



**Transports and Interfaces:
Siebel eBusiness
Application Integration
Volume III**

Version 7.7
August 2004

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2004 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.

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

Other product names, designations, logos, and symbols may be trademarks or registered trademarks of their respective owners.

PRODUCT MODULES AND OPTIONS. This guide contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel's Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

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

Chapter 1: What's New in This Release

Chapter 2: EAI Transports and Interfaces Overview

About EAI Transports 11

Using Named Subsystems for Transport Parameters 13

 Rules of Precedence for Parameter Specification 14

 Common EAI Transport Parameters 15

Configuring Named Subsystems and Receiver Server Components 17

About Object Interfaces and EAI 19

Database Level Interfacing 19

Chapter 3: EAI MQSeries Transports

About Siebel EAI MQSeries Transports 21

 EAI MQSeries Transport Named Subsystems 21

 About the EAI MQSeries Server Transport 22

 About the EAI MQSeries AMI Transport 27

 About the EAI MQSeries Transport Re-Entrance 29

About Message Id Tracking for an Inbound Message 30

Invoking a Workflow Process Using MQSeries Server Receiver 30

Chapter 4: EAI MSMQ Transport

About MSMQ 33

 About EAI MSMQ Transport 33

 Methods for Sending and Receiving Messages 34

 EAI MSMQ Transport Named Subsystems 35

Configuring the EAI MSMQ Transport Servers 35

 MSMQ Primary Enterprise Controller 35

 Regional Enterprise Server and MSMQ Client 35

Configuring EAI MSMQ Transport for Various Send and Receive Scenarios 36

 Sending Outbound Messages with EAI MSMQ Transport 38

 Receiving Inbound Messages with EAI MSMQ Transport 45

Chapter 5: Java Business Service

- About the EAI Java Business Service 53
- Prerequisites for Implementing a JBS 53
- Creating a Java Business Service 55
 - Implementing a Business Service in Java 56
 - About Exception Handling for the Java Business Service 56
- About the Lifecycle of a Java Business Service 57
- Example of a Java Business Service 57
- About Restrictions for Implementing JBS 58
- Troubleshooting the Java Business Service 58

Chapter 6: Java Message Service Transport

- About the Siebel EAI JMS Transport 59
- About Synchronous and Asynchronous Invocation 60
- About the JMS Publish-and-Subscribe Model 60
- About Operations of the JMS Transport 60
- Features Not Supported for Use with the Siebel JMS Transport 61
- About JMS Message Types 61
- About Sending and Receiving XML 62
- About Multi-Step Operations Within a JMS Session 62
- Undeliverable Messages in JMS Transport 63
- Detailed Input and Output Specifications 63
 - JMS Headers and Properties 63
 - About Input Arguments Used by the Dispatch Step 64
 - About Dispatch Input Arguments 65
 - About Output of the JMS Transport 66
- Configuring the JMS Transport 67
 - About the JMSSubsys Named Subsystem 67
 - About the JMS Receiver 68
 - Creating a JMS Subsystem by Using the Siebel Web Client 68
 - Sending and Receiving Messages with the JMS Transport 69
- Receiving, Dispatching, and Sending JMS Messages 72
- Troubleshooting for the JMS Transport 75
- About Logging for the JMS Transport 77

Chapter 7: EAI HTTP Transport

- About the EAI HTTP Transport 79
 - System Requirements for Using the EAI HTTP Transport 80
 - Selecting the Appropriate Business Service for HTTP 80
- Using POST and GET 80
- EAI HTTP Transport Named Subsystems 81
- General Information on How to Send a Message 81
- Using the EAI HTTP Transport for Inbound Integration 84
- EAI HTTP Transport for Inbound Messages 90
- Handling EAI HTTP Transport Business Service Errors 93
- Processing and Sending Outbound XML Documents 93
- Sending and Receiving Messages with the EAI HTTP Transport 95
- Examples Using HTTP Request 98
- Creating Custom Headers for the EAI HTTP Transport Service 99
- About Sending and Receiving Messages through HTTP 100
- About Transport Headers and HTTP Response Headers 100
- EAI HTTP Transport Method Arguments 101

Chapter 8: EAI DLL and EAI File Transports

- About EAI DLL Transport 107
 - Configuring the EAI DLL Transport 107
 - About EAI DLL Transport Parameters 107
 - Creating a DLL to Call a Function in an External DLL 108
- About the EAI File Transport 109
 - Configuring the EAI File Transport 109
 - Using the EAI File Transport Methods 110
 - Generating Unique Filenames 110
 - EAI File Transport Parameters 111

Chapter 9: Using Siebel OLE DB Provider

- About Microsoft OLE DB 113
- About the Siebel OLE DB Provider 113
- About the Software Architecture for the Siebel OLE DB Provider 114
- Installing the Siebel OLE DB Provider 115
 - Configuring and Testing the Siebel OLE DB Provider 116

Contents

Multiple Language Considerations for the Siebel OLE DB Provider	118
About Primary and Foreign Key Relationships	119
Viewing Siebel OLE DB Provider Events	119
Viewing Siebel OLE DB Information	120
Connecting Siebel Data Using OLE DB Consumers	120
Creating and Modifying Siebel OLE DB Rowsets	121
Viewing Siebel OLE DB Rowsets in Microsoft Office Applications	123
How Scripts and Custom Applications Affect Your Data	127
Writing an OLE DB Consumer	128
Retrieving Siebel Data Using VB and ASP	130
Troubleshooting OLE DB	138

Chapter 10: Interfacing with Microsoft BizTalk Server

About Microsoft BizTalk Server	143
Siebel BizTalk Server Adapter	143
Where to Get More Information	144
Siebel BizTalk Interface Architecture	145
Schema Generation Support	146
Exchanging Integration Messages	146
Understanding Siebel BizTalk Server Adapter Through Scenarios	147
Preparing to Use the Siebel BizTalk Adapter	148
Installing and Configuring Software for Servers and Clients	149
Siebel Integration Objects	150
Connecting to BizTalk Using EAI MSMQ Transport	154
Using EAI MSMQ Transport for Outbound Messaging	154
Using EAI MSMQ Transport for Inbound Messages	158
Connecting to BizTalk Using COM and AIC	162
Siebel COM Outbound Transport	162
Siebel AIC Inbound Transport	167
Connecting to BizTalk Using HTTP	176
Siebel HTTP Outbound Transport	177
How to Use EAI HTTP Transport to Receive Documents from BizTalk	181

Chapter 11: Integrating with Java

About Siebel Applications and Java	185
About the JDB Business Service API	186
About the Siebel Code Generator	187
Invoking the Siebel Code Generator	188

Code Generated for a Business Service	188
About Methods of Java Classes Generated for a Business Service	191
About the Code Generated for an Integration Object	192
About Running the Java Data Bean	195
Connect String and Credentials for the SiebelDataBean	195
Connection Parameters for the SiebelDataBean	196
Examples Using Generated Code for Integration Objects	198
About the Siebel Resource Adapter	200
Using the Resource Adapter	200
About the Connect String and Credentials for the Java Connector	201

Index

1

What's New in This Release

What's New in Transports and Interfaces: Siebel eBusiness Application Integration Volume III, Version 7.7

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

Table 1. New Features in Transports and Interfaces: Siebel eBusiness Application Integration Volume III, Version 7.7

Topic	Description
"Common EAI Transport Parameters" on page 15	Information on the Impersonate parameter added.
Chapter 5, "Java Business Service"	A Java Business Service is a framework that provides the ability to invoke an external Java component.
Chapter 6, "Java Message Service Transport"	The Java Message Service transport is an API for accessing enterprise messaging systems. It is part of the Java 2 Platform, Enterprise Edition (J2EE).
Chapter 11, "Integrating with Java"	Chapter is revised placing emphasis on integrating with Java.

2

EAI Transports and Interfaces Overview

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

This chapter includes the following topics on these mechanisms:

- [“About EAI Transports” on page 11](#)
- [“Using Named Subsystems for Transport Parameters” on page 13](#)
- [“Configuring Named Subsystems and Receiver Server Components” on page 17](#)
- [“About Object Interfaces and EAI” on page 19](#)
- [“Database Level Interfacing” on page 19](#)

About 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 and higher, 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. Handling the data as binary 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 leads to an XML string in the local code page, while its header still describes UTF-8. You should treat all self-describing data including XML as binary.

Character conversion argument is available in a number of business services. These business services are:

- EAI Transport business services (MQ Series, DLL, File, HTTP, MSMQ)
- XML Converter business services
- Transcode business service

When business services are invoked from a workflow process, the valid set of encodings is controlled by a picklist. If the business services are invoked through scripting or similar mechanisms, the character set name is supplied textually.

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, its method and supported character sets, see *Global Deployment Guide*.

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.

Available transports include:

- EAI MQSeries Server Transport and EAI MQSeries AMI Transport. For information on these transports, see [Chapter 3, "EAI MQSeries Transports."](#)
- EAI MSMQ Transport. For information on this transport, see [Chapter 4, "EAI MSMQ Transport."](#)
- EAI HTTP Transport. For information on this transport, see [Chapter 7, "EAI HTTP Transport."](#)
- EAI DLL Transport and EAI File Transport. For information on these transports, see [Chapter 8, "EAI DLL and EAI File Transports."](#)

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 ["About the EAI MQSeries Transport Re-Entrance"](#) on page 29.

About 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 3 on page 15](#), as well as transport-specific parameters that are discussed in the respective chapter for each transport.

Outbound Methods for a Transport Business Service

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 the Siebel application 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 a Siebel application to another application:

- **Send.** Sends a message from a Siebel application when the Siebel application does not need a response. This is an asynchronous request method (with the exception of the EAI HTTP Transport, which expects a correct HTTP response), because the Siebel application does not need to wait for a response before continuing with the process.
- **Send and Receive.** Sends a message from the Siebel application when the 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.

Inbound Methods for a Transport Business Service

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. It 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. To suppress the response, you can create an output property, on the dispatch service, of type EmptyResponse and set it to True.

NOTE: There are server components (called receivers) on top of the inbound methods that run as Siebel Server tasks. When running an EAI receiver such as the SAP IDOC Receiver, MQSeries Server, MQSeries AMI Receiver, or MSMQ Receiver—using the methods ReceiveDispatch or ReceiveDispatchSend—if 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 Name 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.

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 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 work for the SAP Connector and the Oracle Connector business services.

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 run-time arguments. If you specify the parameters in both locations, the business service user property takes precedence over the run-time arguments.

NOTE: For additional information on named subsystems, see *Siebel System Administration Guide*.

You specify every other parameter in one of the two named subsystems or as run-time arguments. Siebel eAI looks for the parameter in the `ConnectionSubsystem` or the `DataHandlingSubsystem`, 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 run-time argument, the run-time specification will be ignored. Siebel eAI looks for the parameter in a run-time specification only if no appropriate named subsystem is specified.

Common EAI Transport Parameters

To configure the EAI Transports, you need to create named subsystems for data handling and connection parameters, as presented in Table 2. The data handling parameters are presented in Table 3. These parameters are common to every Transport method. After you create the named subsystems, you then need to specify these named subsystems as parameters in the service method argument or the business service user property.

Table 2. Dispatch Parameter Usage

When You Need to...	Use This Parameter...
...call any Business Service	DispatchService. This parameter must be used in conjunction with DispatchMethod.
...call any Business Service	DispatchMethod. This parameter must be used in conjunction with DispatchService.
...call the Dispatch Rule Set Business Service	DispatchRuleSet.
...call any Workflow	DispatchWorkflowProcess.

Table 3. Common Data Handling Parameters for Transport Methods

Parameter Name	Description
CharSetConversion	<ul style="list-style-type: none"> ■ Default is None. The default value for this parameter 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. <p>Depending on the value of this parameter, transport business services do implicit character set conversions if necessary. Note that same CharSetConversion is assumed for requests and responses.</p>
ConverterService	Default is EAI XML Converter. This is the name of the business service to use for serializing property sets to a buffer and unserializing buffers to property sets. This parameter 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.

Table 3. Common Data Handling Parameters for Transport Methods

Parameter Name	Description
DispatchMethod	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	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.
DispatchService	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.
DispatchWorkflowProcess	DispatchWorkflowProcess 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.
IgnoreCharSetConvErrors	Default is False. This parameter specifies whether character set conversion errors should be ignored. If False, with any such errors, the transport service propagates the error.
Impersonate	Default is False. This parameter indicates whether or not the receiver will execute the incoming request using the default credentials of the receiver or those provided in the incoming XML document. If this parameter is set to True, the receiver will introspect the incoming XML document (<SiebelMessage> element) for the eaiusername and eaipassword credential attributes. If these credentials are found, the receive will attempt to relogin with the credential. If the Impersonate parameter is set to True and the credentials are not found or are not a valid Siebel username or password, an error message will be returned.

Table 3. Common Data Handling Parameters for Transport Methods

Parameter Name	Description
RollbackOnDispatchError	Default is True. This parameter indicates whether or not to roll back transport transaction if a Dispatch Method fails. This parameter is only available for the transactional transports MQSeries Server, MQSeries AMI, and MSMQ.
SiebelTransactions	Default is True. This parameter indicates whether or not to nest the Siebel transaction within the transport transaction. This parameter is only available for the transactional transports MQSeries Server, MQSeries AMI, and MSMQ. If this parameter is set to False, the transaction support is turned off at the transport level. This setting means that if the transaction fails, then there will not be a rollback at the Siebel transaction level.

Configuring Named Subsystems and Receiver Server Components

This section explains how to configure named subsystems for the MQSeries AMI Receiver server component.

To configure the Default Tasks parameter

- 1** From the Site Map choose Administration - Server Configuration > Enterprises.
Select the Component Group tab.
- 2** From the Component Group list applet select Enterprise Application Integration.
A list of receivers appears in the Components applet, for example MQSeries AMI Receiver.
- 3** Select the MQSeries AMI Receiver in the Components list applet.
- 4** Click the Parameters tab.
- 5** Query for the Default Tasks parameter in the Enterprise Parameters applet.
- 6** 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** From the Site Map choose Administration - Server Configuration > Enterprises.
- 2** Click the Profile Configuration tab.
- 3** In the Profile list applet, create a new Named Subsystem record.
 - a** Enter MyAMISubsys in the Alias column.
 - b** Select MQSeriesAMISubsys as the Subsystem Type.

- c In the Profile Parameters list applet, specify the values for each of the connection parameters for the AMI receiver, such as MqLocalHostFileName.

To create a data handling subsystem

- 1 From the Site Map choose Administration - Server Configuration > Enterprises.
- 2 Click the Profile Configuration tab.
- 3 In the Profile list applet, create a new Named Subsystem record.
 - a Enter MyDispSubsys in the Alias column.
 - b Select EAITransportDataHandlingSubsys as the Subsystem Type.
- 4 In the Profile Parameters list 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 From the Site Map choose Administration - Server Management > Servers.
- 2 Click the Components Groups tab and select the Enterprise Application Integration component group.
- 3 In the Components list applet select the MQSeries AMI Receiver.
- 4 Click the Shutdown button.

Wait for the component state to change to Shutdown.
- 5 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 From the Site Map choose Administration - Business Process > Workflow Processes.
- 2 Query and select the workflow process named MyMQOutbound.
- 3 Double-click the MQ Send step.
- 4 In the Process Definition applet, insert a new record to specify the Connection subsystem, and specify the value as MyAMISubsys.
- 5 Insert another input argument to specify the Data Handling Subsystem, and specify the value as MyDispSubsys.
- 6 Insert a third record to specify the Message Text.

About Object Interfaces and EAI

Object Interfaces allow integration between the Siebel application and external applications. Object Interfaces can be called by eScripts and VB or used within a workflow. The workflow can use other business services and transports as needed. For example, the BizTalk Adapter 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 9, "Using Siebel OLE DB Provider."](#)
- Siebel eBusiness Applications interfacing with Microsoft BizTalk. For information, see [Chapter 10, "Interfacing with Microsoft BizTalk Server."](#)
- Siebel eBusiness Applications that produce Java Data Beans to support J2EE applications. For information, see [Chapter 11, "Integrating with Java."](#)

Database Level Interfacing

In addition to Transports and Object Interfaces, Siebel applications provide 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.

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

3

EAI MQSeries Transports

This chapter discusses the Siebel EAI MQSeries Transports and includes the following topics:

- “About Siebel EAI MQSeries Transports” on page 21
- “About Message Id Tracking for an Inbound Message” on page 30
- “Invoking a Workflow Process Using MQSeries Server Receiver” on page 30

About 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 the name EAI MQSeries Transport 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 the EAI MQSeries Server Transport, the named subsystem type is MqSeriesServerSubsys. For the EAI MQSeries AMI Transport, the named subsystem type is MqSeriesAMISubsys.

Following are examples of EAI MQSeries Server Transport and EAI MQSeries AMI Transport commands to create a named subsystem and start a receiver.

```
create named subsystem MyMqSrvrSubsys for subsystem MQSeriesServersubsys with  
MqPhysicalQueueName=Receiver,MqRespPhysicalQueueName=Sender,MqQueueManagerName=myQ  
ueueMgr
```

```
create named subsystem siebelEcho for subsystem EAITransportDataHandlingSubsys with  
DispatchService="Workflow Utilities",DispatchMethod=ECHO
```

```
start task for comp MqSeriesSrvRcvr with  
ReceiverConnectionSubsystem=MyMqSrvrSubsys,ReceiverDataHandlingSubsystem=SiebelEcho,  
ReceiverMethodName=ReceiveDispatchSend
```

```
create named subsystem MqAMIConnSubsys for subsystem MQSeriesAMISubsys with  
MqPolicyName=MY.POLICY,MqReceiverServiceName=MY.RECEIVER,MqTrace=true,MqTraceLocat  
ion=M:\siebel\log\mq
```

```
create named subsystem SiebelEcho for subsystem EAITransportDataHandlingSubsys with  
DispatchService="Workflow Utilities",DispatchMethod=ECHO
```

```
start task for comp mqseriesAMIRcvr with  
ReceiverConnectionSubsystem=MqAMIConnSubsys,ReceiverDataHandlingSubsystem=SiebelEch  
o,ReceiverMethodName=ReceiveDispatchSend
```

- For a discussion of named subsystems for Siebel eAI, see [Chapter 2, "EAI Transports and Interfaces Overview."](#)
- For more information on named subsystems, see *Siebel System Administration Guide*.

