for Siebel Business Services

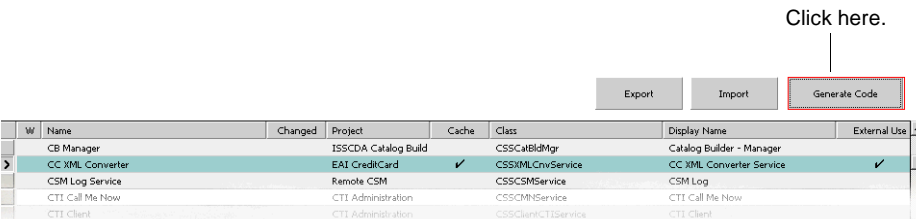
To generate Java code for a Siebel Business Service

- 1 Login to Siebel Tools and lock the project you are going to work on.

NOTE: For information on how to use Siebel Tools and how to lock projects and so on, consult *Siebel Tools Online Help, MidMarket Edition*.

- 2 Select Business Service from the Types tab on the Object Explorer.
- 3 Select the Business Service for which you want to create Java code from the Business Services list (as shown in the following illustration).

On the right top corner of the Business Service list there is a set of three buttons. The following illustration shows the CC XML Converter Business Service highlighted.



- 4 Click the Generate Code button.

NOTE: If the Business Service you selected is not available to be exported, you will see an error message to that effect, rather than the wizard.

- 5 Complete the wizard:
 - a Leave the Business Service as is (there is only one applicable: the Siebel Code Generator).
 - b Select either JDB (Java Data Bean) or JCA (J2EE Connector Architecture/ Siebel Resource Adapter) for the Supported Language.
 - c Browse to select an existing folder as the output folder. Your Java code for the selected Business Service will be stored in subdirectories there, as explained next section.
 - d Click Finish.

About the Folders and Files Created for Business Service Java

A new folder named “com” is created as the first subfolder under the folder you specified in the wizard as your root folder. Beneath it are the “siebel\ei” directories. Finally, under the “service” directory you will find separate folders for the Java code generated for any Business Service for which you have generated Java code, as shown in [Figure 12](#).

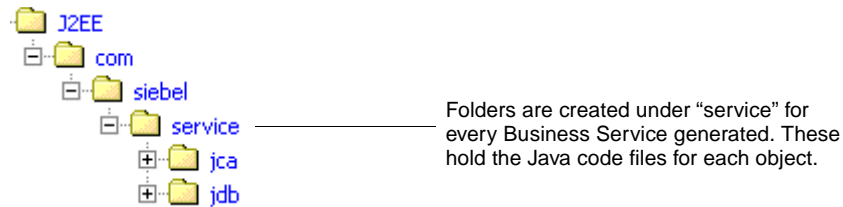


Figure 12. Directory Structure Created to Contain Java Code for Integration Objects

The files will get created under the last leaf of the directory as shown in [Figure 13](#).

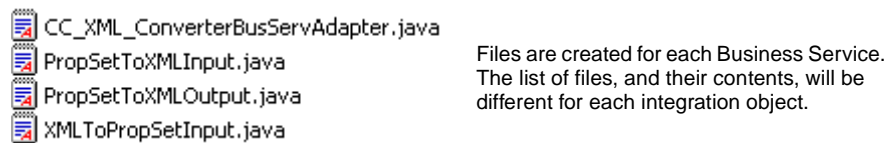


Figure 13. JavaBean Files for Sample Account Integration Object

Generated Java Files for a Typical Business Service

Table 21 provides an explanation for the Java classes generated for the CC XML Converter business service. JavaBeans generated for each business service will vary based upon the method defined and the Input/Output arguments. The table is presented to give you an idea of the Java classes generated for a typical business service and an explanation for what each class represents. Please note that for the CC XML Converter business service, the XMLToPropSet method does not have any Output parameters, which is why the output class for this method was not generated. Table 22, Table 23, and Table 24 show more specific information about the default methods.

Table 21. Java File List for CC XML Converter Business Service

Java File	Description
CC_XML_Converter BusServiceAdapter.java	Java class that represents to the CC XML Converter Business Services. Provides a set of methods to construct the Business Service Adapter Class as well as invoke the Business Service methods.
PropSetToXMLInput.java	Java class that represents the Input parameters of the PropSetToXML method of the CC XML Converter Business Service. Provides a set of access methods to set and get the method input parameters as well as convert to and from a Property Set.
PropSetToXMLOutput.java	Java class the represents the Output parameters of the PropSetToXML method of the CC XML Converter Business Service. Provides a set of access methods to set and get the method output parameters as well as convert to and from a Property Set.
XMLToPropSetInput.java	Java class that represents the Input parameters of the XMLToPropSet method of the CC XML Converter Business Service. Provides a set of access methods to set and get the method input parameters as well as convert to and from a Property Set.

Table 22. Default Java Methods for the Business Service Adapter Class

Method	Description
BusServAdapter()	Constructor.
BusServAdapter(SiebelDataBean)	Constructor.
BusServAdapter(String, String, String)	Constructor.
mBusSvcMethodName(methodInput)	Method that invokes the specified Business Service Method.

Table 23. Default Java Methods for Business Service Method Input Parameters

Method	Description
methodInput()	Constructor.
methodInput(SiebelPropertySet)	Constructor.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an Integration Object from a Property Set.
toPropertySet()	Method that converts the methodInput class to a Property Set.
getMethodArgName()	Access method to get the value of Business Service Method argument.
setfMethodArgName()	Access method to set the value of a Business Service Method argument.

Table 24. Default Java Methods for Business Service Method Output Parameters

Method	Description
MethodOutput()	Constructor.
methodOutput(SiebelPropertySet)	Constructor.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an Integration Object from a Property Set.
toPropertySet()	Method that converts the methodInput class to a Property Set.

Table 24. Default Java Methods for Business Service Method Output Parameters

Method	Description
getMethodArgName()	Access method to get the value of Business Service Method argument.
setfMethodArgName()	Access method to set the value of a Business Service Method argument.

Using the Java Data Bean

The Siebel Java Data Bean provides a Java-based programming interface to the Siebel eBusiness Applications. Using the Siebel Java Data Bean, you can build Java or J2EE components that interact with the Siebel Application Object, Business Objects, Business Components, Business Services and Property Sets.

For additional information on the interfaces exposed by the Siebel Java Data Bean, see *Siebel Tools Online Help, MidMarket Edition*.

The Development Environment

Three Siebel .jar files are needed when you compile your Java application, and run time. The files are:

- SiebelJI.jar
- SiebelJI_common.jar
- SiebelJI_%lang%.jar.

Where %lang% is the installed language pack; for example, SiebelJI_enu.jar for English.