About the EAI MQSeries Server Transport

Before using the EAI MQSeries Server Transport, you 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 the IBM MQSeries server as well as the IBM MQSeries client. The IBM MQSeries server must be running on the same system with your Siebel Server.

The EAI MQSeries Server Transport supports the inbound and outbound methods described in the "Outbound Methods for a Transport Business Service" on page 12 and "Inbound Methods for a Transport Business Service" on page 13.

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.

Using the EAI MQSeries Server Transport on AIX

When you use the EAI MQSeries Server Transport on AIX, the shared memory segment required by the EAI MQSeries Server process can collide with the shared memory segment required by the queue manager. By default, the EAI MQSeries queue manager tries to use shared memory segment number 8. The EAI MQSeries Server Transport does not rely on any specific number and uses whatever segment is given to the process by the AIX operating system.

However, if you are using the default configuration, there is a possibility that the EAI MQSeries Server process gets segment number 8 from the operating system first, and as a result the queue manager cannot get its segment. In this case, the MQSeries Server Transport service fails with an error code of 2059 because it cannot connect to the queue manager.

To fix a shared memory segment conflict with the EAI MQSeries Server Transport 5.2 on AIX

- 1 Shut down any queue manager connected to the EAI MQSeries Transport.
- 2 Edit the file `/var/mqm/mqs.ini`. In the QueueManager section, for each queue manager of interest, add an additional line explicitly specifying the shared memory segment to use. For example:

```
QueueManager:
Name=myQueueManager
Prefix=/var/mqm
Directory=myQueueManager
IPCCBaseAddress=12
```

- 3 Restart each queue manager.

NOTE: This example shows shared number 12 used as the memory segment number. Possible legal values for the `IPCCBaseAddress` are 4, 5, 8, 9, 10, 11, and 12, although 8 has been found to be problematic. It is possible to run into this error even with the memory segment number set as 12 if the operating system has non-deterministically allocated segment 12 to the EAI MQSeries Server process ahead of the queue manager. If this is the case, a different segment number may need to be specified.

If the EAI MQSeries Server Transport business service on AIX continues to fail even after you have followed the previous procedures, you can configure the AIX environment to run Siebel Server with less memory using environment variable `LDR_CNTRL`. After you have finished, follow the procedures in the preceding section.

To configure the AIX environment to run Siebel Server with less memory

- 1 Shut down Siebel Server.
- 2 In the shell that you use to bring up Siebel Server, set the environment variable `LDR_CNTRL`. Using `csh`:

```
setenv LDR_CNTRL MAXDATA=0x30000000
```

NOTE: You can save the setting in the `siebenv.sh` or `siebenv.csh`.

- 3 Restart Siebel Server with this environment variable.

EAI MQSeries Server Transport Parameters

In addition to supporting the common transport parameters presented in [Table 3 on page 15](#), the EAI MQSeries Server Transport also uses the parameters shown in [Table 4](#). 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 4. EAI MQSeries Server Transport Specific Parameters

Argument	Display Name	Description
MqAcknowledgements	Receive Acknowledgements	Default is False. This parameter specifies whether or not delivery and arrival acknowledgements are to be received.
MqAckPhysicalQueueName	Acknowledgement Physical Queue Name	If the MqAcknowledgements is set to True, this parameter contains the name of the physical queue for acknowledgements to responses.
MqAckQueueManagerName	Acknowledgement Queue Manager Name	Defaults to MqQueueManagerName if unspecified. If MqAcknowledgements set to True, this parameter contains the name of the queue manager for acknowledgements to responses.
MqModelQueueName	Model Queue Name	Name of the MQSeries model queue.
MqPhysicalQueueName	Physical Queue Name	Name of the MQSeries physical queue. You can also create an alias queue which points to a target queue and use the alias queue name as the input argument physical queue name and send messages to the target queue. NOTE: Using an alias queue will work. However, since the alias queue does not have a backout queue defined, the receiver cannot roll back to the backout 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. The Response Queue Manager is the same as MqQueueManagerName.
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 4. 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	Default is 20000 milliseconds. The timeout interval on receive calls, in 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.

NOTE: The persistence of the message is the same as the persistence of the queue itself.

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 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`.

Exposing MQMD Headers as Properties

The EAI MQSeries Server Transport feature exposes all the MQMD headers as properties of a property set.

You can set any MQMD message header for the Siebel application by specifying it as property in a property set on the outbound side. Whereas on the inbound side, the MQMD message header of the response is exposed to the user as property on the output property set.

On the inbound side you can have all the MQMD message headers as part of the output property set without having to do extra steps to see these MQMD message headers.

On the outbound side, you can set the MQMD message headers using the EAI MQSeries Server Transport. In order to modify the MQMD message headers on the outbound side, the property value for `FullMQMDControl` needs to be set to `TRUE`.

During the sending business service step (EAI MQSeries Server Transport.Send) within the workflow process, input arguments are added that can modify MQMD headers. Once the property FullMQMDControl is set to TRUE, you can modify other MQMD headers as the examples reflect in Table 5:

Table 5. Input Arguments for Certain MQMD Headers

Property	Type	Values
MQMD_S_In_MsgType	Literal	TestMsgHeader
MQMD_S_In_CodedCharSetId	Literal	1208
MQMD_S_In_Encoding	Literal	MQENC_NATIVE
MQMD_S_In_Expiry	Literal	MQEI_UNLIMITED
MQMD_S_In_Encoding	Literal	MQENC_NATIVE
MQMD_S_In_Persistence	Literal	MQPER_PERSISTENT
MQMD_S_In_Priority	Literal	MQPRI_PRIORITY_AS_Q_DEF

NOTE: When using the Message Type header (MQMD_S_In_MsgType), make sure that the message type set makes sense in context. For example, if the Send method is used to send a message to MQSeries, the MsgType should not be set to MQMT_REQUEST. If the SendReceive method is used to send and request a response from MQSeries, then the MsgType of MQMT_REQUEST is applicable (this is automatically set by the Siebel application). In Table 5, MsgType is set to TestMsgHeader.

Table 6 summarizes the MQMD message headers that are exposed as properties in a property set.

Table 6. MQMD Message Headers

Field	Data Type	Description	Input/Output Property
StrucId	MQCHAR4	Structure Identifier.	Not exposed.
Version	MLONG	Structure version number.	Output.
Report	MLONG	Options for report messages.	Output.
MsgType	MLONG	Message type.	Input and output.
Expiry	MLONG	Message lifetime.	Input and output.
Feedback	MLONG	Feedback or reason code.	Output.
Encoding	MLONG	Numeric encoding of message data.	Input and output.
CodedCharSetId	MLONG	Character set identifier of message data.	Input and output.
Format	MQCHAR8	Format name of message data.	Input and output.
Priority	MLONG	Message priority.	Input and output.

Table 6. MQMD Message Headers

Field	Data Type	Description	Input/Output Property
Persistence	MQLONG	Message persistence.	Input and output.
MsgId	MQBYTE24	Message identifier.	Output.
CorrelId	MQBYTE24	Correlation identifier.	Output.
BackCount	MQLONG	Backout counter.	Output.
ReplyToQ	MQCHAR48	Name of reply queue.	Input and output.
ReplyToQMgr	MQCHAR48	Name of reply queue manager.	Output.
UserIdentifier	MQCHAR12	User identifier.	Output.
AccountingToken	MQBYTE32	Accounting token.	Output.
AppIdentityData	MQCHAR32	Application data relating to identity.	Output.
PutAppType	MQLONG	Type of application that put the message.	Output.
PutAppName	MQCHAR28	Name of application that put the message.	Output.
PutDate	MQCHAR8	Date when message was put.	Output.
PutTime	MQCHAR8	Time when message was put.	Output.
AppOriginData	MQCHAR4	Application data relating to origin.	Output.
GroupId	MQBYTE24	Group Identifier.	Output.
MsgSeqNumber	MQLONG	Sequence number of logical message within group.	Output.
Offset	MQLONG	Offset of data in physical message from start of logical message.	Output.
MsgFlags	MQLONG	Offset of data in physical message from start of logical message.	Output.
OriginalLength	MQLONG	Length of original message.	Output.

About the EAI MQSeries AMI Transport

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

- MQSeries Server
- MQSeries Client if necessary
- MQSeries Application Messaging Interface

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

CAUTION: IBM recommends not using the MQSeries AMI for new projects. For new projects use the WebSphere JMS in place of the MQSeries AMI. See the IBM documentation on using the WebSphere JMS.

For more information on how to perform these tasks, contact your IBM sales representative.

Configuring the EAI MQSeries AMI Transport

If you are using the MQSeries Application Messaging Interface (AMI), you 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 7](#).

Table 7. EAI MQSeries AMI Transport Specific Parameters

Argument	Display Name	Description
MqPolicyName	Policy Name	Defaults to the AMT.SYSTEM.SYNCPOINT.POLICY provided by IBM, if you do not specify an AMI Policy. This parameter is 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.
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 receives messages. The Receiver Service and the Sender Service may specify the same queue and queue manager.

Table 7. EAI MQSeries AMI Transport Specific Parameters

Argument	Display Name	Description
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 sends messages. The Receiver Service and the Sender Service may specify the same queue and queue manager.
MqTrace	Trace	Default is False. Turns on tracing for AMI.
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 make sure that there is a rule in your MQSeries configuration to check the request message to be put onto the MQSeries queue to contain values for the ReplyToQueue and ReplyToQueueManager parameters.

About the EAI MQSeries Transport Re-Entrance

The EAI MQSeries Server Receiver uses the EAI MQSeries Server Transport business service but cannot dispatch to a workflow process 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.

NOTE: In order to successfully roll back messages with the MQSeries AMI Transport, the Syncpoint option under the General tab and the Handle Poison Message option under the Receive tab must be checked.

About Message Id Tracking for an Inbound Message

You can keep track of Message Ids of inbound messages by creating a process property, `MsgId`, of type `String` and adding an output argument with the following configuration to the `Send` step of your process:

Type = Output Argument

Output Argument = MQSeries Message Identifier

This captures the Message Ids that the Queue Manager assigned to the messages in the `MsgId` process property.

Invoking a Workflow Process Using MQSeries Server Receiver

Following are examples of commands to create named subsystems and start a MQSeries Server Receiver to invoke a workflow process.

NOTE: If there is either an exception step or an error process in your workflow, the workflow assumes that the error step or the error process will handle the error and the workflow will not send the error out. In order to capture the error, you need to insert a stop step into your workflow. Note that by putting in a stop step, the caller gets the generic workflow stop error and not the original error, but the original error is stored in the `Error Code` and `Error Message` process properties.

Command to create EAI Transport Data Handling Subsystem

```
create named subsystem MYDataSubSys for subsystem EAITransportDataHandlingSubsys with DispatchWorkflowProcess="MQ Inbound workflow"
```

Command to create EAI Transport Connection Subsystem

```
create named subsystem MYSubSys for subsystem mqserieserversubsys with MQQueueManagerName=QueueMgr, MQPhysicalQueueName=LocalQueue
```

Command to start a MqSeriesSrvRcvr

```
start task for component MqSeriesSrvRcvr with ReceiverConnectionSubsystem=MYSubSys, ReceiverDataHandlingSubsystem=MYDataSubSys, ReceiverMethodName=Receivedispatch
```

When calling your workflow process by the MQSeries Server Receiver, it is not necessary to include a step to pull the messages off the queue and pass them to the next step. The MQSeries Server Receiver automatically pulls the messages off the queue and passes them on if:

- You have created a new process property of data type `String` and a default string of `<Value>`. This process property stores the inbound message text picked up by the `MqSeriesSrvRcvr`.

- In your workflow process step, where you handle the inbound messages from IBM MQSeries, you insert an input argument of <Value> with type Process Property. The Property Name will be the name of the process property you created in the previous step.

NOTE: When you type in <Value>, the display name may change to Message Text or XML Document.

4

EAI MSMQ Transport

This chapter discusses Siebel Systems' implementation of Microsoft MSMQ support with EAI MSMQ Transport and includes the following topics:

- ["About MSMQ" on page 33](#)
- ["Configuring the EAI MSMQ Transport Servers" on page 35](#)
- ["Configuring EAI MSMQ Transport for Various Send and Receive Scenarios" on page 36](#)

About MSMQ

Today, many large organizations are integrating various enterprise business applications into application networks. These networks allow applications to communicate with 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 EAI MSMQ Transport

EAI MSMQ Transport is a Siebel business service that can be customized using Siebel Tools. With Siebel Tools, you define integration objects to be transported across the EAI MSMQ Transport business service. EAI MSMQ Transport is responsible for sending and receiving messages between a Siebel application and MSMQ queues. 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

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

- Send
- Send and Receive Response
- 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 Business 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 a 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 EAI MSMQ Transport as the transport mechanism to inform these external systems.

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. In response, the sales agent then receives a list of transaction details for that customer displayed within a Siebel application form.

Messages to a Siebel Application from an External System

External applications can send messages to a Siebel application using 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.

EAI 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 2, "EAI Transports and Interfaces Overview."](#)
- For more information on named subsystems, see *Siebel System Administration Guide*.

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
 - Windows 2000 Advanced Server
- MSMQ Server
- As many MSMQ Queues as needed
- Relevant ODBC driver
- Siebel Server
- Siebel Gateway Name Server
- Siebel Web Client
- Siebel Tools

Regional Enterprise Server and MSMQ Client

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

- One of the following servers:
 - Microsoft Windows NT Server Enterprise Edition
 - Windows 2000 Server

- Windows 2000 Advanced Server
- The MSMQ Client
- As many MSMQ Queues as needed
- The relevant ODBC driver
- Siebel Server
- Siebel Gateway Name Server
- Siebel HTML Connected Client

NOTE: The 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 DB2, DB2 UDB for z/OS and OS/390, Oracle, and Microsoft SQL Server.

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.

EAI MSMQ Transport Prerequisites

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

EAI MSMQ Transport Parameters

Table 8 presents the parameters used for configuring EAI MSMQ Transport.

Table 8. 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.

Table 8. EAI MSMQ Transport Parameters

Parameter	Description
MsmqRespQueueName	Name of the response queue.
MsmqSleepTime	Default is 20000 milliseconds. The amount of time that the EAI MSMQ Transport business service waits to receive a message.
TimedOut	If no message is received in seconds specified in SleepTime, the TimedOut argument in the Output Property set will be set to True.
IgnoreCorrelationId	Default is False. 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. This parameter must be set to True to support BizTalk integration.
LargeMessageSupport	Default is True. Set to enable or disable Large Messages (messages over 4MB) Support. 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.

NOTE: For more detailed information on creating integration objects, see *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II*.

To define an integration object

- 1 Start Siebel Tools.
- 2 Lock the project you want in Tools, such as Contact Project.
- 3 Click New and choose Integration Object.
- 4 Select Contact as the project and EAI Siebel Wizard as the source system.
- 5 Select Contact as the 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 EAI MSMQ transport, either as part of a business process flow (using Siebel Workflow Process Manager), or as a custom business service.

Sending Outbound Messages with EAI MSMQ Transport

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 a Siebel application to MSMQ:

- Send
- Send and Receive Response (SendReceive)

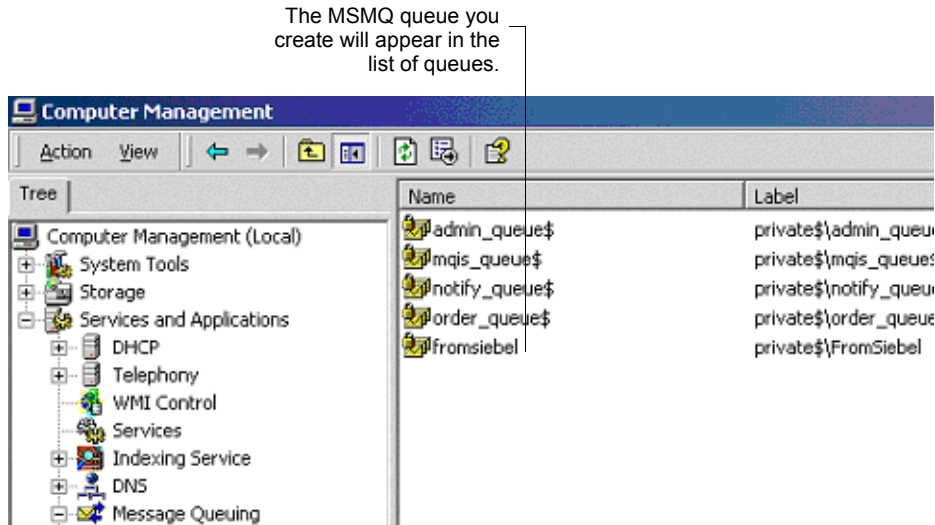
Sending Messages with EAI MSMQ Transport

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

To send messages from a Siebel application to MSMQ

- 1** Access the Windows Computer Management tool by choosing Start > Programs > Administrative Tools > Computer Management.

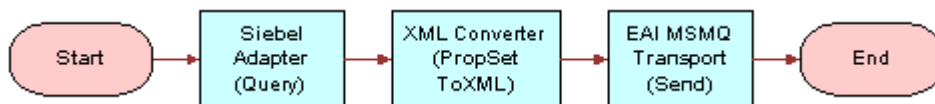
- Set up an MSMQ queue to receive messages from the Siebel application. Give the queue an easy-to-identify name, such as fromsiebel, as shown in the following illustration.



- 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 allows a commit or a rollback to be executed without failure.

- From the Site Map choose Administration - Business Process > Workflow Processes.
- Set up a workflow process for sending a message to MSMQ. Define the flow as shown in the following figure:



NOTE: For details on Business Process Designer, see *Siebel Business Process Designer Administration Guide*.

- Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Employee Message	Hierarchy	In/Out	-
Employee XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-

Name	Data Type	In/Out	Value
Object Id	String	In/Out	<i>Row Id of an Employee record</i>
Siebel Operation Object Id	String	In/Out	-

- 7** Set up the first step of the workflow, after Start, to use EAI Siebel Adapter with the Query method to query the information from the Siebel Database using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Output Integration Object	Literal	Sample Employee	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
Employee Message	Output Argument	-	Siebel Message

- 8** Set up the second step to use XML Converter with the PropSetToXML method to convert the data extracted from the Siebel Database to XML format using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Employee Message	Hierarchy

Property Name	Type	Value	Output Argument
Employee XML	Output Argument	-	XML Document

- 9** Set up the third step to use EAI MSMQ Transport with the Send method to send the information to the external system, using the following input arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	Employee XML	Binary

Input Arguments	Type	Value	Property Name	Property Data Type
MsmqPhysicalQueueName	Literal	private\$\FromSiebel	-	-
MsmqQueueMachineName	Literal	Siebel7	-	-

10 Save the workflow and run it from the Workflow Process Simulator.

Confirm that a message was sent to the queue using the MSMQ Explorer. In this example, a message should be in the fromSiebel queue and should contain an XML file with employee information.

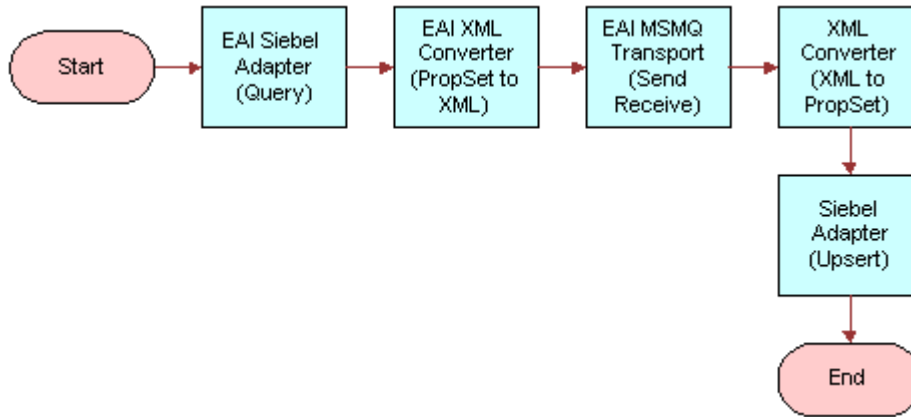
Sending and Receiving Messages with EAI MSMQ Transport

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

To send a literal to MSMQ and receive a response

- 1** Access the Windows Computer Management tool by choosing 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** From the Site Map choose Administration - Business Process > Workflow Processes.

- 5 Set up a workflow process for sending a message out and receiving a message in response using EAI MSMQ Transport. Define the flow as shown in the following figure:



NOTE: For details on Business Process Designer, see *Siebel Business Process Designer Administration Guide*.

- 6 Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Test Message	Hierarchy	In/Out	-
Test XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	-
Siebel Operation Object Id	String	In/Out	-

- 7 Set up the first step of the workflow after Start to use EAI Siebel Adapter with the Query method to query the information from the Siebel Database using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Output Integration Object	Literal	Sample Employee	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
Test Message	Output Argument	-	Siebel Message

- 8 Set up the second step to use XML Converter with the PropSetToXML method to convert the data extracted from the Siebel Database to XML format using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Test Message	Hierarchy

Property Name	Type	Value	Output Argument
Test XML	Output Argument	-	XML Document

- 9 Set up the third step of the workflow process, after Start, to use EAI MSMQ Transport with the SendReceive method to receive the incoming XML message, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
MsmqPhysicalQueueName	Literal	fromsiebel	-	-
MsmqQueueMachineName	Literal	Siebel2001	-	-
		Machine name where the Siebel MSMQ Transport is running.		
MsmqRespQueueMachineName	Literal	Siebel2001A	-	-
MsmqRespQueueName	Literal	tosiebel	-	-

Property Name	Type	Value	Output Argument
Test XML	Output Argument	-	XML Message

- 10 Set up the fourth step to use XML Converter with the XMLToPropSet method to convert the XML message to a Siebel property set using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	Test XML	Binary

Property Name	Type	Value	Output Argument
Test Message	Output Argument	-	Siebel Message

- 11 Set up the last step to use the EAI Siebel Adapter with the Insert or Update method to update the Siebel Database, using the following input argument:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Test Message	Hierarchy

12 Save the workflow and run a test using the Workflow Process Simulator.

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 EAI MSMQ Transport

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 EAI Siebel Adapter updates the appropriate tables within the Siebel Database.

NOTE: EAI 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 by choosing Start > Programs > Administrative Tools > Computer Management.
 - 2** Set up a 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 the queue.
- NOTE:** In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.
- 3** From the Site Map choose Administration - Business Process > Workflow Processes.

- 4 Set up a workflow for receiving a message from MSMQ as shown in the following figure:



NOTE: For details on Business Process Designer, see *Siebel Business Process Designer Administration Guide*.

- 5 Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Test Message	Hierarchy	In/Out	-
Test XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	-
Siebel Operation Object Id	String	In/Out	-

- 6 Set up the first step of the workflow process after Start to use the EAI MSMQ Transport with the Receive method. This step receives the incoming XML message, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
MsmqPhysicalQueueName	Literal	tosiebel	-	-
MsmqQueueMachineName	Literal	Siebel2001	-	-
		Machine name where the Siebel MSMQ Transport is running.		

Property Name	Type	Value	Output Argument
Test XML	Output Argument	-	Message Text

- 7 Set up the second step to use the XML Converter with the XMLToPropSet method to convert the XML message to a Siebel property set, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	Test XML	Binary

Property Name	Type	Value	Output Argument
Test Message	Output Argument	-	Siebel Message

- 8 Set up the third object to use the EAI Siebel Adapter with the Insert or Update method to update the Siebel Database, using the following input arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Test Message	Hierarchy

NOTE: In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.

- 9 Save the workflow process and run a test using the Workflow Process Simulator.

Confirm that the message is removed from the queue using the MSMQ Explorer. In this example, if the message on the fromSiebel is valid, the Siebel Database should be updated with the message in the fromSiebel queue.

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 EAI MSMQ Transport

- 1 Access the Windows Computer Management tool by choosing Start > Programs > Administrative Tools > Computer Management.
- 2 Set up a 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 the queue.

NOTE: In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.

- 3 From the Site Map choose Administration - Business Process > Workflow Processes.
- 4 Set up a workflow process for receiving and dispatching a message from MSMQ as shown in the following figure:



NOTE: For details on Business Process Designer, see *Siebel Business Process Designer Administration Guide*.

- 5 Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Test Message	Hierarchy	In/Out	-
Test XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	-
Siebel Operation Object Id	String	In/Out	-

- 6 Set up the first step of the workflow process after Start to use the EAI MSMQ Transport with the ReceiveDispatch method. This step receives the incoming XML message, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
MsmqPhysicalQueueName	Literal	tosiebel	-	-
MsmqQueueMachineName	Literal	Siebel2001	-	-
		Machine name where the Siebel MSMQ Transport is running.		
DispatchService	Literal	Workflow Utilities	-	-
DispatchMethod	Literal	Echo	-	-

Property Name	Type	Value	Output Argument
Test XML	Output Argument	-	Message Text

- 7 Set up the second step to use the XML Converter with the XMLToPropSet method to convert the XML message to a Siebel property set using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	Test XML	Binary

Property Name	Type	Value	Output Argument
Test Message	Output Argument	-	Siebel Message

- 8 Set up the third object to use the EAI Siebel Adapter with the Insert or Update method to update the Siebel Database, using the following input arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Test Message	Hierarchy

NOTE: In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.

- 9 Save the workflow process and run a test using the Workflow Process Simulator.

The contents of the output property set depend on the business service and method specified in the DispatchService and DispatchMethod arguments. Also, 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 EAI MSMQ Transport

- 1 Access the Windows Computer Management tool by choosing 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 the queue.

NOTE: In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.

- 4 From the Site Map choose Administration - Business Process > Workflow Processes.
- 5 Set up a workflow process for receiving and dispatching a message from MSMQ as shown in the following figure:



NOTE: For details on Business Process Designer, see *Siebel Business Process Designer Administration Guide*.

- 6 Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Test Message	Hierarchy	In/Out	-
Test XML	Binary	In/Out	Test Message from Siebel
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	-
Siebel Operation Object Id	String	In/Out	-

- 7 Set up the first step of the workflow process after Start to use EAI MSMQ Transport with the ReceiveDispatchSend method to receive the incoming XML message, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
MsmqPhysicalQueueName	Literal	tosiebel	-	-
MsmqQueueMachineName	Literal	Siebel2001	-	-
		Machine name where the Siebel MSMQ Transport is running.		
MsmqResponseMachineName	Literal	Siebel2001A	-	-
		Name of the machine where the queue receiving messages from Siebel application is located.		
MsmqRespQueueName	Literal	fromsiebel	-	-
DispatchService	Literal	Workflow Utilities	-	-
DispatchMethod	Literal	DispatchReceiveMethod	-	-

Property Name	Type	Value	Output Argument
Test XML	Output Argument	-	Message Text

NOTE: For illustration purposes, Workflow Utilities Echo method is used as dispatch service method. This could be changed to any dispatch service method as per your business requirements.

- 8 Set up the second step to use the XML Converter with the XMLToPropSet method to convert the XML message to a Siebel property set using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	Test XML	Binary

Property Name	Type	Value	Output Argument
Test Message	Output Argument	-	Siebel Message

- 9 Set up the third object to use the EAI Siebel Adapter with the Insert or Update method to update the Siebel Database, using the following input arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Test Message	Hierarchy

NOTE: In order to test this scenario adequately, you must have a partner application that can put a valid message for the Siebel application in the queue.

- 10 Save the workflow process and run a test using the Workflow Process Simulator.

The contents of the output property set depends on the business service and method specified in the DispatchService and DispatchMethod arguments. Confirm that the message is removed from the queue using the MSMQ Explorer. In this example, the Siebel Database should be updated with the message in the fromSiebel queue. Also, a response message will be in the queue specified by the MSMQRespQueueName and MSMQRespQueueMachineName arguments.

5

Java Business Service

This chapter includes the following topics:

- “About the EAI Java Business Service” on page 53
- “Prerequisites for Implementing a JBS” on page 53
- “Creating a Java Business Service” on page 55
- “About the Lifecycle of a Java Business Service” on page 57
- “Example of a Java Business Service” on page 57
- “About Restrictions for Implementing JBS” on page 58
- “Troubleshooting the Java Business Service” on page 58

About the EAI Java Business Service

The EAI Java Business Service is a service framework that allows custom business services to be implemented in Java and run from a Siebel application. The framework consists of the following:

- A template Business Service, EAI Java Business Service, defined in the repository.
- An abstract Java class, `com.siebel.eai.SiebelBusinessService`, that defines the interface of the Java class that implements the business service.

The EAI Java Business Service works by creating a Java Virtual Machine (JVM) in-process with the Siebel application and invoking Java implementations using JNI. Each Siebel process (component) has at most one JVM. JVMs are not shared across components.

Prerequisites for Implementing a JBS

In order to implement a Java business service, the following software must be installed and properly configured on each Siebel Server or Siebel Mobile and Dedicated Web Clients:

- A Java Runtime Environment.
- All necessary Java code.
- A configured named subsystem of type `JVMSubSys`.

The named subsystem supplies the parameters to the JBS. There are three parameters:

- **DLL.** The complete path of the Java Runtime Environment library. On Windows, it is `jvm.dll`; on AIX, `libjvm.a`; on HP-UX, `libjvm.sl`; and on Solaris, `libjvm.so`.

- **CLASSPATH.** The classpath used by the Java virtual machine.

The classpath must include the following Siebel JAR files as well as all Java code implementing the desired business service.

The required Siebel JAR files are:

- Siebel.jar
- SiebelJI_lang.jar (*lang* corresponds to the default language for your installation).
- **VMOPTIONS.** Java virtual machine options. On all platforms, except AIX, it is recommended that the option `-Xusealtsigs` be used to make sure that the signal handlers used by the Siebel Server do not conflict with those of the JVM.

The following example shows how to create a named subsystem using the Siebel Server Manager:

```
create named subsystem JAVA for subsystem JVMSubSys with
  DLL="D:\j2sdk1.4.2\jre\bin\server\jvm.dll",
  CLASSPATH="c:\cp\Siebel.jar;c:\cp\SiebelJI_enu.jar;c:\cp\myJARs.jar;.",
  VMOPTIONS = "-Xrs -Djava.compiler=NONE"
```

Alternatively, the parameters to the Java Business Service may be specified in the application configuration file instead of a named subsystem. This should only be used in conjunction with the Siebel Mobile and Dedicated Web clients, and not the Siebel Server.

```
[JAVA]
DLL = D:\j2sdk1.4.2\jre\bin\server\jvm.dll
CLASSPATH = c:\cp\Siebel.jar;c:\cp\SiebelJI_enu.jar;c:\cp\myJARs.jar;.
VMOPTIONS = -Xrs -Djava.compiler=NONE
```

Creating a Java Subsystem by Using the Siebel Web Client

The following is an alternative procedure for creating a Java subsystem by using the Siebel Web Client.

To create a Java subsystem by using the Siebel Web Client

- 1 Start any Siebel eBusiness application and navigate to Site Map > Administration - Server Configuration > Enterprises.
- 2 In the top list applet, select the Enterprise Server that you want to configure.
- 3 In the middle applet, click the Profile Configuration tab.
- 4 Click New to create a new component profile and set the following parameters:
 - a Profile = JAVA
 - b Alias = JAVA
 - c Subsystem Type = JVMSubsys
- 5 In the Profile Parameters list applet (the bottom applet), set the following values:
 - a Set the Value of the JVM Classpath parameter to one of the following:
 - The location of the JNDI.properties file (if using the JMS Transport).

- The JMS provider JAR files (if using the JMS Transport).
- The Siebel.jar and SiebelJI_enu.jar files. These files can be installed by using either Siebel Tools or the Siebel Server. An example of these files for Microsoft Windows follows:

```
c:\bea\weblogic.jar;c:\siebel\jndi;c:\siebel\siebsrvr\CLASSES\Siebel.jar;
c:\siebel\siebsrvr\classes\SiebelJI_enu.jar
```
- b** Set the Value of the JVM DLL Name parameter to the path where you have the jvm.dll file installed. For example, DLL= D:\j2sdk1.4.2\jre\bin\server\jvm.dll.
- c** Set the Value of the JVM Options record to any JVM-specific options that you would like to enable.

About Platform-Specific Configurations for the JVM

Depending on the platform, it is necessary to set certain environment variables to make sure of the proper loading of the JVM:

- **AIX.** Make sure that you have the environment variable LIBPATH set to include the JVM's .lib and classic directories. For example:

```
setenv LIBPATH /usr/j2sdk14/lib:/usr/j2sdk14/classic:${LIBPATH}
```

Also, make sure that the LD_LIBRARY_PATH is set correctly.

- **HP-UX.** Make sure that you have the environment variable SHLIB_PATH set to include the JVM's .jre and .bin directories and the server directory. For example:

```
setenv SHLIB_PATH
/opt/java1.4/jre/lib/PA_RISC2.0:/opt/java1.4/jre/lib/PA_RISC2.0/server:${SHLIB_PATH}
```

Set the variable LD_PRELOAD to the full path of the Java library.

- **Solaris, Windows.** No additional settings are needed.

When a Java business service is invoked on UNIX from a server component (for example, the JMS Receiver; see [Chapter 6, "Java Message Service Transport"](#) for more information), the necessary settings must be done in the script that creates the component.

For the receiver, the script is siebshw; for the application object managers, it is siebmtshw. These scripts are present in the .bin directory where the Siebel Server is installed.

Creating a Java Business Service

Implement a business service in Java by:

- Defining a new business service in the repository using Siebel Tools.
- Specifying the necessary Java classes.

To define and specify a new Java business service in Siebel Tools

- 1 Copy the EAI Java Business Service (using the Copy Record command in Siebel Tools).

NOTE: Checking the Cache column when you are creating the new Java business service will cause the same Java object to be reused by subsequent invocations within the same session. (See “About the Lifecycle of a Java Business Service” on page 57).
- 2 Add a Business Service User Property named @class, whose value is the fully qualified name of the Java class (for example, com.mycompany.siebelBusinessService.ImportCustomer).

Implementing a Business Service in Java

Once the Java business service has been defined in Siebel Tools, the Java class must be implemented. The Java class implementing the business service must extend com.siebel.eai.SiebelBusinessService.

SiebelBusinessService is an abstract Java class found in Siebel.jar. It declares three methods:

- **destroy.** This method is called when the Java object is released by the Siebel application. It has a default empty implementation and can be overridden for the purpose of performing any cleanup.
- **invokeMethod.** This method contains a default implementation that calls doInvokeMethod and catches any exceptions that are thrown by it. It does not declare any exceptions. It is invoked by means of JNI in the Siebel application’s native process. This method is not intended to be overridden.
- **doInvokeMethod.** This method must be implemented by the subclass that implements the business service. It takes as arguments the methodName, input property set, and output property set. The property sets are instances of com.siebel.data.SiebelPropertySet. This method throws SiebelBusinessServiceException.

About Exception Handling for the Java Business Service

Errors are handled by throwing a com.siebel.eai.SiebelBusinessServiceException class. The constructor for this class takes two String arguments, an error code and an error message. The error code may be used for programmatic handling in Siebel eScript when the business service is called. Both the error code and the error message will be displayed as an ordinary Siebel error message.

It is strongly recommended that proper error handling be employed when implementing the Java Business Service class. By invoking a SiebelBusinessServiceException, the standard Siebel error handling facilities will be employed.

If any other exception is received from doInvokeMethod, an error is produced with the details of the exception.

About the Lifecycle of a Java Business Service

A JVM is created in-process with the Siebel process the first time a Java business service is invoked. Thereafter, the same JVM is used for all invocations of any Java business services.

An instance of the Java class implementing a business service is created the first time that business service is invoked. This instance is released through JNI when the native business service is destroyed. For business services that are not cached, this occurs whenever the caller (workflow process, script) releases the native business service. For business services that are cached, this occurs when the session is destroyed (for example the user logs out). For a business service marked as cached in the repository, repeated invocations by a user during a single session will invoke methods on the same Java object.