The .jar files may be installed through the Siebel Enterprise Server install, or by installing Siebel Tools. For more information please refer to the *Siebel Server Installation Guide for Microsoft Windows, MidMarket Edition*. To deploy the J2EE applications using the Siebel Java Data Bean or JavaBeans generated by the Java Wizard, include these .jar files, so that the J2EE application can access the Java Data Bean class files.

To deploy in WebLogic the sample Servlet/JSP application, for example, place the .jar files in the lib subdirectory of WEB-INF corresponding to the application, as shown in Figure 14.

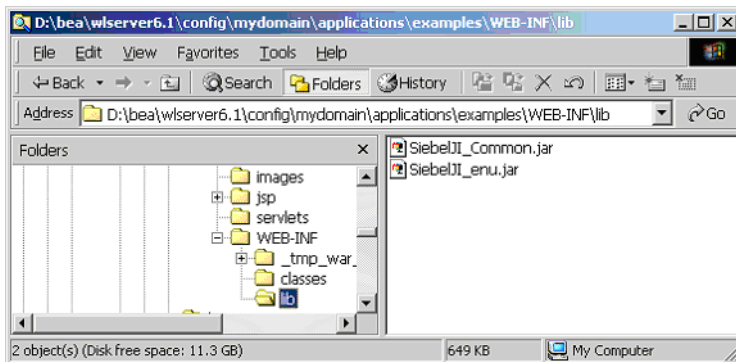


Figure 14. Example of .jar Files Location

NOTE: For other application servers, refer to the application server’s documentation for information on where to place the .jar files.

To install the Siebel .jar files

- To install, simply copy the .jar files to your working directories and include them in the classpath when compiling your source files.

To compile source files using Siebel Data Bean

- You combine source files using the command line executable, “javac.exe.” The Siebel files can also be used directly in your development environment. To compile using command line, enter one of the following strings, depending on which flavor of Java you are using. The command string is entered on a single line.

For Sun’s JVM:

```
javac -classpath
.;SiebelJI_Common.jar;SiebelJI_%lang%.jar;SiebelJI.jar
<source files>
```

To run applications containing Siebel Java Data Beans

- To run class files that include Siebel Data Beans or JavaBeans generated by the Java Wizard, enter the following at the command line:

```
jre -cp .;SiebelJI.jar;SiebelJI_Common.jar;SiebelJI_%lang%.jar
<class files>
```

Using Java Code Generated by Siebel eBusiness Applications

Set up the development environment as described in [“Using the Java Data Bean” on page 203](#). If you are using an Integrated Development Environment (IDE), be sure to include the .jar files in the classpath or input classes for your project. Refer to your IDE documentation on how to include external class/jar files in to your project.

NOTE: A javadoc jar file (Siebel_JavaDoc.zip) is installed when you choose to install the Siebel Java Integrator option (a component of the Tools or Siebel Server installer). This jar file is in the same location as SiebelJI* jar files under the %INSTALLED_DIR%\classes directory. This jar file contains the up-to-date java doc for the Siebel Java Data Bean, Siebel Resource Adapter, and dependent classes.

Sample Java Code 1

The following is a code sample that describes the use of the Siebel Java Data Bean in a Servlet. The Java Data Bean can be used in a similar way in Java Server Pages (JSPs) or in Enterprise Java Beans (EJBs).

```
import com.siebel.data.*;
...
SiebelDataBean m_dataBean = null;
SiebelBusObject sbo_busObject = null;
SiebelBusComp sbc_busComp = null;

try {
    boolean result;
    ...
    String userLoginName = ...
    String strJdbLoginUser = ...
    String strJdbLoginPwd = ...
    ...
    m_dataBean = new SiebelDataBean();
    result = m_dataBean.login("siebel://trothwein2/703/SCCObjMgr/trothwein2",
strJdbLoginUser, strJdbLoginPwd, "enu");
    SiebelBusObject sbo_busObject = m_dataBean.getBusObject("Contact");
    SiebelBusComp sbc_busComp = sbo_busObject.getBusComp("Contact");
```

```
sbc_busComp.setViewMode(3);
sbc_busComp.clearToQuery();
sbc_busComp.activateField("First Name");
sbc_busComp.activateField("Last Name");
sbc_busComp.activateField("Id");
sbc_busComp.setSearchSpec("Login Name",userLoginName);
sbc_busComp.executeQuery2(true,true);

if (sbc_busComp.firstRecord()) {
System.out.println("Contact ID: " + sbc_busComp.getFieldValue("Id")+
    " ; First Name:"+sbc_busComp.getFieldValue("First Name")+
    " ;Last Name:"+sbc_busComp.getFieldValue("Last Name"));
}
sbc_busComp.release();
sbo_busObject.release();
m_dataBean.logoff();
} catch (SiebelException se) {
    System.err.println("Error:"+se.getErrorMessage());
    se.printStackTrace();
}
```

About the Code Sample

The code sample above shows a simple use of the Java Data Bean from a Java Servlet to search for a particular login name in the “Contact” business component. The first step in using the Java data bean is to log in to the Object Manager of the Siebel system. The first parameter, the connection string, specifies the protocol, machine name, enterprise name, object manager and the server name. After you are logged in to the object manager you can use the `getBusObject` and `getBusComp` methods to obtain the business object and business component objects.

NOTE: You may have to change this based on your login permissions.

The code sample activates fields to allow the query to retrieve data for the specified fields, specifies the search criteria, and executes the query. For additional details on these methods see the *Object Interfaces Guide: Java Data Bean Quick Reference*. If the query is successful, the first and last name of the contact is printed to the `System.out`.

If the query results in multiple records, the record set can be iterated as follows:

```
if (sbc_busComp.firstRecord()) {
//obtain the fields/values from this record
...
while (sbc_busComp.nextRecord()){
    //obtain the fields/values from this record
    ...
}
}
```

Code Sample Using Integration Object JavaBeans

The following is a code sample invoking the QueryByExample method of the Siebel Account Business Service. The JDBSiebelAccount class, which is described in the section below, provides code examples that show how to perform the following tasks:

- How to invoke a business service method, passing an integration object as a method parameter.
- Query for a list of Accounts that start with the letters “Ai.”
- Transform the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information see [“The Siebel JavaBean Wizard” on page 189](#), for both business services and integration objects. By using the code generation facilities, many of the complexities of the Siebel property sets and business service interfaces have been abstracted, providing a standards-based JavaBean interface.

```
/* import declaration here */

public class JDBSiebelAccount {
    public static void main(String[] args) {
        Siebel_AccountBusServAdapter    svc    = null;
        QueryByExampleInput              qbeIn   = new QueryByExampleInput();
        QueryByExampleOutput             qbeOut  = new QueryByExampleOutput();
        AccountIC                       acctIC  = new AccountIC();
        Sample_AccountIO                acctIO   = new Sample_AccountIO();
        Vector                           ioc      = null;

        /* query on accounts that start with 'Ai'. */
        acctIC.setName("Ai*");
        acctIO.addfintObjInst(acctIC);

        qbeIn.setfSiebelMessage(acctIO);

        try {
            /* initialize the Siebel Account service using the
             * user name, password, connect stringm language constructor. The
             * logoff method will be invoked when class destroyed
             */
            svc = new Siebel_AccountBusServAdapter("username", "password",
                "siebel://namesrvr/entserver/objectmanager/siebelserver",
                "lang");

            /* invoking QueryById method */
            qbeOut = svc.mQueryByExample(qbeIn);
        }
    }
}
```

```
acctIO = null;
acctIC = null;
acctIO = new Sample_AccountIO(qbeOut.getfSiebelMessage().
toPropertySet());
ioc     = acctIO.getfintObjInst();

/* loop through all accounts returned */
if(!ioc.isEmpty()) {
for(int i=0; i < ioc.size(); i++) {
acctIC = ((AccountIC)ioc.get(i));
System.out.println(i + " " + acctIC.getfName() + " " +
    acctIC.getfLocation());
}/* for */
}/* if */
}catch(SiebelException se) {
    System.out.println(se.getDetailedMessage());
}/* try-catch */
}/* main */
}/* public class JCASiebelAccount */
```

Code Sample Using the JavaBean Wizard

The following is a code example using the Java Data Bean and JavaBeans based upon the Sample Account Integration Object and EAI Siebel Adapter Business Service. The DataBeanDemo class, implemented below, provides code examples that show how to perform the following tasks:

- Initialize a business service using the Java Data Bean Constructor.
- Invoke the QueryPage method.
- Transform the results of the QueryPage method into an integration object.

The following code uses the code generation facilities provided in Siebel Tools. For more information see [“The Siebel JavaBean Wizard” on page 189](#) for both business services and integration objects. By using the code generation facilities, many of the complexities of various Siebel API's (PropertySet, Business Services) have been abstracted and encapsulated into a JavaBean based interface.

```
/* import declaration here */

public class DataBeanDemo {

    private String      userName    = "SADMIN";
    private String      passwd     = "SADMIN";
    private String      lang       = "enu";
    private String      connStr    = "siebel://trothwein2/siebel/SCCObjMgr_enu/
trothwein2";
    private String      intObjName = "Sample Account";
```

```

public static void main(String[] args) {
    DataBeanDemo demo = new DataBeanDemo();
} /* main */

public DataBeanDemo() {
    SiebelDataBean m_dataBean          = null;
    EAI_Siebel_AdapterBusServAdapter   svc = null;
    QueryPageInput qpInput              = new QueryPageInput();
    QueryPageOutput qpOutput            = new QueryPageOutput();
    int rowNum                         = 0;
    int pageSize                       = 10;
    Sample_AccountIO acctIO             = null;
    Vector ioc                         = null;

    try {
        /* instantiate the Siebel Data Bean */
        m_dataBean = new SiebelDataBean();

        /* login to Siebel using connect string, user name, password, lang
constructor */
        m_dataBean.login(connStr, userName, passwd, lang);

        /* construct the EAI Siebel Adapter, using the data bean constructor */
        svc = new EAI_Siebel_AdapterBusServAdapter(m_dataBean);
        svc.initialize();

        /* set QueryPage parameters to be used by the Siebel Adapter.QueryPage */
        /* method the request will return 10 records, starting at record 0, using */
        /* the Sample Account integration object */
        qpInput.setfPageSize(java.lang.Integer.toString(pageSize));
        qpInput.setfOutputIntObjectName(intObjName);
        qpInput.setfStartRowNum(java.lang.Integer.toString(rowNum));
        qpOutput = svc.mQueryPage(qpInput);

        /* construct the Integration Object using the output of the QueryPage */
        /* method, and the property set constructor*/
        acctIO = new
            Sample_AccountIO(qpOutput.getfSiebelMessage().toPropertySet());

        /* convert the results of the Integration Object to a vector for */
        /* processing */
        ioc = acctIO.getfintObjInst();

        /* Loop through the result set, printing out account name and location */
        if (!ioc.isEmpty()) {
            for (int i = 0; i < ioc.size(); i++) {
                AccountIC acctIC = ((AccountIC) ioc.get(i));
                System.out.println(acctIC.getfName() + " ----- " +
                    acctIC.getfLocation());
            } /* for */
        } /* if */
    }
}

```

```
/* logout once the request has been completed */
m_dataBean.logout();
} catch (SiebelException e) {
e.printStackTrace();
try {

    /* if an error occurs, ensure that the connection is closed */
    m_dataBean.logout();
} catch (SiebelException obj) {
    System.out.println(obj.getErrorMessage());
}/* try-catch */
}/* try-catch */
}/* public DataBeanDemo */
}/* public class DataBeanDemo */
```

The Siebel Resource Adapter

The Siebel Resource Adapter uses the Siebel JavaBean to call business services and functions as a point of contact between application components, application servers, and enterprise information systems, like Siebel eBusiness Applications. This adapter provides support for the J2EE Connector Architecture through system-level contracts. These contracts specify how a system external to the J2EE platform can integrate with it by supporting basic functions that are handled by the J2EE container.

Using the Resource Adapter

When deploying the Siebel Resource Adapter in BEA or WebLogic, the necessary Siebel and J2EE Connector JAR files are included in the server classpath. The Siebel JAR files that need to be added to the classpath are:

- SiebelJI.jar
- SiebelJI_< lang > .jar
- SiebelJI_Common.jar

Once you have added the necessary Siebel and J2EE Connector JAR files to server classpath, see the J2EE application server documentation with respect to deploying a Resource Adapter.

NOTE: Some client applications can require the Siebel Resource Adapter .jar file (Siebel_JCA_ra.jar) to be included in the classpath. The Siebel resource Adapter can be extracted from the resource archive (Siebel_JCA.jar) using the jar.exe utility provided with your Java Virtual Machine.

Java Data Bean/Siebel Resource Adapter and the siebel.properties File

The siebel.properties file, which is located in your classpath, can be used to provide default parameters for client applications connecting to Siebel applications using the Java Data Bean.

[Table 25](#) shows the properties in the siebel.properties file.

Table 25. Properties in the siebel.properties File

Property Type	Property	Description
Siebel Connection Manager Connection properties	siebel.conmgr.txttimeout	Indicates the transaction timeout (in seconds). Defaulted to 2700 = 45m.
	siebel.conmgr.poolsize	Indicates the connection pool size. Connection pool maintains a set of connections to a specific server process. Defaulted to 2. Max connection pool size is 500.
	siebel.conmgr.sesstimeout	Indicates the transaction timeout (in seconds) on the client side. Defaulted to 600 = 10m.
	siebel.conmgr.retry	Indicates the number of open session retries. Defaulted to 3.
	siebel.conmgr.jce	Indicates the usage of Java Cryptography Extension. 1 for jce usage and 0 for nonusage.

Table 25. Properties in the siebel.properties File

Property Type	Property	Description
Siebel Generated code for JCA/JDB properties	siebel.connection.string	Specifies the siebel connection string.
	siebel.user.name	Specifies the user name to be used for logging in to Object Manager.
	siebel.user.password	Specifies the password to be used for logging in to Object Manager.
	siebel.user.language	Specifies the user's preferred language.
	siebel.user.encrypted	Specifies whether the username and password is encrypted.
	siebel.jdb.classname	Specifies the default JDB classname.
Java System Properties	file.encoding	Indicates the code page on the client side. For example, cp1252, utf8, unicodeBig, cp942.

NOTE: Java System Properties are System Properties, not Siebel Properties.

Here is a sample siebel.properties file:

```
siebel.connection.string = siebel.tcpip.rsa.none://
test.siebel.com/siebel/sseobjmgr_enu/test
siebel.user.name        = User1
siebel.user.password    = password
siebel.user.language    = enu
siebel.user.encrypted   = false
siebel.conmgr.txttimeout = 3600
siebel.conmgr.poolsize  = 5
siebel.conmgr.sesstimeout = 300000
siebel.conmgr.retry     = 5
siebel.conmgr.jce       = 1
```

Using Java Code Generated by Siebel eBusiness Applications

The following two sections contain code samples for both managed and nonmanaged environments.

Managed Code Sample

The following is a code sample using the Siebel Resource Adapter in a managed environment. The SiebelAccountBean class, which is described in this section, provide code examples that show how to perform the following tasks:

- Configure the Siebel Resource Adapter by invoking methods on the Siebel_Account_BusServAdapter class.
- Query for a list of Accounts that start with the letters 'Ai'.
- Transform the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information, see [“The Siebel JavaBean Wizard” on page 189](#) for both business services and integration objects. By using the code generation facilities, many of the complexities of the J2EE Connector specification (Interactions, ConnectionSpec, CCI) have been abstracted and encapsulated into a consistent, JavaBean-based interface.

```
/* import declaration here */

public class SiebelBean implements SessionBean {
    protected static final String CONNECT_STRING = "siebel://name_server/
enterprise/objectmanager/siebel_server";
    protected static final String    JNDI_NAME      = "eis
SiebelResourceAdapterJNDIName";
    protected static final String    USER_NAME     = "SADMIN";
    protected static final String    PASSWORD       = "SADMIN";
    protected static final String    LANGUAGE       = "enu";

    /**
     * Get account list via query parameter
     */
    public Vector getAccountList(String queryParam) throws RemoteException {
        QueryByExampleInput    qbeIn  = new QueryByExampleInput();
        QueryByExampleOutput   qbeOut  = new QueryByExampleOutput();
        Sample_AccountIO       acctIO  = new Sample_AccountIO();
        AccountIC              acctIC  = new AccountIC();
        Vector                  ioc     = null;

        /* query on accounts based upon the inputs provided */
        acctIC.setName(queryParam);
        acctIO.addfintObjInst(acctIC);
    }
}
```

```

qbeIn.setfSiebelMessage(acctIO);

try {
    /* establish connection */
    getConnectionFactory(jndiName);
    /* initialize business service using default constructor */
    Siebel_AccountBusServAdapter svc = new Siebel_AccountBusServAdapter();

    /*
     * set connection specific user properties. if not specified, values
     * specified in siebel.properties will be used.
     */
    svc.setConnectString(CONNECT_STRING);
    svc.setUserName(USER_NAME);
    svc.setPassword(PASSWORD);
    svc.setLanguage(LANGUAGE);

    /* set the connection factory to be used for the request. */
    svc.setConnectionFactory(m_cf);

    /* invoke QueryByExample method */
    qbeOut = svc.mQueryByExample(qbeIn);

    acctIO = null;

    /* process output and return to client as a Vector */
    acctIO = new
        Sample_AccountIO(qbeOut.getfSiebelMessage().toPropertySet());
    ioc = acctIO.getfintObjInst();

    return ioc;
} catch (Exception e) {
    return ioc;
} /* try-catch */
} /* getAccountList(...) */

/* function to obtain a connectionFactory reference */
private void getConnectionFactory(String name) throws Exception {
    SiebelConnectionFactory    connectionFactory = null;
    InitialContext              context           = null;
    Object                      reference         = null;

    try {
        context = new InitialContext();
    } catch (Exception e) {
        System.out.println("Could not acquire the initial context");
    } /* try-catch */

    try {
        reference = context.lookup(name);
        connectionFactory = (SiebelConnectionFactory) PortableRemoteObject.narrow(reference, ConnectionFactory.class);
    } catch (Exception e) {
        System.out.println("Unable to obtain initial reference to connection factory");
    } /* try-catch */
}

```

```
m_ctx = context;
m_cf = connectionFactory;
}/* getConnectionFactory(...) */

public javax.ejb.SessionContext getSessionContext() { return mySessionCtx; }

public void setSessionContext(javax.ejb.SessionContext ctx) { mySessionCtx =
    ctx; }

public void ejbActivate() { }

public void ejbCreate() throws javax.ejb.CreateException { }

public void ejbPassivate() { }

public void ejbRemove() { }

protected SiebelConnectionFactory m_cf = null;
protected InitialContext m_ctx = null;
private javax.ejb.SessionContext mySessionCtx = null;
}/* public class SiebelBean implements SessionBean */
```

Nonmanaged Code Sample

The following is a code sample using the Siebel Resource Adapter in a nonmanaged environment. The JCASiebelAccount class (implemented below), provides code examples that show how to perform the following tasks:

- Configure the Siebel Resource Adapter by invoking methods on the Siebel_Account_BusServAdapter class.
- Query for a list of Accounts that start with the letters 'Ai'.
- Transform the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information, see [“The Siebel JavaBean Wizard” on page 189](#) for both business services and integration objects. By using the code generation facilities, many of the complexities of the J2EE Connector specification (Interactions, ConnectionSpec, CCI) have been abstracted and encapsulated into a consistent, JavaBean-based interface.

```
package com.siebel.service.jca.siebelaccount;

/* import required classes here */

public class JCASiebelAccount {
    public static void main(String[] args) {
        Siebel_AccountBusServAdapter svc = new
        Siebel_AccountBusServAdapter("siebel.properties");
```

```

QueryByExampleInput          qbeIn  = new QueryByExampleInput();
QueryByExampleOutput         qbeOut = new QueryByExampleOutput();
AccountIC                   acctIC  = new AccountIC();
Sample_AccountIO            acctIO  = new Sample_AccountIO();
SiebelConnectionFactory     m_cf   = null;
Vector                      ioc     = null;

/* query on accounts that start with 'Ai'. */
acctIC.setfName("Ai*");
acctIO.addfintObjInst(acctIC);

qbeIn.setfSiebelMessage(acctIO);

try {
    /* establish connection factory and connection */
    m_cf = new SiebelNoTxConnectionFactory(new
        SiebelNoTxManagedConnectionFactory());
    svc.setConnectionFactory(m_cf);

    /* invoking QueryById method */
    qbeOut = svc.mQueryByExample(qbeIn);

    acctIO = null;
    acctIC = null;
    acctIO = new Sample_AccountIO(qbeOut.getfSiebelMessage().toPropertySet());
    ioc    = acctIO.getfintObjInst();

    /* loop through all accounts returned */
    if(!ioc.isEmpty()) {
        for(int i=0; i < ioc.size(); i++) {
            acctIC = ((AccountIC)ioc.get(i));
            System.out.println(i + " " + acctIC.getfName() + " " +
                acctIC.getfLocation());
        } /* for */
    } /* if */
} catch(SiebelException se) {
    System.out.println(se.getDetailedMessage());
} /* try-catch */
} /* main */
} /* public class JCASiebelAccount */