Example of a Java Business Service

Following is an example of a Java class implementing a business service:

```
package com.mycompany.jbs;
import com.siebel.data.SiebelPropertySet;
import com.siebel.eai.SiebelBusinessServiceException;

public class AddBusinessService extends com.siebel.eai.SiebelBusinessService {

    public void doInvokeMethod(String methodName, SiebelPropertySet input,
        SiebelPropertySet output) throws SiebelBusinessServiceException {
        String X = input.getProperty("X");
        String Y = input.getProperty("Y");

        if (X == null || X.equals("") || (Y == null) || Y.equals(""))
            throw(new SiebelBusinessServiceException("NO_PAR", "Missing param"));

        if (!methodName.equals ("Add"))
            throw new SiebelBusinessServiceException("NO_SUCH_METHOD", "No such method");
        else {
            int x = 0;
            int y = 0;
            try {
                x = Integer.parseInt(X);
                y = Integer.parseInt(Y);
            }
            catch (NumberFormatException e) {
                throw new SiebelBusinessServiceException("NOT_INT", "Non-integer passed");
            }
            int z = x + y;
            output.setProperty("Z", new Integer(z).toString());
        }
    }
}
```

About Restrictions for Implementing JBS

When implementing a Java business service, keep in mind the following recommendations:

- Each business service method invocation should be atomic and stateless.
- The explicit creation of Threads is discouraged. It is not recommended that customers invoke a multi-threaded component from a Java business service.
- All data and context required to perform the necessary business functions must be provided as input to the class. The external Java class cannot call back into the Siebel application to obtain additional context.

Troubleshooting the Java Business Service

A common source of errors is the Java CLASSPATH.

Remember the following conventions of the Java CLASSPATH:

- On UNIX, CLASSPATH entries are separated by a colon (:); on Windows, by a semicolon (;).
- If .class files are to be used instead of .jar files, the root directory (for example, the one containing the com folder) must be listed in the CLASSPATH.

If the Java business service states that the `com.siebel.data.SiebelPropertySet` class is not found, then the `Siebel.jar` files are not correctly specified in the CLASSPATH.

If the Java business service implementation cannot be found, then the .class or .jar file containing its code is not properly specified in the CLASSPATH.

6

Java Message Service Transport

This chapter discusses the Siebel Java Message Service Transport and includes the following topics:

- "About the Siebel EAI JMS Transport" on page 59
- "About Synchronous and Asynchronous Invocation" on page 60
- "About the JMS Publish-and-Subscribe Model" on page 60
- "About Operations of the JMS Transport" on page 60
- "Features Not Supported for Use with the Siebel JMS Transport" on page 61
- "About JMS Message Types" on page 61
- "About Sending and Receiving XML" on page 62
- "About Multi-Step Operations Within a JMS Session" on page 62
- "Undeliverable Messages in JMS Transport" on page 63
- "Detailed Input and Output Specifications" on page 63
- "Configuring the JMS Transport" on page 67
- "Receiving, Dispatching, and Sending JMS Messages" on page 72
- "Troubleshooting for the JMS Transport" on page 75
- "About Logging for the JMS Transport" on page 77

About the Siebel EAI JMS Transport

The JMS Transport is an API for accessing enterprise messaging systems. It supports the ability to send and receive messages by way of JMS servers. JMS defines two messaging models: point-to-point (by way of JMS queues) and publish-and-subscribe (by way of JMS topics). Both are supported by Siebel's JMS Transport.

JMS queues and topics are identified by their JNDI names. A JNDI naming service is required in order to use the JMS Transport. It should contain entries for the desired queues and topics.

The API of JMS Transport is very similar to other Siebel messaging APIs such as the MQ Transport and MSMQ Transport.

The JMS Transport is built using the Java Business Service and therefore inherits all the requirements of that business service. This includes the independent installation of a Java virtual machine (JVM) and the configuration of the Siebel application to identify and create the VM.

About Synchronous and Asynchronous Invocation

Like the MQ Transport, the JMS Transport has two modes of execution—synchronous and asynchronous. Synchronous execution involves invoking individual methods of the JMS Transport directly, just like any other business service. Because the caller waits for the method to return, such invocation is synchronous. Asynchronous execution means listening for messages arriving on a particular queue and taking action whenever one arrives. This involves the creation of a separate Siebel component, called a JMS Receiver. Like the MQ Receiver, whenever a message arrives on the queue, the JMS Receiver *dispatches* to a business service (or workflow) and optionally sends a reply message.

About the JMS Publish-and-Subscribe Model

The traditional message model, where a message is sent to a queue and later removed by a single receiver, is called point-to-point messaging. In addition to this familiar model, JMS also supports the publish-and-subscribe messaging model. Here, messages are *published* to *topics*, rather than sent to queues. Interested receivers *subscribe* to individual topics and receive a copy of each message published to the topic. In order to subscribe, a subscriber registers with the topic, providing a unique identifier. For more information about the JMS publish/subscribe model, refer to the specification that is available from Sun at <http://java.sun.com/products/jms>.

About Operations of the JMS Transport

This section describes the operations (methods) of the JMS Transport.

JMS queues and topics are identified by their JNDI names. A JNDI naming service is required in order to use the JMS Transport. The JNDI naming service contains entries for the desired JMS Queues (implementers of `javax.jms.Queue`) and Topics (implementers of `javax.jms.Topic`) as well as the necessary JMS ConnectionFactories (implementers of either `javax.jms.QueueFactory` or `javax.jms.Topic`).

All methods that receive messages automatically time out if no message is available. The timeout length is three seconds by default and can be specified by the `ReceiveTimeout` argument. A value of zero for this argument disables the timeout, causing the method to wait indefinitely for a message to arrive. Whether a call to `Receive` or `Subscribe` timed out is provided as the `TimedOut` property of the output property set.

Subscriptions to JMS topics are always *durable* subscriptions.

The term *dispatch* is used to refer to the operation of calling a business service or workflow, passing as input, the content of a newly received message.

The arguments to these methods and their exact semantics (along with valid values, default values, and so on) are described in the section "[Detailed Input and Output Specifications](#)" on page 63. All methods require the JNDI name of JMS ConnectionFactory and the JNDI name of Queue or Topic.

The following is a summary of supported operations for use with the JMS Transport:

- **Receive.** Receive a message from a JMS queue.
- **ReceiveDispatch.** Receive a message from a JMS queue, then dispatch.
- **ReceiveDispatchSend.** Receive a message from a JMS queue, dispatch, and then send the result to a (possibly different) JMS queue.

The reply is sent to the SendQueue specified in the input to this method. (The JMSReplyTo header of the received message is ignored.)

- **Send.** Send a message to a JMS queue.
- **SendReceive.** Send a message to a JMS queue then receive a message from a (possibly different) JMS queue.

The JMSCorrelationID header of the reply message must be equal to the JMSCorrelationID of the message sent, unless it is null (if none was provided as an input to SendReceive), in which case it must be the JMSMessageID of the message sent.

- **Subscribe.** Receive a message from a JMS topic. The subscriber identifier must be supplied as an input to this method.
- **SubscribeDispatch.** Receive a message from a JMS topic, then dispatch. The subscriber identifier must be supplied as an input to this method.
- **Publish.** Publish a message to a JMS topic.

Features Not Supported for Use with the Siebel JMS Transport

JMS has a feature called Message Selection, by which a receiver or subscriber may filter the messages it receives by specifying certain criteria. This feature is not supported by Siebel's JMS Transport.

Also, it is not recommended that JMS messaging be used concurrently (for a single queue) with non-JMS messaging. For example, it is not recommended that a message be sent by way of JMS and later read using native tools. JMS vendors do not typically support such usage; it may result in the appearance of additional headers or additional obscure data in the body of the message.

About JMS Message Types

JMS defines five types of messages: TextMessage, BytesMessage, ObjectMessage, MapMessage, and StreamMessage. The Siebel JMS Transport supports only the types TextMessage and BytesMessage. If the JMS Transport receives an ObjectMessage, MapMessage, or StreamMessage from the JMS server, a *Unsupported Message Type* error is produced.

Like all Siebel business services, the output of any method is a property set. If a BytesMessage is received, then the value of the property set has Binary type. If a TextMessage is received, then the value has String type.

Conversely, the input to any method is also a property set. For methods that involve sending or publishing a message, the type of message sent or published depends on the type of the value of the input property set. If the type is Binary, then a `BytesMessage` is sent and published. If the type is String, then a `TextMessage` is sent and published.

NOTE: The Siebel Business Service Simulator in the Siebel Call Center always creates the input with a value type of String.

About Sending and Receiving XML

Messages whose content is XML are generally best treated as binary data and sent as `BytesMessages`. For example, the output of the Siebel business service EAI XML Converter is binary; therefore, if this is passed as the input to `Send`, then a `BytesMessage` will be sent.

If XML is sent as a `TextMessage`, the characters will be encoded as UTF-16. Therefore, the XML document should declare its encoding to be UTF-16.

Typically, when a message containing an XML document is received by the Siebel application, it is desirable to convert the document to a property set representation before processing it. This is accomplished automatically during the `Dispatch` step by specifying the `ConverterService` argument to be either XML Converter or EAI XML Converter. For more details about these converter services see the *XML Reference: Siebel eBusiness Application Integration Volume V* guide.

About Multi-Step Operations Within a JMS Session

All JMS operations are performed in the context of a transactional JMS `QueueSession`. If a send or receive operation throws an exception, the session is immediately rolled back. If the operation is successful, then the session is committed, unless the operation is part of a larger multi-step operation. In the case of multi-step operations, the transaction is handled as follows:

- **SendReceive.** If the send operation succeeds, then the JMS session is committed and a receive operation is performed. This is necessary because the receive operation may depend on a response to the first message.
- **ReceiveDispatch.** If the receive operation fails, then the JMS session is rolled back, and the dispatch operation is not attempted. If the receive operation succeeds, then the dispatch operation is attempted. If the dispatch succeeds, then the JMS session is committed; otherwise, both the Siebel transaction and the JMS session are rolled back.
- **SubscribeDispatch.** Same as `ReceiveDispatch`.
- **ReceiveDispatchSend.** If the receive operation fails, then the JMS session is rolled back, and further operations are not attempted. If the receive operation succeeds, then the dispatch operation is attempted. If the dispatch operation fails, then the JMS session and the Siebel transaction are rolled back; otherwise, the send operation is attempted. If the send operation fails, then the JMS session and the Siebel transaction are rolled back; otherwise, both are committed.

Each Dispatch operation is performed within a Siebel transaction.

NOTE: `ReceiveDispatch` and `ReceiveDispatchSend` should not be attempted from within an existing Siebel transaction, as nested transactions are not supported.

Also, as with all Siebel EAI receivers, if an operation fails during the execution of the JMS Receiver, the JMS Receiver component terminates. (A timeout is not a failure.)

Undeliverable Messages in JMS Transport

If a message is undeliverable, in the sense that repeated attempts by the Siebel JMS Transport to receive the message fail, then the message must be removed from the queue. Most JMS vendors provide some mechanism for dealing with such "poison messages." Weblogic, for example, can be configured to limit the number of times it will attempt to deliver a message before redirecting the message to an error queue or deleting the message altogether.

Detailed Input and Output Specifications

This section provides detailed information about the exact semantics of all input arguments and output values for each method of the JMS Transport.

JMS Headers and Properties

Every JMS message has a set of standard *headers*. Some of these headers can be specified as arguments to the methods of the JMS Transport that involve sending or publishing, and some are available as properties of the output property set of methods that involve receiving or subscribing. These are detailed in the tables below.

A JMS message may also be assigned *properties*. These may be user-defined properties specific to a particular application, or JMS-defined properties (for example `JMSXProducerTXID`) that are optionally supported by the JMS vendor. A property may be an instance of any Java class or any of the primitive Java types. All properties of a message received by the Siebel JMS Transport are available as properties of the output property set. The name of the property is the original name with the eleven characters `SIEBEL_JMS:` prepended; the value is the string obtained by converting the original value to a Java String. Conversely, when sending a message, any property of the input property set whose name begins with `SIEBEL_JMS:` is added to the message being sent as a property whose name has the prefix `SIEBEL_JMS:` removed.

About Input Arguments Used by the Dispatch Step

Table 9 reflects the optionality of each input argument except user-defined properties and arguments used by the Dispatch step. An asterisk (*) denotes a required argument, OPT denotes an optional argument, and a dash (-) denotes an argument that is ignored.

Table 9. Dispatch Step Input Arguments

Input	Send	Publish	Send Receive	Receive	Subscribe	Receive Dispatch	Receive Dispatch Send	Subscribe Dispatch
ConnectionFactory	*	*	*	*	*	*	*	*
ReceiveQueue	-	-	*	*	-	*	*	-
ReceiveTimeout	-	-	OPT	OPT	OPT	OPT	OPT	OPT
SendQueue	*	-	*	-	-	-	*1	-
Topic	-	*	-	-	*	-	-	*
SubscriberIdentifier	-	-	-	-	*	-	-	*
JMSPriority	OPT	OPT	OPT	-	-	-	OPT ²	-
JMSDeliveryMode	OPT	OPT	OPT	-	-	-	OPT ²	-
JMSExpiration	OPT	OPT	OPT	-	-	-	OPT ²	-
JMSReplyTo	OPT	OPT	- ⁴	-	-	-	OPT ²	-
JMSType	OPT	OPT	OPT	-	-	-	OPT ²	-
JMSCorrelationID	OPT	OPT	OPT	-	-	-	- ³	-

Some special notes regarding particular values in Table 9:

- OPT²: For ReceiveDispatchSend, the JMS headers are assigned to the reply message during the Send step
- *1: For ReceiveDispatchSend, the JMSReplyTo header of the received message is ignored. The reply is always sent to the given SendQueue. If JMSReplyTo is provided as an input to ReceiveDispatchSend, then it is assigned to the header of the reply sent during the Send step.
- -3: For ReceiveDispatchSend, the JMSCorrelationID of the reply message cannot be provided as an input. The JMSCorrelationID is automatically set to the JMSCorrelationID of the received message, unless empty, in which case it is set to its JMSMessageID.
- -4: For SendReceive, the JMSReplyTo header of the sent message is set to the value of the ReceiveQueue argument.

In lieu of providing the arguments individually, the single argument ConnectionSubsystem may be provided. Its value must be the name of a valid named subsystem of type JMSSubsys, and it must include all of the arguments that are required by the method to which it is passed. See “About the JMS Receiver” on page 68 for more information about that named subsystem.

JMS message properties are also supported as input arguments (properties) as described in “JMS Headers and Properties” on page 63.

Table 10 provides details for each input argument about the allowable values, default values, and special values, as well as the behavior if an invalid value is passed.

Table 10. Values for Input Arguments

Input	Default	Allowable Vaues	Special Values	If Value Invalid
ConnectionFactory	NONE	JNDI connection factory name		ERROR
ReceiveQueue	NONE	JNDI queue name		ERROR
ReceiveTimeout	3000	ANY INTEGER x >= 0	0: wait indefinitely	Non-int = 3000 < 0 = 0
SendQueue	NONE	JNDI queue name		ERROR
Topic	NONE	JNDI topic name		ERROR
SubscriberIdentifier	NONE	ANY STRING		N/A
JMSCorrelationID	NOT SET	ANY STRING		N/A
JMSPriority	javax.jms.Message.DEFAULT_PRIORITY (= 4)	ANY INTEGER 0 <= x <= 9	(0 lowest; 9 highest)	DEFAULT
JMSDeliveryMode	javax.jms.DeliveryMode.PERSISTENT	PERSISTENT, NON_PERSISTENT		DEFAULT
JMSExpiration	javax.jms.Message.DEFAULT_TIME_TO_LIVE (= 0)	ANY INTEGER x >= 0	0: Message never expires	DEFAULT
JMSReplyTo	NOT SET	JNDI queue name		ERROR
JMSType	SiebelJMSMessage	ANY STRING		N/A

About Dispatch Input Arguments

The methods that involve dispatching also take arguments that are needed for the Dispatch step. Those input arguments are the following:

- **DispatchService.** The business service to call.
- **DispatchMethod.** The business service method to call.
- **DispatchWorkflowProcess.** The workflow process to call.
- **DispatchRuleSet.** The rule set to apply.
- **ConverterService.** The converter service to apply (EAI XML Converter or XML Converter).

Dispatch may invoke either business service method, workflow process, or rule set. Only the applicable arguments need to be set. The Dispatch step is identical to the Dispatch step taken by other EAI Receiver components such as the MQ Series Receiver.

In lieu of providing the arguments individually, the single argument `DataHandlingSubsystem` may be provided. Its value must be the name of a valid named subsystem of type `EAITransportDataHandlingSubsys` and it must include all of the arguments that are required by the method to which it is passed.

About Output of the JMS Transport

The output of the JMS Transport methods includes the following parts:

- The content of the received message (if the method involves receiving a message). See the previous section, "About Dispatch Input Arguments" on page 65 for details about typing.
- JMS properties of the received message (if the method involves receiving a message), as described in the section "JMS Headers and Properties" on page 63.
- Certain JMS headers of the message sent or received, as described in Table 11.
- The special properties `TimedOut` (if the method involves receiving a message) and `DispatchError` (if the method involves dispatching), as described in Table 11. Each property is either `True` or `False`.

Table 11 enumerates for each method of JMS Transport the JMS headers and other distinguished properties that appear as properties of the output property set of the method. An asterisk (*) means the argument appears; a dash (-) means the argument is absent.

Table 11. Dispatch Step Input Arguments

Output	Send	Publish	Send Receive	Receive	Subscribe	Receive Dispatch	Receive Dispatch Send	Subscribe Dispatch
<code>TimedOut</code>	-	-	*	*	*	*	*	*
<code>*JMSType</code>	-	-	*	*	*	-	-	-
<code>*JMSCorrelationID</code>	-	-	*	*	*	-	-	-
<code>JMSRedelivered</code>	-	-	*	*	*	-	-	-
<code>JMSTimestamp</code>	-	-	*	*	*	-	-	-
<code>JMSMessageID</code>	*1	*1	*2	*2	*2	-	*1	-
<code>DispatchError</code>	-	-	-	-	-	*	*	*

Some special notes regarding the information in this table:

- *1: For `Send`, `Publish`, and `ReceiveDispatchSend`, the `JMSMessageID` is the value assigned by the JMS server to the message that was sent or published.
- *2: For `SendReceive`, `Receive`, and `Subscribe`, the `JMSMessageID` is the value of the header on the message received.