```

Siebel Java/XML Framework

To further simplify client-side integration, Siebel applications provide a Java Framework designed to receive XML requests sent by Siebel applications over HTTP or MQSeries. The Java/XML Framework provides a uniform way to receive and process Siebel application requests within a J2EE environment.

CAUTION: You may use this code only to receive XML requests from Siebel applications. You may extend this code solely for use in object code form and solely for the purpose of integrating Siebel applications with non-Siebel applications. However, any modification or extension of this code is outside of the scope of Maintenance Services and will void all applicable warranties.

Requests initiated from within Siebel applications are transmitted to the appropriate J2EE Application Server using Siebel's eAI integration infrastructure, as illustrated in [Figure 15](#).

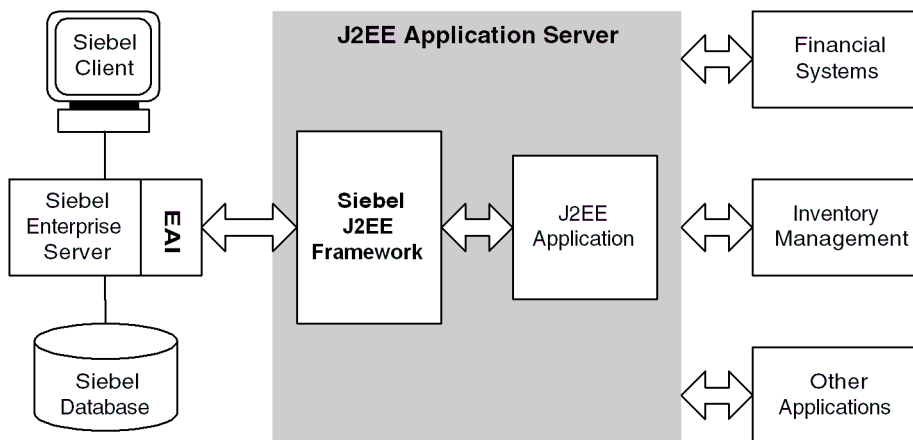


Figure 15. JavaBean Architecture

The Java/XML Framework consists of a Servlet to receive HTTP requests and an MQSeries Base Server designed to retrieve messages from an MQSeries queue. When implementing the Java/XML Framework, you will need to implement a single interface (ProcessRequest) responsible for understanding the contents of the incoming request and dispatching it to the appropriate Java component.

During application design, you can choose the transport that best achieves your integration requirements, as well as the structure of the XML document to be sent to the J2EE Application Server.

The Java/XML Framework is installed with Siebel Tools as part of the Siebel Java Integration component (Custom installation). The HTTP Servlet and MQSeries Base Server are installed in the <Siebel Tools>\CLASSES directory, and are packaged in the SiebelJI.jar file.

Sample implementations of the Siebel HTTP Base Servlet and the MQSeries Base Server may be found in these directories:

```
<Siebel Tools>\CLASSES\SRC\COM\SIEBEL\INTEGRATION\MQ
```

or

```
<Siebel Tools>\CLASSES\SRC\COM\SIEBEL\INTEGRATION\SERVLET
```


Index

A

Access. *See* Microsoft Access
Account_AttachmentIC.java class 193
Account_OrganizationIC.java class 193
AccountIC.java class 193
Active Server Pages 164, 174, 175
 Siebel OLE DB Provider, about
 accessing 89
 viewing Siebel application data, code
 sample 112
Active X Data Control, connect string 172
addfChildIntObjComp integration object
 component 196
addfIntObject method 194
AIC Inbound Transport
 about 157
 checklist 157
 installing 158
 receiving BizTalk messages, checklist
 for 166
 registering 158, 163
 Siebel Server setup checklist 161
 Siebel workflow information,
 supplying 145, 170, 173, 185
 XML documents, creating workflow
 for 166, 170
AMI receiver 19
AMI Repository 28
AMI, Application Messaging Interface 27
ANSI X12 format 121
Application Integration Components
 Inbound Transport. *See* AIC Inbound
 Transport
Application Messaging Interface 27
AppObjMgr 165

B

batch loading 22
BizTalk Server
 about 121
 about exposing integration objects 131
 channel, setting up 171, 184
 COM object definition 155
 configuration objects, setting up 152,
 153
 configuration objects, Siebel HTTP
 Outbound Transport 139, 176
 configuration recommendation 130
 data maps, about 171, 184
 document specifications, creating 136
 exposing integration objects,
 checklist 133
 features 122
 File Receive function, configuring 144,
 172, 184
 formats 121
 Interchange Submit parameter 151
 installation and document specification
 checklist 129
 interface architecture 124
 Messaging Manager, configuring 164
 port, setting up 170, 183
 receiving messages from 166
 required components for HTTP 174
 schemas importing 136
 Server DLL, copying to Siebel
 Server 162
 Siebel data type mapping 137
 Siebel HTTP Outbound Transport,
 checklist 142, 176
 SubmitSync parameter 151

- Type Library files, copying to Siebel Server 162
- business objects, SQL commands for querying 109
- business service user properties 15
 - as transport properties 15
- business services
 - See also* methods
 - adapter class, default Java methods 201
 - BizTalk COM object definition 155
 - eAI MSMQ transport 32
 - generated Java classes 200
 - input parameters, default Java methods 201
 - Java code directory structure 199
 - Java code, generating 198
 - JavaBeans, about generating 197
 - JavaBeans, use of 189
 - non-exportable (error message) 198
 - output parameters, default Java methods 201
 - parameter templates, using 77
 - setting up for EAI HTTP Transport 70, 71
 - transport methods 13
- Business_AddressIC.java class 193
- BusServAdapter method 201

C

- CC_XML_Converter BusServiceAdapter.java class 200
- CharsetConversion parameter 17
- clone integration object component 196
- clone method 194
- COM Outbound Transport
 - BizTalk communication link, establishing 149
 - checklist 148
 - parameters 151
 - sending messages, checklist 150
- COM + package 158, 163
- Command objects 109
- Command OLE DB Object 107
- CommandType property 101