For the multi-step methods `ReceiveDispatch`, `ReceiveDispatchSend`, and `SubscribeDispatch`, properties are passed between the individual steps according to the following rules:

- All outputs of the Receive (or Subscribe) step are passed as inputs to the subsequent Dispatch step.
- The Dispatch step also has available to it all properties in the original input (assuming they are not overwritten by output properties with the same name from the Receive/Subscribe step).
- ReceiveDispatch and SubscribeDispatch return the output of the Dispatch step with the two additional properties TimedOut and DispatchError.
- If an error occurs during the Dispatch step of ReceiveDispatchSend, the output is that of the Dispatch step with the two additional properties TimedOut and DispatchError.

Configuring the JMS Transport

The JMS Transport is built using the Java Business Service and therefore inherits all the requirements of that business service. This includes the independent installation of a Java virtual machine (JVM) and the configuration of the Siebel application to identify and create the VM. Configuration of the Siebel application requires creating a named subsystem of type JVMSubSys with the necessary properties. Refer to the Java Business Service documentation for instructions on how to configure the JVM named subsystem.

JMS Transport requires the CLASSPATH property of the JVM subsystem must include the following packages or classes:

- Siebel.jar.
- SiebelJI_*lang*.jar (where *lang* corresponds to the default language for your installation).
- A directory containing the location of the jndi.properties file.

The jndi.properties file contains the necessary name value pairs required to perform a JNDI lookup and bind to the remote queue.

- Necessary classes and JAR files as required by the JMS provider.

To verify that the CLASSPATH and jndi.properties are properly configured, refer to [“Troubleshooting for the JMS Transport” on page 75](#).

About the JMSSubsys Named Subsystem

The arguments to any method of JMS Transport may be supplied individually as properties of the input property set or as part of a named subsystem of type JMSSubsys. When invoking the JMS Transport asynchronously by starting a JMS Receiver component, the arguments must be supplied by way of a named subsystem.

This subsystem supplies all of the necessary parameters for any one of the three methods ReceiveDispatch, ReceiveDispatchSend, or SubscribeDispatch. The parameters for the three methods are ConnectionFactory, ReceiveQueue, SendQueue, Topic, SubscriberIdentifier, ReceiveTimeout, JMSType, JMSPriority, JMSExpiration, JMSDeliveryMode.

In addition, this subsystem has a property `JVMSubsys`, which can be given the name of the JVM subsystem instance to use. The default value is `JAVA`. Therefore, if the property `JVMSubsys` is not explicitly given a value, there must be a properly configured instance of the type `JVMSubSys` named `JAVA`.

About the JMS Receiver

The JMS Receiver is a Siebel Server component that makes it possible for the JMS Transport to be invoked asynchronously. The JMS Receiver listens for messages arriving on a JMS Queue or Topic and takes action whenever a message arrives. The JMS Receiver repeatedly invokes a single method of the JMS Transport, either `ReceiveDispatch`, `ReceiveDispatchSend`, or `SubscribeDispatch`.

An instance of the JMS Receiver is configured with the parameters of a `JMSSubsys` named `subsystem`, which fix the queue or topic to listen to, as well as the action to be taken.

The following is an example of how an instance of the JMS Receiver could be configured and run by using the Siebel Server Manager command-line interface:

```
create named subsystem MyJMSSubsys_SR for subsystem JMSSubsys with
ConnectionFactory="weblogic.examples.jms.QueueConnectionFactory",
ReceiveQueue="weblogic.examples.jms.exampleQueueReceive",
SendQueue="weblogic.examples.jms.exampleQueueSend",
ReceiveTimeout=3000
```

```
create named subsystem SiebelEcho for subsystem EAITransportDataHandlingSubsys with
DispatchService="Workflow Utilities",
DispatchMethod="ECHO"
```

```
start task for comp JMSReceiver with
ReceiverConnectionSubsystem=MyJMSSubsys_SR,
ReceiverDataHandlingSubsystem=SiebelEcho,
ReceiverMethodName=ReceiveDispatchSend
```

For a discussion of named subsystems for the Siebel eAI, see [Chapter 2, "EAI Transports and Interfaces Overview."](#)

For more information on administering named subsystems, see the *Siebel System Administration Guide*.

Creating a JMS Subsystem by Using the Siebel Web Client

The following is an alternative procedure for creating a JMS Subsystem by using the Siebel Web Client and then configuring the JMS Transport.

To configure the JMS Transport by using the Siebel Web Client

- 1** Start any Siebel eBusiness Application and navigate to Site Map > Administration - Server Configuration > Enterprises.
- 2** In the top list applet, select the desired Enterprise Server that you want to configure.
- 3** In the middle applet, click the Profile Configuration tab.
- 4** Click New to create a new component profile and set the following parameters:
 - a** Profile = JMS_Q1ReceiveDispatchSend
 - b** Alias = JMS_Q1ReceiveDispatchSend
 - c** Subsystem Type = JMSSubsys
- 5** In the Profile Parameters list applet (the bottom applet), specify the parameters required for the type of operations the subsystem will need to support (for example, Receive or ReceiveDispatchSend).

For example, if this subsystem needed to support the ReceiveDispatchSend operation, at least the following values must be set:

- a** ConnectionFactory name = examples.jms.QueueConnectionFactory
- b** JVM Subsystem name = JAVA
- c** ReceiveQueue name = examples.jms.fromSiebel
- d** SendQueue name = examples.jms.toSiebel
- e** Receive Timeout = 1000

Sending and Receiving Messages with the JMS Transport

The procedure below describes how to set up the Siebel application to send a message to an external system using the EAI JMS Transport and receive a corresponding reply from the external system.

To send and receive messages with the JMS Transport

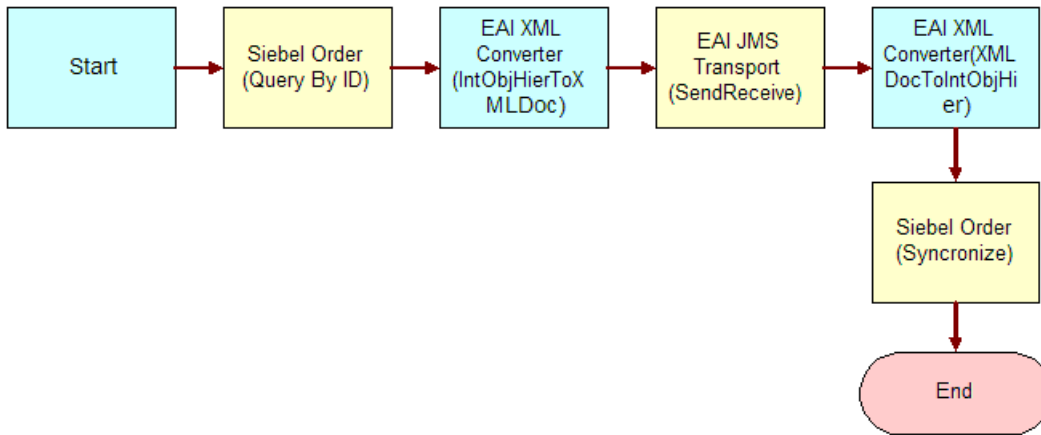
- 1** Set up a JMS queue to receive messages from the Siebel application and give the queue an easy-to-identify name, such as fromSiebel.
Refer to your JMS provider documentation on how to administer, monitor, and define new persistent queues.
- 2** Set up a JMS queue to send messages to the Siebel application.
Refer to your JMS provider documentation on how to administer, monitor and define new persistent queues.
 - a** Give the queue an easy-to-identify name, such as toSiebel.

b Create a message in the queue.

NOTE: In order to test this scenario adequately, you must have a partner application that can place a valid message for the Siebel application in the queue.

- 3** From the Site Map choose Administration - Business Process > Workflow Processes.
- 4** Configure and deploy a workflow process for sending a message out and receiving a message in response using the EAI JMS Transport.

Define the flow as shown in the following figure.



For details on the Business Process Designer, see the *Siebel Business Process Designer Administration Guide*.

- 5** Create the following process properties in the Process Properties applet:

Name	Data Type	In/Out	Default String	Comments
OrderXML	Binary	In		
JMSConnectionFactory	String	In	examples.jms.ConnectionFactory	JNDI name of the JMS connection factory
JMSReceiveQueue	String	In	examples.jms.toSiebel	JNDI name of the queue
JMSSendQueue	String	In	examples.jms.fromSiebel	JNDI name of the queue
JMSReceiveTimeout	String	In	180000	
Order Message	Integration Object	In		

- 6 Set up the first step of the workflow to use the Siebel Order ASI with the QueryById method to query the information from the Siebel database using the following input and output arguments:

Input Argument	Type	Value	Property Name
PrimaryRowId	Process Property		Object Id

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

- 7 Set up the second step of the workflow to use the EAI XML Converter with the IntObjHierToXMLDoc method to convert the data extracted from the Siebel database to XML using the following input and output arguments:

Input Argument	Type	Value	Property Name
generateProcessingInstructions	Literal	False	
SiebelMessage	Process Property		Order Message

Property Name	Type	Value	Output Argument
OrderXML	Output Argument		<value>

- 8 Set up the third step of the workflow process, after Start, to use the EAI JMS Transport with the SendReceive method using the following input and output arguments:

Input Argument	Type	Value	Property Name
<value>	Process Property		OrderXML
ConnectionFactory	Process Property		JMSConnectionFactory
ReceiveQueue	Process Property		JMSReceiveQueue
ReceiveTimeout	Process Property		JMSReceiveTimeout
SendQueue	Process Property		JMSSendQueue

Property Name	Type	Value	Output Argument
OrderXML	Output Argument		<value>

- Set up the fourth step to use the EAI XML Converter with the XMLDocToIntObjHier method to convert the XML message to an Integration Object using the following input and output arguments:

Input Argument	Type	Value	Property Name
<value>	Process Property		OrderXML

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

- Set up the last step to use the Siebel Order ASI with the Synchronize message to update the Siebel database using the following input and output arguments:

Input Argument	Type	Value	Property Name
SiebelMessage	Process Property		Order Message

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

- Save and deploy the workflow process.

It is recommended that the Workflow Process Simulator be used for testing purposes.

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, Dispatching, and Sending JMS Messages

The procedure below describes how to set up your system to receive inbound messages from JMS, perform an action within the Siebel application based upon the message, and send a synchronous response back to the external system.

To receive, dispatch, and send messages using EAI JMS Transport

- Set up a JMS queue to receive messages from the Siebel application and give the queue an easy to identify name, such as fromSiebel.

Refer to your JMS provider documentation on how to administer, monitor and define new persistent queues.

- 2 Set up a JMS queue to send messages to the Siebel application.

Refer to your JMS provider documentation on how to administer, monitor and define new persistent queues.

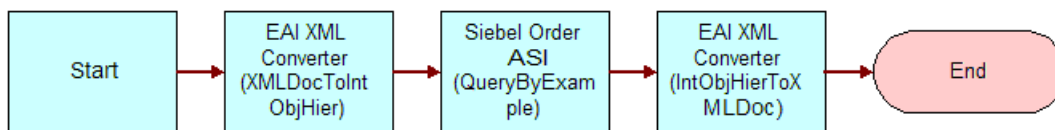
- a Give the queue an easy-to-identify name such as toSiebel.
- b Create a message in the queue.

NOTE: In order to test this scenario adequately, you must have a partner application that can place a valid message for the Siebel application in the queue.

- 3 From the Site Map choose Administration - Business Process > Workflow Processes.

- 4 Configure and deploy a workflow process to process the incoming XML request.

The following workflow process receives the incoming XML document and converts it to an integration object, executes a query using Siebel Order application service, and converts the response to an XML document as shown in the following figure:



- 5 Create the following process properties in the Process Properties applet:

Name	Data Type	In/Out	Default String	Comments
<value>	Binary	In/Out	<value>	Order Integration Object
Order Message	Hierarchy	In/Out		XML representation of the integration object

- 6 Set up the first step of the workflow process, after Start, to use the EAI XML Converter with the XMLDocToIntObjHier method.

This step will convert the incoming XML document to an integration object representation using the following input and output arguments:

Input Argument	Type	Value	Property Name
<value>	Process Property		<value>

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

- 7 Set up the second step to use the Siebel Order ASI with the QueryByExample method.

This step will query the Order business object based upon the provided XML document using the following input and output arguments:

Input Argument	Type	Value	Property Name
SiebelMessage	Process Property		Order Message

Property Name	Type	Value	Output Argument
Order Message	Output Argument		SiebelMessage

- 8 Set up the third step to use the EAI XML Converter with the IntObjHierToXMLDoc method.

This step will convert the integration object to a well-formed XML document using the following input and output arguments:

Input Argument	Type	Value	Property Name
SiebelMessage	Process Property		Order Message

Property Name	Type	Value	Output Argument
<value>	Output Argument		<value>

- 9 Save and deploy the workflow process.

Refer to the *Siebel Business Process Designer Administration Guide* for details on deploying workflow processes.

- 10 Define a JMS Connection subsystem using SrvrMgr (command line utility) or the Server Administration screen.

NOTE: The Siebel Server needs to be restarted in order for the new subsystem to be available.

Following is an example using SrvrMgr.

NOTE: ConnectionFactory, ReceiveQueue and SendQueue require JNDI names, which will vary depending upon the JMS provider and your implementation.

```
create named subsystem JMSToFromSiebel for subsystem JMSSubsys with
ConnectionFactory="jndiName.ConnectionFactory",
ReceiveQueue="jndiName.toSiebel ",
SendQueue="jndiName.fromSiebel",
ReceiveTimeout=3000
```

- 11 Define a data handling subsystem to dispatch the message from the toSiebel queue to the workflow process defined as previously defined (JMS Query Order).

NOTE: The Siebel Server needs to be restarted in order for the data handling subsystem to be available.

create named subsystem QueryOrder for subsystem EAITransportDataHandlingSubsys with DispatchWorkflowProcess="JMS Query Order"

- 12** After restarting the Siebel Server, start a new JMS Receiver from the SrvrMgr command line.

Following is an example that instructs the receiver to use the JMSToFromSiebel connection subsystem defined in [Step 10](#), the QueryOrder data handling subsystem defined in [Step 11](#), and instructs the receiver to use the ReceiveDispatchSend method of the EAI JMS Transport.

```
start task for comp JMSReceiver with
ReceiverConnectionSubsystem= JMSToFromSiebel,
ReceiverDataHandlingSubsystem=QueryOrder,
ReceiverMethodName=ReceiveDispatchSend
```

- 13** Place a message resembling the following on the toSiebel queue.

NOTE: A third-party product such as Hermes (available from Sourceforge.net) is required in order to place a message on a queue. In the following sample document, the Siebel Order ASI will query for all orders associated with the Hibblings Manufacturing account.

```
<?xml version="1.0" encoding="UTF-16"?>
<SiebelMessage IntObjectName="Order Interface">
  <ListOfOrderInterface>
    <Orders>
      <Account>Hibblings Manufacturing</Account>
    </Orders>
  </ListOfOrderInterface>
</SiebelMessage>
```

Troubleshooting for the JMS Transport

Several diagnostic methods are present in the EAI JMS Transport to assist in troubleshooting CLASSPATH, JNDI, and problems connecting to the JMS server.

- **CheckClasspath.** Iterates through the JVM's classpath, checking for the existence of each in the file system.
- **CheckJNDIContext.** Creates a JNDI InitialContext based on parameters (context factory class, URL) in resource file jndi.properties.
Lists the parameters and the entries found in the context, the names and classes of the administered objects.
- **CheckJNDIObjects.** Retrieves administered objects (connection factory, queue, topic) from JNDI.
- **CheckJMSServer.** Invokes JMS methods directly and simply.
If SendQueue is specified, it sends a message and then receives a message. If SendQueue is not specified and Topic is, it then creates a durable subscriber, publishes a message, receives it, and then unsubscribes.
- **CheckAll.** Executes all checks: CheckClasspath, CheckJNDIContext, CheckJNDIObjects, CheckJMSServer.

Table 12 contains more details of the arguments for use with the JMS Transport. The arguments listed are used by all three methods.

Table 12. Arguments to Use with the JMS Transport for Certain Methods

Method	Argument	Display Name	Type	Description
CheckJNDIObjects CheckJMSServer CheckAll	ConnectionFactory	Connection Factory	Input	The JNDI name for the JMSConnectionFactory.
	SendQueue	Send Queue	Input	The JNDI name for the queue (optional).
	Topic	Topic	Input	The JNDI name of the topic (optional).

About Logging for the JMS Transport

The JMS Transport will log messages to a file if the Java system property `jms.log` is set. This property is specified among the `VMOPTIONS` in the JVM subsystem. An example of such a log follows:

```
VMOPTIONS="-Djms.log=C:\temp\mylog.txt"
```


7

EAI HTTP Transport

This chapter discusses EAI HTTP Transport, its methods, and workflow examples illustrating using EAI HTTP Transport with different methods.

This chapter consists of the following topics:

- “About the EAI HTTP Transport” on page 79
- “Using POST and GET” on page 80
- “EAI HTTP Transport Named Subsystems” on page 81
- “General Information on How to Send a Message” on page 81
- “Using the EAI HTTP Transport for Inbound Integration” on page 84
- “EAI HTTP Transport for Inbound Messages” on page 90
- “Handling EAI HTTP Transport Business Service Errors” on page 93
- “Processing and Sending Outbound XML Documents” on page 93
- “Sending and Receiving Messages with the EAI HTTP Transport” on page 95
- “Examples Using HTTP Request” on page 98
- “About Transport Headers and HTTP Response Headers” on page 100
- “EAI HTTP Transport Method Arguments” on page 101

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 into 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 *CSSHHTTPTransService* class. You can use one of the following two methods with this transport:

- **Send.** This method supports outbound messages (XML documents sent from a Siebel application to an external system). The Send method means that the response coming back from the external application is not interpreted by the Siebel application, but the Web server should be sending back a correct HTTP response.
- **SendReceive.** This method supports outbound messages (XML documents sent to a Siebel application from an external system). This method is called *Send and Receive a Response* and the HTTP response body is the response for the request.