- configuration files
 - EAI HTTP Transport 60
- configuration parameters, setting for application server 63
- configuring named subsystems 19
- connect string
 - local client mode, Siebel OLE DB Provider 92
 - Siebel OLE DB Provider 90
 - Siebel OLE DB Provider, server connected mode 91
 - TCP/IP example 63
- Connection property 102
- connection subsystem 19
- consumers. *See* OLE DB Consumers
- ContactIC.java class 194
- ConverterService parameter 17
- cookies 78
- creating a connection subsystem 19
- creating a data handling subsystem 20
- CSSHTTPTransService 57
- CSSHTTPTransService class 57

D

- data handling subsystem 20
- data mapping, OLE DB consumers 85
- data source initialization failure, troubleshooting 117
- data sources
 - testing OLE DB connection 90
- data transfer
 - eAI MSMQ send and receive scenarios 38
 - high volume 22
- DataSource objects
 - about 107
 - DataSource properties 108
 - Session object, relation to 108
 - supported interfaces 107
- DataSource OLE DB object 107
- DBPROP_AUTH_PASSWORD property 108
- DBPROP_AUTH_USERID OLE DB property 108

- DBPROP_INIT_DATASOURCE
 - property 108
- DBPROP_INIT_PROMPT property 108
- dead letter queue 31
- DispatchMethod parameter 17
- DispatchRuleSet parameter 17
- DispatchService parameter 18
- DispatchWorkflowProcess parameter 18
- distribution list, role in BizTalk
 - transport 128
- document specifications
 - checklist for creating 129
 - importing schema 136
- Document Type Definition (DTD) format
 - generating new schema 134, 135
 - importing into BizTalk, overview 125
- E**
- EAI BTS COM Transport 155
- EAI Connectors, BizTalk Server
 - integration 130
- EAI HTTP Transport
 - about 57
 - business service, setting up 70, 71
 - business services, about selecting 58
 - error handling 74
 - GET and POST method restrictions 59
 - inbound message parameters,
 - setting 62, 63
 - inbound, about 62
 - Send and SendReceive method
 - arguments 80, 83
 - sending and receiving messages,
 - workflows 74
 - sending messages, process overview 61
 - Siebel components, required 58
 - usage checklist 70
 - workflow process, creating 71, 73
- EAI HTTP Transport, outbound
 - client-side parameters, specifying 76
 - HTTP parameter templates 77
 - HTTP parameters as run-time
 - properties 77
 - server-side parameters, specifying 76
 - specifying parameters, about 75
- EAI HTTP Transport, Session Cookie mode
 - inbound transport 64
 - login and logoff examples 78, 79
 - login examples 65, 66
 - logoff example 67
- EAI HTTP Transport, Sessionless mode
 - examples 79
 - login example 67
 - variables 68
- EAI MQSeries Transport
 - configuring 24
- eAI MSMQ transport
 - See also* MSMQ transport
 - about 32
 - configuration parameters 38
 - integration objects, about 39
 - messages, receiving 47, 49
 - messages, receiving and dispatching 50, 52
 - messages, receiving, dispatching, and
 - sending 52, 55
 - prerequisites 38
 - queue name parameter 38
 - receiving messages, methods 46
 - receiving messages, overview 33
 - sending and receiving messages,
 - procedure 44, 46
 - sending messages, overview 33, 40
 - sending messages, procedure 41, 43
 - servers, about configuring 35
 - size limitations 40
 - transport modes 32
 - wait time parameter 38
- EAI XML Converter, role in HTTP
 - messages 61
- eai.cfg 60
- EAITransportDataHandlingSubsys 20
- eapps.cfg, configuring for HTTP inbound
 - messages 62, 63
- EDIFACT format 121
- Edit OLE DB Rowset wizard 100

Electronic Data Exchange (EDI)
 format 121
Electronic Data Interchange for
 Administration, Commerce and
 Transport (EDIFACT) format 121
EndOfData parameter 38
Enterprise Java Beans (EJBs), code
 sample 205
equals integration object component 196
equals method 194
Excel. *See* Microsoft Excel
external OLE DB-enabled applications. *See*
 Microsoft OLE DB

F

fromPropertySet integration object
 component 196
fromPropertySet method 194, 201

G

GET
 HTTP method 59
 Session Cookie mode login example 65,
 66
getfChildIntOb integration object
 component 196
getfFieldName integration object
 component 196
getfIntObjectFormat method 194
getfIntObjectName method 194
getfIntObjInst method 194
getfMessageId method 195
getfMessageType method 195
getfMethodArgName method 201, 202
getfOutputIntObjectName method 195

H

HTTP Body 59
HTTP methods 59
HTTP Session Cookie mode
 about 64
 login and logoff examples 78, 79

 login examples 65, 66
 logoff example 67
HTTP Sessionless mode
 examples 79
 login example 67
 variables 68
HTTPAllowCaching parameter 82
HTTPAllowPersistentCookies
 parameter 82
HTTPIsSecureConn parameter 82
HTTPLoginBodyTemplate parameter 81
HTTPLoginMethod parameter 80
HTTPLoginURLTemplate parameter 80
HTTPLogoffMethod parameter 81
HTTPLogoffURLTemplate parameter 81
HTTPMaxIdleSeconds parameter 81
HTTPNoAutoRedirect parameter 83
HTTPRequestBodyTemplate parameter 80
HTTPRequestMethod parameter 80
HTTPRequestURLTemplate parameter 80

I

IAccessor interface 109
IBM MQSeries 24
IColumnsInfo interface 109, 110
ICommand interface 109
ICommandProperties interface 109
ICommandText interface 109
ICommandWithParameters interface 109
IConvertType interface 109, 110
IDBCreateCommand interface 108
IDBCreateSession interface 107
IDBInitialize OLE DB interface 107
IDBProperties OLE DB interface 107
IDBSchemaRowset interface 108
IGetDataSource interface 108
IgnoreCharSetConvErrors parameter 18
IIS 89, 174
inbound methods 14
inbound transport, HTTP Session Cookie
 mode
 about 64
 login and logoff examples 78, 79

- login examples 65, 66
 - logoff example 67
 - inbound transport, HTTP Sessionless mode
 - examples 79
 - login example 67
 - variables 68
 - InputBizdoc process property 168
 - Integrated Development Environment (IDE) 205
 - integration objects
 - about exposing 131
 - checklist for creating 129
 - configuring Siebel Server workflow 164
 - creating 39
 - default Java methods 194, 195, 196
 - exposing, checklist for 133
 - generating Java code 191, 192
 - generating new schema 134, 135
 - Java classes, sample 193, 194
 - Java code directory structure 192
 - JavaBeans, use of 189
 - OLE DB differences in 99
 - OLE DB rowsets 96
 - relation to OLE DB rowsets 87
 - Siebel OLE DB Rowset object,
 - creating 97, 99
 - for trading partners 133
 - Integration_ObjectIO method 195
 - IntObjCompIC integration object
 - component 196
 - IOpenRowset interface 108
 - IPersist OLE DB interface 107
 - IRowset interface 110
 - IRowsetInfo interface 110
 - ISessionProperties interface 108
 - ISupportErrorInfo interface 108, 109
 - ISupportErrorInfo OLE DB interface 107
- J**
- J2EE environment
 - application server integration architecture
 - diagram 188
 - .jar files 205
 - Java/XML Framework, about 218
 - .jar files 205
 - Java classes
 - CC XML, generated for 200
 - sample structure 193, 194
 - Java code
 - business services objects, directory
 - structure 199
 - business services, generating for 198
 - documentation 205
 - .jar files 205
 - Siebel Java Data Bean in a Servlet,
 - sample 205, 206
 - Java code, default methods
 - business service adapter class 201
 - business service input parameters 201
 - business service output parameters 201
 - Java code, integration objects
 - directory structure 192
 - generating 191, 192
 - methods, default 194, 195, 196
 - Java Data Beans. *See* Siebel Java Data Beans
 - Java Server Pages (JSPs), code sample 205
 - Java/XML Framework, about 219
 - JavaBean Wizard 190
 - JavaBeans
 - about generating for business
 - services 197
 - architecture diagram 219
 - creation wizard 189
 - directory structure example 192
 - JavaBean Wizard 190
 - javac.exe 204
 - javadoc zip file 205

L

 - local client mode 92

M

 - mBusSvcMethodName method 201
 - methodInput method 201
 - methodOutput method 201

- methods
 - business service adapter class, default
 - Java methods 201
 - default business service input
 - parameters, Java 201
 - default business service output
 - parameters, Java 201
 - GET 59
 - inbound methods 14
 - Java integration object methods
 - (default) 194, 195
 - outbound methods, overview 13
 - POST 59
 - Send 13
 - transport methods, about 13
 - Microsoft Access
 - about accessing Siebel OLE DB
 - Provider 89
 - Data Access Page Designer 102
 - Siebel application data, viewing 102, 103
 - Microsoft ADO version 2.1 89
 - Microsoft BizTalk Server 2000. *See* BizTalk Server
 - Microsoft Excel
 - about accessing Siebel OLE DB
 - Provider 89
 - query properties 101
 - Siebel application data, viewing 101
 - Microsoft Excel 2000, OLE DB connection to 91
 - Microsoft Internet Information Server (IIS)
 - BizTalk Server 174
 - OLE DB Consumer 89
 - Microsoft Message Queuing Transport. *See* MSMQ Transport
 - Microsoft OLE DB
 - See also* Siebel OLE DB Provider
 - about 85
 - supported foundation version 89
 - Microsoft SQL Server, distributed
 - queries 103
 - Microsoft Visual Basic, code sample 110
 - model queue 28
 - MqLocalHostFileName 19
 - MqModelQueueName argument 25
 - MqPhysicalQueueName parameter 25
 - MqQueueManagerName parameter 25
 - MqRespModelQueueName parameter 25
 - MqRespPhysicalQueueName
 - parameter 25
 - MQSeries 28
 - server component 26
 - MQSeries AMI policy 28
 - MQSeries AMI Receiver 19
 - MQSeries Transport 27
 - MQSeriesAMISubsys 19
 - MSMQ transport
 - See also* eAI MSMQ transport
 - about 31
 - client setup 37
 - NT 4.0 Server setup 36
 - primary enterprise controller (PEC)
 - setup 36
 - size limitations 40
 - MsmqPhysicalQueueName parameter 38
 - MsmqQueueMachineName parameter 38
 - MsmqRespQueueMachineName
 - parameter 38
 - MsmqRespQueueName parameter 38
 - multivalue fields 95
- N**
- named subsystems 19
 - about 15
 - common parameters 16, 18
 - compatibility issues 60
 - EAI HTTP Transport, use in 60
 - precedence rules 16
 - relation to server components 15
 - named subsystems in a workflow 21
 - NT 4.0 Server, MSMQ transport setup 36
- O**
- object interfaces, about 22
 - OLE DB Consumers
 - applications supported 96
 - defined 85
 - multivalue fields, use of 95
 - retrieving data, about 96

- OLE DB Provider
 - connecting to from external applications 86
 - IDBCreateCommand interface 108
 - IDBCreateSession 107
 - IDBInitialize 107
 - IDBProperties interface 107
 - IPersist interface 107
 - ISupportErrorInfo interface 107
 - siebel.udl file 90
- OLE DB rowsets. *See* rowsets; Siebel OLE DB Rowset
- Outbound EAI HTTP Transport business service. *See* EAI HTTP Transport, outbound
- outbound methods, overview 13
- P**
- parameter templates 77
- parameters
 - HTTPLoginURLTemplate 80
 - MqModelQueueName 25
 - MqPhysicalQueueName 25
 - MqQueueManagerName 25
 - MqRespModelQueueName 25
 - MqRespPhysicalQueueName 25
 - specifying as run-time properties 77
- Polling Location 145, 173, 185
- POST
 - HTTP method 59
 - Session Cookie Mode login example 65, 66
- primary enterprise controller (PEC)
 - setup 36
- PropSetToXMLInput.java class 200
- PropSetToXMLOutput.java class 200
- Q**
- queries
 - Command object, creating 109
 - distributed 103
 - Microsoft Excel query properties 101
 - queries against one virtual table 109
 - query file, Siebel OLE DB Provider 101
 - Siebel OLE DB Provider, using 101, 102, 103
- QueryText property 102
- QueryType property 101
- queue manager 28
- R**
- Receiver Server 20
- regsvr32 utility 162
- regtlb utility 162
- RollbackOnDispatchError parameter 18
- Rowset OLE DB Object 107
- rowsets
 - See also* Siebel OLE DB Rowset
 - modifying rowset object 100
 - naming convention 98
 - relation to integration objects 87
 - resequencing field names 99
 - retrieving data, about 96
 - Rowset object 110
 - rowset object, creating 97, 99
 - Session objects, relation to 108
 - visibility level 98
- .rqy file 101
- run-time properties, parameters as 77
- S**
- Sample_AccountIO.java class 194
- SAP Connector, business service user
 - properties 15
- Schema Generator Wizard 131
- schemas
 - formats 125
 - generating 134, 135
 - importing 136
- SDBJavaDoc.zip 205
- Send method 13, 80, 83
- SendReceive method 80, 83
- server components
 - MQSeries 26