Each method has its own arguments, techniques, and applications. The EAI HTTP Transport 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 for Using the EAI HTTP Transport

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 that is 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 XMLDocToIntObjHier method to pass an integration object to EAI 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 13](#) identifies restrictions on these HTTP methods.

Table 13. Restrictions on GET and POST Methods with EAI 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.

EAI 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 is 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 2, "EAI Transports and Interfaces Overview."](#) For more information on named subsystems, see *Siebel System Administration Guide*.

General Information on How to Send a Message

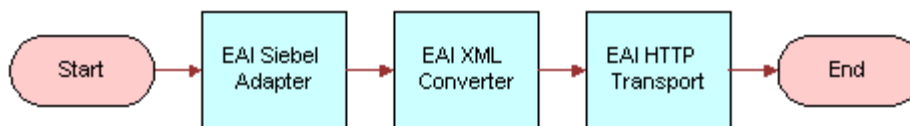
The following steps demonstrate how to send information from a Siebel application to another Web-based application using the EAI HTTP Transport.

- 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.

NOTE: For details about integration objects, see *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II*.

- 3 Create a workflow process, using Business Process Administration, to send this information to an external system as shown in the following figure:



- a Create the following process properties in the Process Property applet:

Name	Data Type	In/Out	Value
Account Message	Integration Object	In/Out	-
Account XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	<i>Row Id of an account</i>
Siebel Operation Object Id	String	In/Out	-

- b Set up the first step of the workflow after Start to use the EAI Siebel Adapter with the Query method to query the information from the Siebel Database, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Output Integration Object	Literal	Sample Account	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
Account Message	Output Argument	-	Siebel Message

- c Set up the second step to use the EAI XML Converter with the IntObjHierToXMLDoc method to convert the data extracted from the Siebel Database to XML format, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Account Message	Integration Object

Property Name	Type	Value	Output Argument
Account XML	Output Argument	-	XML Document

- d Set up the third step to use the EAI HTTP Transport with the Send method to send the information to the external system, using the following input arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	Account XML	String
Request Method	Literal	POST		
Request URL Template	Literal	http:// \$web_address\$/ \$request_param\$	-	-

Property Name	Type	Value	Output Argument	Property Data Type
Account XML	Output Argument	-	Message Text	-

- e Save the workflow and run it from the Workflow Process Simulator.
- 4 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, Siebel Gateway Name Server, and Siebel Server be installed, configured, and up and running.

NOTE: Type `http://web_server_name` URL on any machine that already has connectivity to the Web server to check the connectivity between the URL (for EAI HTTP Transport) issuing machine and SWE. This URL brings up the Home page of the Web server confirming the connectivity between SWE 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 installation directory. This file is on the Web server side of your configuration, as opposed to on the Siebel Server side of your installation. You should review the configuration file to make sure that the parameters are set properly. Use named subsystems to dispatch to a workflow as described in the section “Using Named Subsystems for Transport Parameters” on page 13 in Chapter 2, “EAI Transports and Interfaces Overview.”

To configure the SWE 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_1ang]`. Where `1ang` is the three-letter language code for the language you are using, such as `enu` for U.S. English.
- 3 Add the `EnableExtServiceOnly` configuration parameter or set it as follows, if it already exists, to enable the HTTP inbound transport. This example shown is for use with UNIX in an English environment.

```
[/eai_enu]
ConnectionString = Connect String
EnableExtServiceOnly = TRUE
```

For the virtual directory, you need to set the `ConnectionString` parameter. The syntax for the `ConnectionString` is:

```
ConnectString = siebel[.transport][.encryption][.compression]://SiebelServer  
:SCBPort/EnterpriseName/XXXObjMgr_lang
```

Where:

transport = TCPIP or http.

encryption = none or mscrypto.

compression = none or zlib.

SiebelServer = the name of your Siebel Server.

SCBPort = Listening port number for the SCBroker component (default is 2321).

EnterpriseName = the name of your Siebel Enterprise Server.

XXXObjMgr_lang = the type of Object Manager for the Siebel eBusiness application you are installing and the language code used for this installation.

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 the Siebel eBusiness Application you are installing.

In the example connect string, the Siebel eBusiness Application installed is Siebel eSales, and the Siebel Object Manager is called *EAIObjMgr*.

```
ConnectString = siebel.TCPIP.None.None://server1:2321/siebel77/EAIObjMgr_enu
```

4 Save and close the configuration file.

Setting Configuration Parameters for Siebel Server

You must also set certain configuration parameters for whatever Siebel 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

The EAI HTTP Transport can be used in two modes:

- Session mode
- Sessionless mode

The following sections explain the use of these two modes.

Viewing the Session Mode Between HTTP Requests

This mode uses HTTP Session Cookie to retain the session information between the HTTP requests. The session mode can be viewed when a sequence of calls is supported from an HTTP application into the EAI HTTP Transport.

To view the session mode from an HTTP application into an 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](#).

NOTE: For session mode inbound HTTP requests, the expiration date of the cookie sent to the client application will not be set as it is expected that this cookie will be used to send multiple requests within the same session.

Login Examples for the HTTP Protocol

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 14](#) presents each of the Login HTTP Request variables for session mode.

Table 14. Session Mode Variables

Variable	Description
webserver	URL of the Web server that has Siebel Web Engine installed, such as <code>www.myserver.com</code> .
path	Virtual path on the server referring to specific SWE configuration. This value should be <code>eai</code> .

Table 14. Session Mode Variables

Variable	Description
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&UserName=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
```

where *data text* is the business service input data. Most of the time, this is the text of an XML document that on the server side is 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>
```

To use this URL, you 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 Unicode-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"`.

NOTE: For XML messages being received by way of the Inbound HTTP Transport, only a Unicode (UTF-8 or UTF-16) format (with accordant *encoding* XML-processing header attribute and encoded XML data) is allowed. No ISO or Windows code pages are accepted.

Logoff HTTP Request

This request must include the session cookie from Login.

Using HTTP GET

URL = `http://webserver/path/start.swe?SWEEExtCmd=Logoff`

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

Example Logoff URL

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

EAI HTTP Transport in 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 because 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 of URLs Used for the HTTP Inbound Transport

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 15 presents each of the variables for sessionless mode.

Table 15. Sessionless Mode Variables

Variable	Description
webserver	URL of the Web server that has Siebel Web Engine installed, such as <i>www.myserver.com</i> .
path	Default is <i>eai</i> . Virtual path on the server referring to specific SWE configuration.
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 is converted to a PropertySet and passed to the business service. For more information on how to pass Properties and PropertySet to Business Services, see the <i>Siebel Business Process Designer Administration Guide</i> .

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?SWEExtSource=SiebelQuery&
SWEExtCmd=Execute&UserName=user1&Password=login123
&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>
```

To use this URL you will:

- Change the WebServer address, *www.myserver.com*, to your actual Web server URL.
- Verify that the SWEExtSource 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, such as "SADMIN/SADMIN."

EAI HTTP Transport for Inbound Messages

To use the EAI Transport, you complete two steps:

- Set up the business service for use in the workflow.
- Create the workflow.

Both steps are explained in this section.

This scenario assumes incoming XML. Your business requirements dictate if and how you need to adapt these steps to fit your needs.

To set up the business service

- 1 Start Siebel Tools, connecting to the server.
- 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 ProcessName in the Name column.
- b Enter EAITEST in the Value column, as shown in the following illustration.

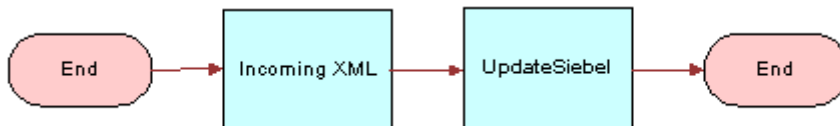
W	Name	Changed	Project	Cache	Class
>	EAITEST	✓	Account	✓	CSSWFEngine
	WI Web Proxy Service		WI - Web Integration		CSSWIService
	Web Collab Service		Web Collaboration	✓	CSSWebCollabService
	Web Engine HTTP TXN		SWE		CSSServiceSweHttpTxn
	Web Engine Interface		SWE	✓	CSSServiceSWEInterface

W	Name	Changed	Value	Inactive	Comments
>	Process Name	✓	EAITEST		

- 5 Compile a new .srf file and copy it to the *SIEBSRVR_ROOT*\Object directory.
- 6 Restart the Siebel Server.
- 7 Verify that the EAI Object Manager has started.

To create the new workflow process to receive messages

- 1 Log in to the Siebel client as an administrator connected to the server.
- 2 From the Site Map choose Administration - Business Process > Workflow Processes.
- 3 Create a new workflow process as shown below and give it a unique name, such as EAITEST.



- 4 Select the Process Properties tab and add the following properties:

Name	Data Type	Default String	In/Out	Description
IncomingXML	Binary	<Value>	In/Out	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	Hierarchy	-	In/Out	This is hierarchy format of the incoming XML.

Name	Data Type	Default String	In/Out	Description
<Value>	Binary	-	In/Out	Used to get the XML string that has been read or converted.
Content-Type	text/html	-	Out	It indicates the content type of the response body. If you want to see the response in the same Web page then you need to set this parameter to text/html.

- Set up the first step of the workflow after Start to use the EAI XML Converter with the XML Document to Integration Object Hierarchy method. This step converts the message, using the following input and output arguments.

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	IncomingXML	Hierarchy

Property Name	Type	Value	Output Argument
Account message	Output Argument	-	Siebel Message

- Set up the second step to use the EAI Siebel Adapter with the Insert or the Update method and the following input and output argument to update the Siebel Database.

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Account Message	Hierarchy

Property Name	Type	Value	Output Argument
<Value>	Literal	<h1>Update Completed</h1>	-

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.

- Save your workflow process and test it using the Workflow Process Simulator.

Handling EAI HTTP Transport Business Service Errors

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 also returns 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.

Processing and Sending Outbound XML Documents

This section explains how to use Siebel Tools and the Siebel application to set up the EAI 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.

Controlling the Behavior of EAI HTTP Transports

You can specify the parameters that control the behavior of transports in the following order:

- [Specifying Parameters as Business Service User Properties on page 93.](#)
- [Specifying Parameters as Subsystem Parameters on page 94.](#)
- [About Parameters as Run-Time Properties on page 95.](#)
- [About Parameters in Parameter Templates on page 95.](#)

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 overrides any values you have defined in Siebel Tools and compiled into the .srf file.

Specifying Parameters as Subsystem Parameters

You can specify parameters 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 Siebel Dedicated Web Client

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

```
    HTTPRequestURLTemplate="http://www.anyURL.com"
    HTTPRequestMethod="GET"
```
- 4 Save the file and exit the text editor.
- 5 In the workflow process, specify HTTPSubSys in the Connection Subsystem parameter value.

To specify the parameters on the Siebel Web Client

- 1 Start any Siebel eBusiness Application and navigate to Site Map > Administration - Server Configuration > Enterprises.
- 2 In the top list applet, select the Enterprise Server that you want to configure.
- 3 In the middle applet, click the Profile Configuration tab.
- 4 Click New to create a new component profile and set the following parameters:
 - a Profile = HTTP_test
 - b Alias = HTTP_test
 - c Subsystem Type = HTTPSubSys
- 5 In the Profile Parameters list applet (the bottom applet), specify the parameters required for the type of operations the subsystem will need to support:
 - a HTTPRequestURLTemplate = "http://www.anyURL.com"
 - b HTTPRequestMethod = "GET"

Then, in the workflow process on the Siebel Web Client, you will specify the Connection Subsystem input argument to the HTTP Transport, and the value will be the named subsystem that you created. For the case above, it will be HTTP_test. You can run the workflow process in Simulator using the Siebel Web Client.

About 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 EAI 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 EAI HTTP Transport business service directly.

NOTE: Subsystem parameters take precedence over run-time parameters.

About 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=$PWD$
```

where

```
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.

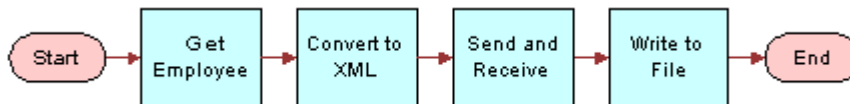
The token must be defined as either a business service user property or as a run-time parameter in the input property set. For example, you could specify the HTTPLoginURLTemplate as a user property of the business service, and *username* and *password* as run-time properties. Any logins that specify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

Sending and Receiving Messages with the EAI HTTP Transport

You can use the EAI HTTP Transport to send and receive messages. The following procedure illustrates how you can use EAI HTTP Transport with the SendReceive method to query employee information from the Siebel Database, send it out, echo it using the Workflow Utilities ECHO service, and send it back to the workflow to write the response back to a file.

To create a workflow process to send and receive messages

- 1 Create a named subsystem HTTPsendreceive_conn for subsystem HTTPSubSys with HTTPLoginMethod=GET
 HTTPLoginURLTemplate="http://smthpa12.siebel.com:16007/eai_enu/start.swe?SWEExtCmd=ExecuteLogin&SWEExtSource=MyEcho&UserName=SADMIN&Password=db2"
 HTTPLogoffMethod=GET
 HTTPLogoffURLTemplate="http://smthpa12.siebel.com:16007/eai_enu/start.swe?SWEExtCmd=Logoff"
 HTTPRequestMethod=POST
 HTTPRequestURLTemplate="http://smthpa12.siebel.com:16007/eai_enu/start.swe"
- 2 Create a named subsystem MyEchoSubsys for subsystem EAITransportDataHandlingSubsys with DispatchService="Workflow Utilities"
 DispatchMethod=ECHO
NOTE: For details on named subsystems, see "Configuring Named Subsystems and Receiver Server Components" on page 17.
- 3 In your eai.cfg file, add the following line in the [HTTP Services] section:
 MyEcho = MyEchoSubsys
- 4 Log in to the Siebel client as an administrator connected to the server.
- 5 From the Site Map choose Administration - Business Process > Workflow Processes.
- 6 Create a new workflow process as shown below.



- 7 Select the Process Properties tab and add the following properties.

Name	Data Type	In/Out	Default String
Employee Message	Hierarchy	In/Out	-
Employee XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	1-548
Response	Binary	In/Out	-

- 8 Retrieve employee message using the EAI Siebel Adapter with the Query method to query the information from the database using the following input and output arguments.

Input Arguments	Type	Value	Property Name	Property Data Type
Output Integration Object Name	Literal	Sample Employee	-	-
Object Id	Process Property	-	Object Id	Sting

Property Name	Type	Value	Output Argument
Employee Message	Output Argument	-	Siebel Message

- 9 Convert the message to XML using the EAI XML Converter with the Integration Object Hierarchy to XML Document method and the following input and output arguments to convert the message.

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Employee Message	Hierarchy

Property Name	Type	Value	Output Argument
Employee XML	Output Argument	-	XML Document

- 10 Send and receive the converted XML message using the EAI HTTP Transport with the Send and Receive Response method and the following input and output arguments.

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	Employee XML	String
Connection Subsystem	Literal	HTTPsendreceive_conn	-	-

Property Name	Type	Value	Output Argument
Response	Output Argument	-	Message Text

- 11 Write the message to the file using the EAI File Transport with the Send method and the following input arguments.

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	Response	Binary
File Name	Literal	C:\SendRec.txt	-	-

- 12 Save your workflow process and test it using the Workflow Process Simulator.

Examples Using HTTP Request

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

Controlling Login Sessions with 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=$Source&SWEExtCmd=ExecuteLogin&UserName=$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 = "siebel1/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 `SWEExtData` can be put into the body. This would change the sample code so that the `HTTPRequestURLTemplate` would read as:

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

Sending Requests 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.