- server connected mode, Siebel OLE DB Provider 91
- Server: Msg 7357 119
- Server: Msg 7399 118
- service point 28
- Servlet, sample Java code 205, 206
- session cookies 78
- session mode 78
- Session object 108
- Session OLE DB object 107
- setfFieldName integration object component 196
- setfIntObjectFormat method 195
- setfIntObjectName method 195
- setfMessageId method 195
- setfMessageType method 195
- setfMethodArgName method 201, 202
- setfOutputIntObjectName method 195
- Siebel ActiveX Data Controls, installing 158
- Siebel AIC BTS 171, 183
- Siebel AIC, Siebel Server Workflow configuration 164
- Siebel application data
 - viewing in Microsoft Access 102, 103
 - viewing in Microsoft Excel 101
 - viewing in SQL Analyzer 104, 105
- Siebel Application Server, setting configuration parameters 63
- Siebel BizTalk adapter
 - business scenarios 128
 - COM Outbound Siebel workflow, creating 154, 156
 - generating new schema 134, 135
 - inbound and outbound interfacing processes 126
 - interface file installation 130
 - protocols supported 123
 - supported data formats 126
 - supported transport protocols 126
 - XDR data type mappings 137
- Siebel Client
 - COM Outbound workflow, creating 154, 156
 - configuration recommendation 130
 - Siebel database, writing XML document to 168, 170
 - Siebel eAI Transports. *See* transports
 - Siebel HTTP Inbound Transport
 - about 180
 - receiving messages from BizTalk, checklist 182
 - Siebel Server, about configuring 180
 - Siebel HTTP Outbound Transport
 - ASP files, copying to directory 175
 - BizTalk configuration objects, setting up 139, 176
 - BizTalk, sending messages, checklist 142, 176
 - checklist 174
 - Siebel workflow, creating 141, 142, 147, 178, 179
 - virtual directory, creating 175
 - Siebel Integration Objects. *See* integration objects; Java code, integration objects
 - Siebel Java Data Bean
 - running 205
 - sample code (Servlet) 205, 206
 - Siebel OLE DB Provider
 - See also* Microsoft OLE DB
 - about 85
 - about accessing from ASP 89
 - about accessing from Excel or Access 89
 - about accessing from SQL Server 89
 - architecture diagram 88
 - business scenarios 86
 - Command object 109
 - connection properties 90, 92
 - data mapping 85
 - default interface, applications 89
 - DLL properties, viewing 95
 - events, viewing 94
 - foreign keys, using 93
 - local client mode, about 92
 - Microsoft SQL, using with 89

- MS SQL Server and distributed queries 103
- multivalue fields 95
- primary keys, using 93
- read-only status 96
- Rowset object 110
- .rqy file 101
- scripting languages supported 106
- server connected mode, about 91
- Session object 108
- Siebel clients 88
- Siebel Mobile Client 89
- Siebel Server 89
- siebel.udl file 90
- supported DataSource objects 107
- supported objects, listed 107
- test file 90
- testing connection 90
- virtual OLE DB tables 86
- visibility levels 98
- Windows client 87
- Windows server 87
- Siebel OLE DB Provider Command object 109
- Siebel OLE DB Rowset
 - See also* rowsets
 - differences in integration objects 99
 - modifying rowset object 100
 - rowset object, creating 97, 99
 - wizard, about 96
- Siebel Server
 - AIC Inbound Transport, about 157
 - AIC Inbound Transport, setup checklist 161
 - BizTalk Server DLL, copying 162
 - COM+ remote communication, creating 163
 - configuration recommendation 130
 - configuring for integration component workflow 164
 - Type Library files, copying 162
- Siebel Tools
 - business services, role in setting up 70
 - DTD schemas, role in generating 134
 - eAI MSMQ transport, about
 - customizing 32
 - exposing integration objects to BizTalk Server 125
 - information objects, about defining 39
 - integration objects for BizTalk Server 131
 - JavaBeans, generating 189, 191
 - messages, configuring integration objects 33
 - outbound HTTP Transport messages 75
 - role in creating inbound HTTP transport messages 70
 - role in specifying business service user properties 75
 - schema, about generating 125
 - setup for EAI MSMQ Transport Server 35
 - Siebel OLE DB rowset object, creating 97
 - Siebel OLE DB rowset object, modifying 100
 - Siebel OLE DB rowset, role in defining 87
 - XDR schemas, role in generating 134
- Siebel Web client, server connected mode, Siebel OLE DB Provider 91
- Siebel Web Engine, configuring for HTTP inbound messages 62, 63
- Siebel workflows
 - See also* workflows
 - COM Outbound, creating 154, 156
 - creating for Siebel HTTP Outbound Transport 141, 142, 147, 178, 179
 - Siebel Server, configuring for integration objects 164
 - Siebel Workflow Process Manager 38
 - workflow information, supplying for AIC Inbound Transport 145, 170, 173, 185
- siebel.udl file 90
- SiebelJI_%lang%.jar 203

- SiebelJI_common.jar 203
- SiebelMessage process property 168
- SleepTime parameter 38, 83
- specifying named subsystems in a workflow 21
- SQL Analyzer, viewing Siebel Application data 104, 105
- SQL commands, supported 109
- SQL Query Analyzer message 7399, troubleshooting 118
- SQL Server, about accessing Siebel OLE DB Provider 89
- sscaeiba.dll 160

T

- test connection, troubleshooting 118
- TimedOut parameter 38
- toPropertySet integration object component 196
- toPropertySet method 195, 201
- trading partners
 - BizTalk Server scenario 128
 - integration objects for 133
- transport adapters
 - advantages of 12
 - business service methods, about 13
 - inbound methods 14
 - outbound methods, overview 13
- transport parameters
 - named subsystems, about 15
 - parameter specification precedence 16
- transports
 - about 12
 - common parameters for methods 16, 18
 - data exchange mechanisms, listed 11
 - methods and parameters, about specifying 15
 - role of 11
- troubleshooting
 - data source initialization failure 117
 - Server: Msg 7357 119
 - Server: Msg 7399 118
 - test connection failure 118

U

- Upsert method 170

V

- Version property 101
- virtual tables, querying 109
- Visual Basic, code sample 110

W

- WebDAV BizTalk Repository 133
- Windows Event Viewer, using 94
- workflow 21
- workflows
 - See also* Siebel workflows
 - EAI HTTP Transport, sending messages 61
 - EAI HTTP Transport, setting up for 71, 73
 - HTTP messages, sending and receiving 74
 - integration objects, creating 39
 - Siebel Workflow Designer, sending EAI HTTP messages 61
 - www.myserver.com 67

X

- XDR format
 - data type mappings 137
 - generating new schema 134, 135
 - importing into BizTalk, overview 125
- XML
 - CC XML, Java classes 200
 - Java/XML Framework 218
 - role in BizTalk Server 122
 - workflow for inbound documents 166, 170
- XMLToPropSet method 58, 200
- XMLToPropSetInput.java class 200