```
HTTPRequestMethod = "POST"
```

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

```
HTTPRequestBodyTemplate = "Acct=ABCInt1&User=$Username$&pwd=$Password$"
```

```
Username = "acctuser"
```

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

Accessing a URL Protected by Basic Authentication

You can use basic authentication with the EAI HTTP Transport to send messages. The format to use in the URL to be able to access a URL protected by basic authentication with HTTP Outbound is:

```
http://username:password@host/rest of the URL
```

For example, `"http://Administrator:manage@169.254.85.21:5555/."`

NOTE: For outbound HTTP requests only server-side authentication is supported.

Creating Custom Headers for the EAI HTTP Transport Service

Custom headers can be created when sending a request through the EAI HTTP Transport service using a script or a workflow process.

To create custom headers for the EAI HTTP Transport service

- Create a new input property in the input to the HTTP transport.

The name of the property needs to have a prefix of `HDR`, followed by the desired name of the http header. For example:

```
HDR.CustomHTTPHeader  
  
httpInPS.SetProperty("HDR.CustomHTTPHeader", "MyValue");  
httpSvc.InvokeMethod("SendReceive", httpIn, httpOut);
```

A custom HTTP header with a name of "CustomHTTPHeader", and a value of "MyValue" is the result.

About Sending and Receiving Messages through HTTP

To send and receive messages through HTTP, you need to set up a workflow with the SendReceive method.

The Receive part of that method receives the response in an output argument of that method. You can then use the response to perform an upsert using an integration object and EAI Siebel Adapter, or display the response to your user. In this scenario, none of your quote integration design uses the eai.cfg or the SWE. You are performing an outbound HTTP call and waiting for a response synchronously.

You can then communicate the response to the user by displaying the returned error message in a browser alert or use the new User Interact step of the Workflow Process to refresh the view and show any new updates to fields to the user. The User Interact step can run synchronously or asynchronously, in the local Object Manager or on the server.

About Transport Headers and HTTP Response Headers

This section describes how transport headers and HTTP response headers work with HTTP Transport (outbound) to form a cookie handling system. HTTP Transport handles the cookie it receives from the server by storing and then creating a valid request transport header that it sends back to the server as a part of the request.

By exposing all the HTTP response headers as a part of output property set, you will be able to handle the response accordingly. You can have all the HTTP response headers, as well as HTTP Status code, as part of the output property set.

Transport headers are preserved across various connections and are a part of the transport service and not the HTTP connection.

Transport headers have the following features:

- Every connection has its own transport header.
- The transport header will separately store each cookie sent by the server during a connection.

For example, each name, domain, value pair, along with path, and other attributes (if present) are stored as a separate cookie in the transport header.

- Each cookie in the transport header has a distinct name.
Two cookies with the same name cannot be present in the transport header at the same time. The second cookie will overwrite the first one. Therefore, since the transport header is implemented as a CSSMapStringToPtr class, each cookie is hashed in the transport header based on its name.
- The transport header classifies cookies into two categories:
 - Type HTTP Version 1 and above.
 - Preliminary Netscape cookie spec type.
- When a ToString function is called on the transport header, it scans through the header and collects all the cookies in the header and creates a request transport header (based on the cookie category).
- The transport header is cleared when the connection is terminated.
- During SendReceive, the HTTP response has HTTP headers associated with it. Expose those response HTTP headers as properties of the output property set.
All these HTTP header properties are distinguished from other properties by appending the prefix *HDR.* in front of the property (header) name.
- Also, HTTP Status code for the HTTP request sent by way of EAI HTTP Transport is exposed as a property in the output property set. The property is called *StatusCode*.

EAI HTTP Transport Method Arguments

EAI HTTP Transport methods take the arguments presented in [Table 16](#).

- 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 16. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
<value>	<i>User-Defined Message Text</i>			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.

Table 16. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPRequestMethod	Request Method	■	■	HTTP method to use with the data request, such as Post and Get.
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.
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. Generally, the login parameters in a login query are specified in the body of the request that uses the POST method. This is 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.

Table 16. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPLogoffMethod	Log Off Method	■		Defaults is HTTPLoginMethod. HTTP method to be used for logging off.
HTTPAccept	HTTP Accept			Default is <i>text/*</i> . The explicit value for the Accept: header to override the default. Specifies the MIME types accepted by the sender.
HTTPContentType	HTTP Content Type			Default is <i>application/xxx-form-urlencoded</i> . The explicit value for the Content-Type: header to override the default. Specifies the type of data sent in the body of the request.
HTTPUserAgent	HTTP User Agent			Default is <i>Mozilla/4.0</i> . The explicit value for the User-Agent: header to override the default. Specifies the name/version of the client program.
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.
HTTPAllowCaching	Allow Caching			Default is N. By default, the responses for specific URL addresses are not cached by the EAI HTTP Transport. Set this flag to Y to enable caching. Note that this can lead to undesirable side effects, as old data from earlier requests can be exposed from the cache buffer.

Table 16. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPAllowPersistentCookies	Allow Persistent Cookies			<p>Default is N. 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 in the control of the EAI 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>Default is N. If set to N then 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 <i>HTTPS:</i> designation in an URL, by default enables the Secure flag and specifies <i>clear text</i> unencrypted communications.</p>
HTTPNoAutoRedirect	No Auto Redirect			<p>Default is N. This means auto-redirect is enabled. Setting this parameter to Y disables auto-redirection of messages to other URLs.</p>

Table 16. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPSleepTime	Sleep Time			Default is 120000 milliseconds. The timeout interval on login, send, and logoff requests in milliseconds.
HTTPImplicitCharsetDetection	Implicit Character Set Detection			Default is False. This is implicit character set detection for incoming data and should not be set to True for self-describing documents like XML. If set to True, this overrides the CharSetConversion parameter.

8

EAI DLL and EAI File Transports

This chapter discusses the EAI DLL and EAI File Transports and includes the following topics:

- “About EAI DLL Transport” on page 107
- “About the EAI File Transport” on page 109

About EAI DLL Transport

You use the EAI DLL Transport when you want to call a function that exists in an external DLL. You must know the exported function in the DLL that you want to invoke. You specify the EAI DLL Transport as one of the business services in your workflow.

NOTE: The EAI DLL Transport only accepts String type as input or output to the external DLL. The external function also must return String type.

Configuring the EAI DLL Transport

The EAI DLL Transport supports sending messages with the following methods:

- Send
- Send and Receive Response

About EAI DLL Transport Parameters

Use the Send or Send and Receive Response methods as needed when you want to pass data from the Siebel Database to an external system. These methods require an input property set. In addition to the common parameters described in [Chapter 2, “EAI Transports and Interfaces Overview,”](#) the EAI DLL Transport takes the parameters presented in [Table 17](#).

Table 17. EAI DLL Transport Parameters

Argument	Description
DLLName	Name of the (request/response) DLL.
ExternalFunction	Function in the DLL to invoke.
Return Value	The return value from the function called. This value is an output property.

To call a function in an external DLL

- 1 Create a workflow process.
NOTE: For details on Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.
- 2 Set the first business service, after the Start, to use the EAI DLL Transport. Usually, this object is named Send.
- 3 Double-click to set the input properties for the EAI DLL Transport.
- 4 Select a method, either Send, or Send and Receive Response.
- 5 Select the input arguments that you want to use from the list, as presented in [Table 17 on page 107](#).
- 6 Enter any output arguments required and save your work.

Creating a DLL to Call a Function in an External DLL

Following procedure illustrates how to create a DLL to use the EAI DLL Transport business service to call a function in an external DLL.

Starting in release 7.5.3, it is not necessary for the DLL to release the memory either on Microsoft Windows or UNIX. The DLL transport business service will release the memory. If the DLL does a memory deallocation, it will most likely crash. The basic assumption is that the DLL must do the memory allocation with a C-style malloc only. Any other type of allocation will not be handled properly and may even lead to crashes.

To make a DLL

- 1 Open a VC++ project (Open > New).
- 2 Select a Win32 Dynamic Link Library and give the name of the project.
- 3 In the next dialog box, select the option Simple dll project.

Following files are created by default:

- Project.cpp
- StdAfx.h
- StdAfx.cpp

- 4 Make the following changes in the StdAfx.h and Main.cpp files and check the results in the process simulator.

```
StdAfx.h
struct XMLDataBuf
{
    int         nLength;
    void*      pData;
};
```

```

extern "C" int __declspec(dllexport) TestEAI(const XMLDataBuf* Value, XMLDataBuf*
pReply);
Main.cpp
#include "stdafx.h"
#include <string.h>
#include <stdio.h>
#include <io.h>

BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved
                      )
{
    return TRUE;
}
extern "C" int __declspec(dllexport) TestEAI(const XMLDataBuf* Value, XMLDataBuf*
pReply)
{
    FILE *p;
    p = fopen("c:\\test.txt","w");
    fprintf(p,"before test");
    fprintf(p,"%s After Test",Value->pData);
    //strcpy(s,"Hello world");
    fclose(p);
    return 0;
}

```

About the EAI File Transport

The EAI File Transport helps move data between a Siebel application and an external file.

NOTE: The EAI File Transport is different from EAI XML Read from File. The EAI XML Read from File uses a Siebel Message in Hierarchical format as the output property. The EAI File Transport uses a process property with a DataType of String as the output property.

Configuring the EAI File Transport

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

- Send
- Send and Receive Response
- Receive
- Receive and Execute Service
- Receive, Execute, Send Response

Using the EAI File Transport Methods

You create a workflow to use the EAI File Transport, defining and refining the workflow as needed to meet your unique business requirements.

To create a workflow using the EAI File Transport

- 1 Create a workflow process in the Siebel application.
NOTE: For details on Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.
- 2 Set up a step in the workflow to use the EAI File Transport. Usually, this object is named Send.
- 3 Double-click to set the input properties for the EAI File Transport.
- 4 Select a method that fits your business needs.
- 5 Select the input arguments that you want to use from the list of arguments. The full list is presented in [Table 18 on page 111](#).
- 6 Enter any output arguments required and save your work.

Generating Unique Filenames

When using the EAI File Transport, you can have the system generate unique file names for you, as needed. One way is to specify the directory name only. The other way is to include \$\$ in the filename.

NOTE: If a directory is not specified when using the EAI XML Write to File, EAI XML Read from File, or the EAI File Transport business service, the FileName input argument defaults to the directory where the Siebel application is running.

Directory Only. To generate the unique file name, only enter the directory name. For example, instead of specifying the filename as d:\data\record1.xml, just specify d:\data. For every call of the Workflow Process, a unique name is generated in the directory. To find out the file name generated, specify FileName as an output argument for the File Transport Workflow Step.

Using \$\$. For generating filenames based on the \$\$ wildcard, specify the filename in the form d:\data\record\$.xml. At run time, Siebel application replaces the \$\$ with a unique row-id—for example, d:\data\record3-149.xml.

NOTE: The file name generated by using \$\$ is not returned as the output filename property.

EAI File Transport Parameters

In addition to the common parameters presented in [Chapter 2, "EAI Transports and Interfaces Overview,"](#) the EAI File Transport takes the parameters presented in [Table 18](#). These parameters can be specified as service method arguments, subsystem parameters, or user properties.

Table 18. EAI File Transport Parameters

Display Name	Argument	Description
Append To File	AppendToFile	Default is False. A value of True means that if the file exists, the method appends the message to the existing file. A value of False specifies that the method should overwrite any existing file.
Delete File after Receive	DeleteFile	Default is False. A value of True means that an attempt is made to delete the file after receiving it. If permissions prevent deletion, no error is given, but the information is traced.
File Name	FileName	<p>The name of the file to be received by the file transport.</p> <p>For the Send method, if a file name is not provided, a random name is used for the output file. You must specify an explicit path for file name. You can also use \$\$ as the wildcard symbol in the file name. For example, if you specify a file name of "file\$.xml," then Siebel creates files like file1-134.xml, fileA25.xml, and file242_12B.xml.</p> <p>For the Receive method, a specific file name must be provided. The use of wildcards such as \$\$ is not allowed. The source file is deleted upon receiving if set to True. If set to False (the default), the source file is not deleted.</p>
Response File Name	RespFileName	Name of the file containing the response when using the SendRecieve Method.
Sleep Time	FileSleepTime	<p>The timeout interval on receive calls, in milliseconds.</p> <p>This specifies the maximum amount of time that the service waits for a response. Default is 20000 milliseconds.</p>

9

Using Siebel OLE DB Provider

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.

This chapter includes the following topics:

- ["About Microsoft OLE DB" on page 113](#)
- ["About the Siebel OLE DB Provider" on page 113](#)
- ["About the Software Architecture for the Siebel OLE DB Provider" on page 114](#)
- ["Installing the Siebel OLE DB Provider" on page 115](#)
- ["Connecting Siebel Data Using OLE DB Consumers" on page 120](#)
- ["How Scripts and Custom Applications Affect Your Data" on page 127](#)
- ["Troubleshooting OLE DB" on page 138](#)

About 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 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.

About the 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 business 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

About the Software Architecture for the Siebel OLE DB Provider

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.

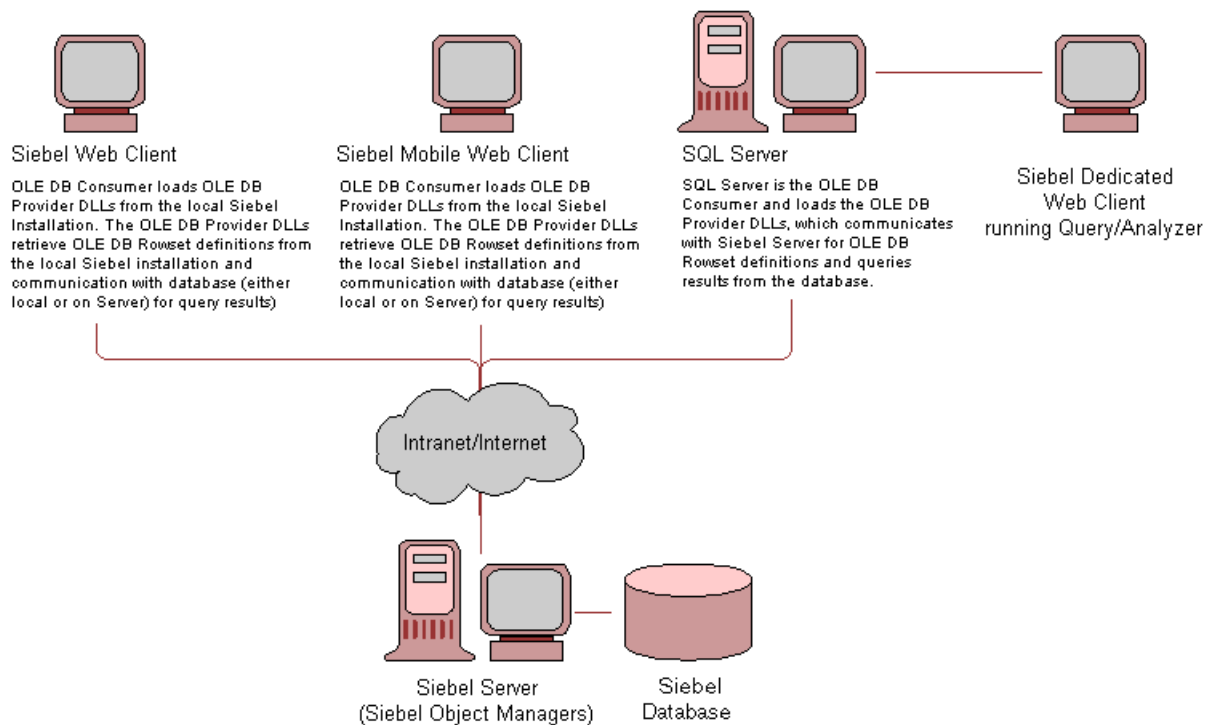


Figure 1. OLE DB Providers and Consumers of Siebel Data

Installing the 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 stand-alone client.

Siebel OLE DB Provider must be installed on the same system where the OLE DB Consumer has been installed. The DB Consumer can be installed on either the Siebel Server or one of the Siebel Web Clients.

To use Siebel OLE DB Provider with Microsoft SQL, 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 OLE DB Consumer used depends on how the Siebel OLE DB Provider is accessed:

- 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 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

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 *System Requirements and Supported Platforms* on Siebel SupportWeb for further information.

This section covers the following topics:

- [Configuring and Testing the Siebel OLE DB Provider on page 116](#)
- [Multiple Language Considerations for the Siebel OLE DB Provider on page 118](#)
- [About Primary and Foreign Key Relationships on page 119](#)
- [Viewing Siebel OLE DB Provider Events on page 119](#)
- [Viewing Siebel OLE DB Information on page 120](#)

Configuring and Testing the Siebel OLE DB Provider

You configure and test Siebel OLE DB Provider using the siebel.udl file that is installed in the *siebel\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 19 presents the necessary connection properties.

Table 19. Siebel OLE DB Provider Connection Properties

Property	Description
Data Source Name	The connection string as explained in <i>Using Siebel Tools</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 this property 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 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.

About the Siebel OLE DB Provider in the 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. 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.

About the Siebel OLE DB Provider 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 file. Table 20 below defines the syntax.

Table 20. 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 for the Siebel OLE DB Provider

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.

The following example illustrates this situation.

- Siebel OLE DB Provider is installed with Japanese language support. The default language is specified as JPN. In that case, the connect string for the Siebel OLE DB Provider is:

```
siebel://SiebelServer:2321/siebel/SCCObjMgr_JPN
```

- An additional language, English, is installed and becomes the default language. To use the Japanese language, the connect string for the Siebel OLE DB Provider would be:

```
siebel://SiebelServer:2321/siebel/SCCObjMgr,lang=JPN
```

About 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.

Viewing Siebel OLE DB Provider Events

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 Siebel 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 Siebel 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 Siebel 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 is the Siebel code for the Language Pack you are installing for this server, such as 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.

About Siebel OLE DB and Multivalue Fields

Siebel allows you to create rowsets that contain multivalue fields (MVF). 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 Product field or column will contain only the first product in the Multivalue fields list of products for the opportunity.

Connecting 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 creating and modifying Siebel OLE DB rowsets.

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.

To create the Siebel OLE DB Rowset object in Siebel Tools

- 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.
- 4** Select the items from the drop-down lists to define your rowset as follows:
 - a** Choose the locked project from which you will be creating the OLE DB rowset.
 - b** 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.

- c 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), such as **Account_Contact_1**. You can change this name as needed.

NOTE: Siebel Tools limits this field to a maximum of 75 characters. If the combination of the business object and the business component names is 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*.

- d Choose the Visibility Type for the rowset.

NOTE: 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 to 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 the field names between and including your two selections.

- 6 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.

- 7 Click Next.

The OLE DB Wizard displays the OLE DB Rowset. This page allows 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.

- 8 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 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 convert 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.

- 5** Click Next and 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.
- 7** Click Next to get to the finish page 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.
- 8** If you are satisfied with your choices, click Finish.

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

- 9** Recompile the .srf file.

Siebel OLE DB Provider retrieves the 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 Microsoft Office Applications

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

Business analysts 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.

To view Siebel data in Microsoft Excel

- 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 than 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 *.rqy 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 21 shows the required properties for an Excel query file.

Table 21. Excel Query Properties

Property	Value/Description
QueryType	OleDb.
Version	1.
CommandType	Default.
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 prompts 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: You should not have any blank lines between each commands such as QueryType and CommandText. Connection information should be on one continuous line.

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.

To view Siebel data in Microsoft Access

- 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', 'ConnectString'; 'UserId'; 'Password', Siebel OLE
DB Provider query text)
```

For example:

```
SELECT * from OPENROWSET('SiebelOLE DB.Provider.1',
'someIhost,siebel,objmgr,w_name';
'SADMIN'; 'SADMIN',
SELECT "First Name", "Last Name" from Contact_Contact_1 where ("Job Title" =
"Manager")
```

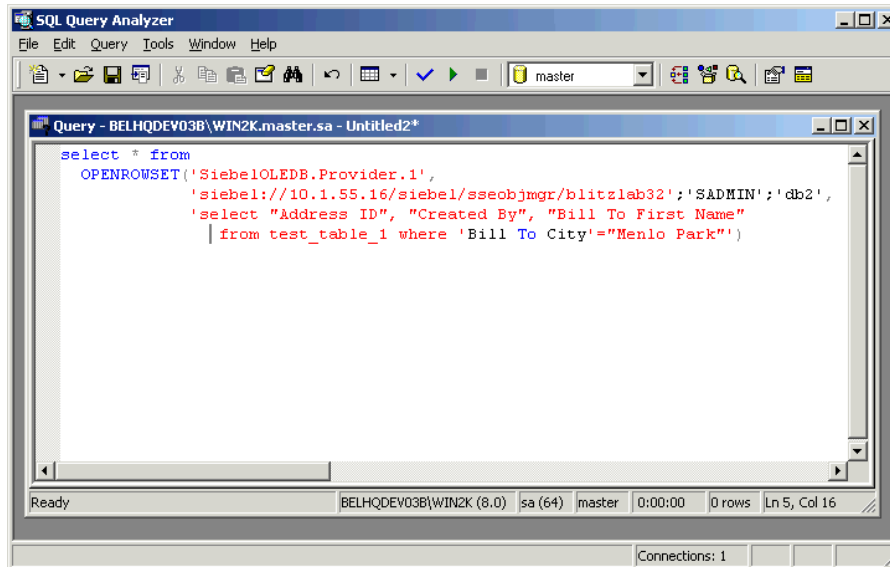
NOTE: For more information about Microsoft SQL Server distributed queries, please refer to Microsoft SQL Server 7.0 documentation.

Database administrators (DBAs) find the OLE DB support in Siebel eAI useful for checking data integrity in Siebel applications and 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"')
```

How Scripts and Custom Applications Affect Your Data

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 any way using Siebel OLE DB Provider connection.

This section covers these topics:

- [Writing an OLE DB Consumer](#)
- [Retrieving Siebel Data Using VB and ASP on page 130](#)

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

About 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 22](#).

Table 22. 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	Default is DBPROMPT_NOPROMPT. Specifies the prompt mode supported for data source initialization. This provider supports every prompting mode.

About 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

About 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

About 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.

About 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 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.
```

```
ConnectionString = "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) & ConnectionString & 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 there is 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 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")
ConnectionString = "Provider=SiebelOLEDB.Provider.1;User Id=" + UserId +
";Password=" + Password + ";Data Source=siebel://MyGateway/MyEnterprise/
MyObjMgr/MyServer;"
Connection.Open ConnectionString
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.

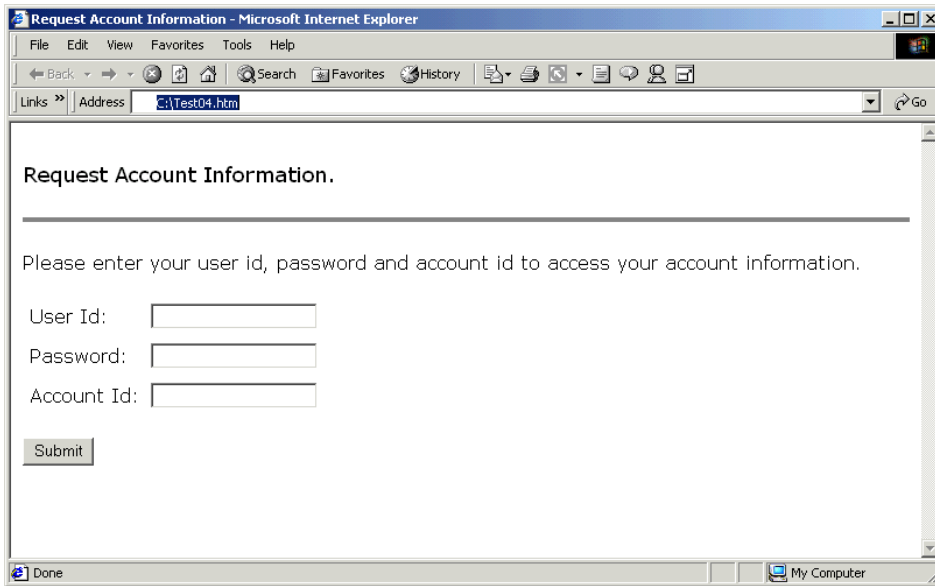


Figure 2. Sample HTML User Input Page

The output is shown in Figure 3.

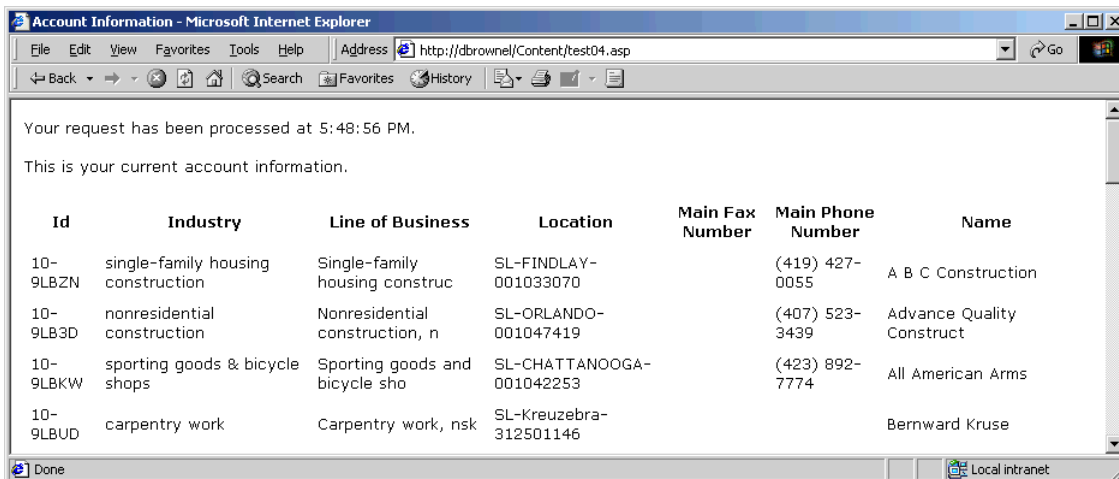


Figure 3. Sample HTML Output

Troubleshooting OLE DB

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

Error

Initialization of the data source failed with following error:

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.

Cause

You have selected an invalid .cfg file in your Connection string or the .cfg file you have selected has not been updated to support OLE DB.

Sample Problem Code

```
QueryType=OLEDB
Version=1
Connection=Provider=SiebelOLEDB.Provider.1;Persist Security Info;Data
Source="c:\siebel\client\bin\enu\uagnet.cfg,ServerDataSrc";
CommandType=Default
CommandText= select * from LabRowset
```

Solution

Check to make sure you have set the Security Info parameter in your connection string.

Error

The query did not run or the database table could not be opened and you received the following error:

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.

Cause

The database name in your .cfg file is not set correctly or the name of the table in your query is incorrect.

Sample Problem Code

```
QueryType=OLEDB
Version=1
Connection=Provider=Siebel OLEDB Provider.1;Persist Security Info=True;Data
Source="c:\siebel\client\bin\enu\siebel.cfg,ServerDataSrc";
CommandType=Default
CommandText= select * from LabRowset
```

Resolution

Check to make sure that the Provider parameter is set correctly and also that the table used for the CommandText is a valid table.

Error

SQL Query Analyzer error message. 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. Provider caused a server
fault in an external process.
```

Cause

This error typically occurs when Object Manager is not initialized or may be related to SQL Server caching of the OLEDB datasource or to the servers not being successfully restarted.

Sample Problem Code

n/a

Resolution

Restart the servers.

Error

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

Cause

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

Sample Problem Code

n/a

Resolution

Check your connection in the Siebel.UDL file and test it by providing the correct username and password. Also in the .cfg file, make sure the DataSource property in the [Siebel] section is set to the correct data source—for example, set to Sample for Sample database.

Problem

Receiving “Could not process object select * from GPTest2” error message

Description

If you did not make the custom OLE DB available to the Siebel Server Object Manager, you see the following error when you run the query:

```
Server: Msg 7357, Level 16, State 2, Line 1. Could not process object 'select * from GPTest2'
```

Sample Problem Code

n/a

Solution

Make sure that you have copied the latest .srf to the Server\objects directory and have restarted the server. These actions 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 columns in the two applications does 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 Siebel Tools.

Problem

Unable to query SODP using MSSQL Query Analyzer

Description

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 fails on the second connection attempt, and generates the following error message:

```
Error message with MSSQL: OLE DB provider "siebelOLEDB.Provider.1"reported an error. Provider caused a server fault in an external process.
```

Sample Problem Code

n/a

Solution

Restart the MS SQL Server.

10 Interfacing with Microsoft BizTalk Server

This chapter discusses the Microsoft BizTalk Server and includes the following topics:

- “About Microsoft BizTalk Server” on page 143
- “Siebel BizTalk Interface Architecture” on page 145
- “Preparing to Use the Siebel BizTalk Adapter” on page 148
- “Connecting to BizTalk Using EAI MSMQ Transport” on page 154
- “Connecting to BizTalk Using COM and AIC” on page 162
- “Connecting to BizTalk Using HTTP” on page 176

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, partners in e-commerce communities, and automated procurement arrangements. The Siebel interface for BizTalk Server provides the following features and functionality.

- **Generation and Transformation of XML Documents.** To convert 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 convert them 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 applications 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 converts 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*.

- **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 23 shows other sources of information on associated technologies.

Table 23. Other Information Sources

Technology	Reference
EAI HTTP Transport	Chapter 7, “EAI HTTP Transport” in this book
eScripts	<i>Using Siebel Tools</i>
ActiveX	<i>Using Siebel Tools</i>
Microsoft BizTalk Server (BTS)	Microsoft BizTalk Server documentation. (Available with BizTalk or download from http://www.microsoft.com .)
Siebel Installation	<i>Siebel Installation Guide for Microsoft Windows: Servers, Mobile Web Clients, Tools</i> <i>Siebel Installation Guide for UNIX: Servers, Mobile Web Clients, Tools</i>
Siebel Integration Objects	<i>Integration Platform Technologies: Siebel eBusiness Application Integration Volume II</i>

Table 23. Other Information Sources

Technology	Reference
Siebel Workflow	<i>Siebel Business Process Designer Administration Guide</i>
XML (Siebel-specific)	<i>XML Reference: Siebel eBusiness Application Integration Volume V</i>

Siebel BizTalk Interface Architecture

Communication between Siebel applications and Microsoft’s BizTalk Server is based on the Siebel eBusiness Application Integration (Siebel eAI) framework. The BizTalk Server provides the technology for application integration and data transformation. As illustrated in Figure 4, 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.

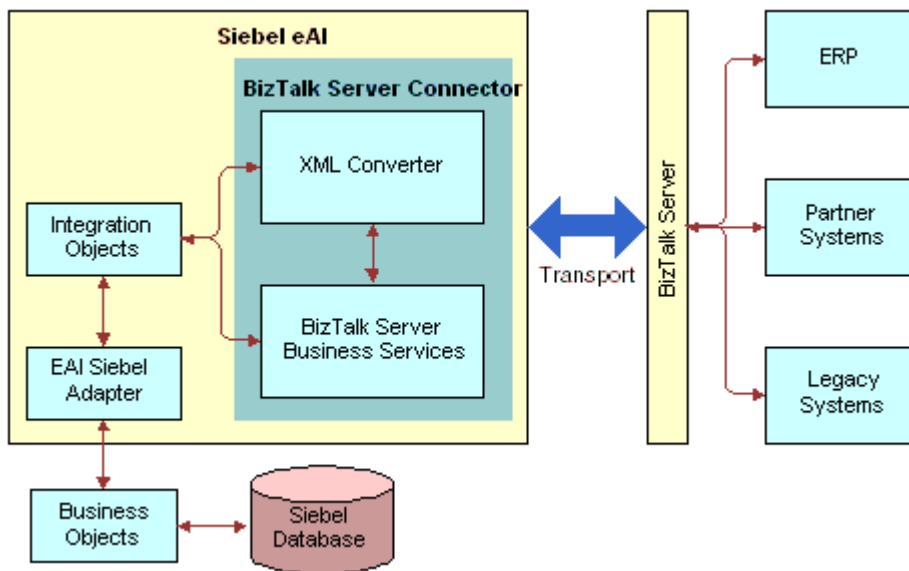


Figure 4. Siebel BizTalk Interface Architecture

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 is 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.

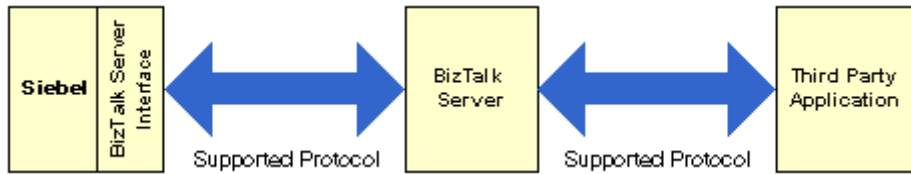


Figure 5. Siebel Application and BizTalk Inbound and Outbound Interfacing

- **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. For example, if IBM’s WebSphere Commerce Suite (WCS) sends information to a Siebel application to generate an order, the request will be in the form of an XML document that the Siebel application can understand because BizTalk Server acts as a communications bridge by mapping the data appropriately for both systems.

Suppose your partner needs to get an order to you. They initially send it as an ANSI X12 EDI document using WCS to BizTalk Server. BizTalk takes this XML document and converts it into an XML document for a Siebel application and sends it over HTTPS to the Siebel application. After the document has been accepted by the Siebel application, a workflow is triggered that processes the order and sends a response as an XML document back to BizTalk. There, BizTalk remaps the document from XML to EDI and ships it off to WCS, which, upon receiving it, performs the necessary actions.

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 a 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 is 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. Because 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.

In one last scenario, the Siebel application sends an order in XML format to the IT group's BizTalk Server over the HTTP protocol (or by calling BizTalk Server Interchange COM interface). The IT group's BizTalk Server sends the order to SAP by first converting the XML order into an SAP IDOC and sending it by means of HTTP (or any other protocol as defined in BizTalk Port specifications) to SAP. SAP creates the order in SAP R/3 and sends back an Order Status in IDOC format to the IT group's BizTalk Server. The IT group's BizTalk Server converts the SAP IDOC into an XML Order Status document and the BizTalk Server sends the document by means of HTTP or AIC to the Siebel application, where the corresponding order is updated with the new order status.

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. Use the setup here as a guideline, but your setup should reflect decisions for using BizTalk that were made at your organization.

Also, 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

Use the following checklist as a guideline to install software and create BizTalk doc specification 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 149](#).

- ❑ 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 150](#) and ["About Siebel Data Types Mappings in BizTalk" on page 153](#).

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.

You should have multiple-machine configuration—Siebel Server, BizTalk Server, and Siebel clients. However, your setup may be different and your configuration may vary from the suggested setup.

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 installed on the Siebel Server machine. Your deployment, however, might 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: For Siebel Server in a UNIX environment, you *must* install the EAI Connectors again separately under Windows. There is no UNIX version of BizTalk Server. One possibility is to install on the same machine with the BizTalk Server. Note that the EAI Connectors installation is not shipped on the UNIX CD, only on the Windows CD. For information on the supported UNIX and Windows platforms, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

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

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

You export either DTD or XDR schema. These exported files are imported into BizTalk where they are converted 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.

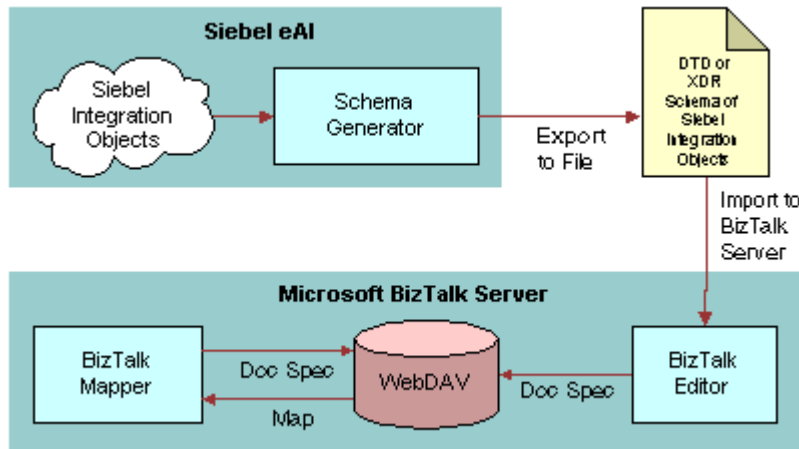


Figure 6. Exposing Siebel Integration Objects to BizTalk Server

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.

Exposing Integration Objects to BizTalk

Use the following checklist as a guideline for how to expose 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 need).

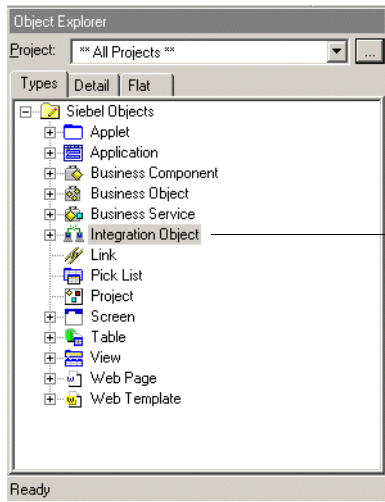
For details, see [“To generate a new DTD or XDR schema” on page 152.](#)

- ❑ 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 153.](#)

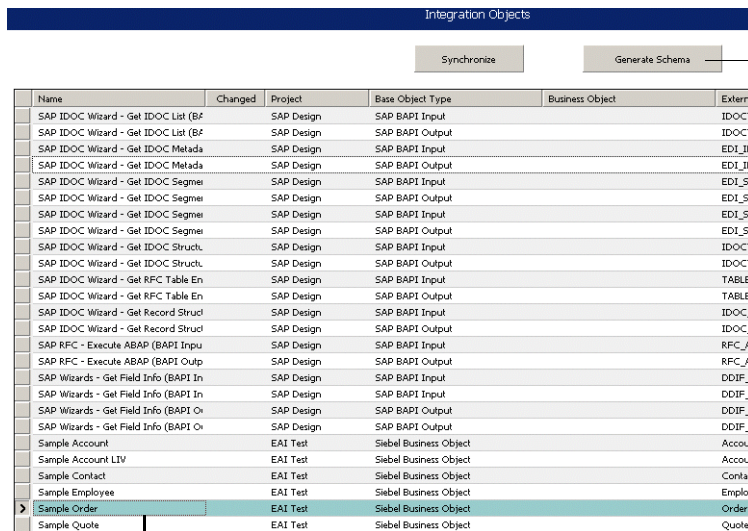
To generate a new DTD or XDR schema

- 1 Start Siebel Tools.
- 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.



Click the Generate Schema button after you select the integration object

Select an integration object; here "SampleOrder" is selected

The Generate Schema Wizard appears.

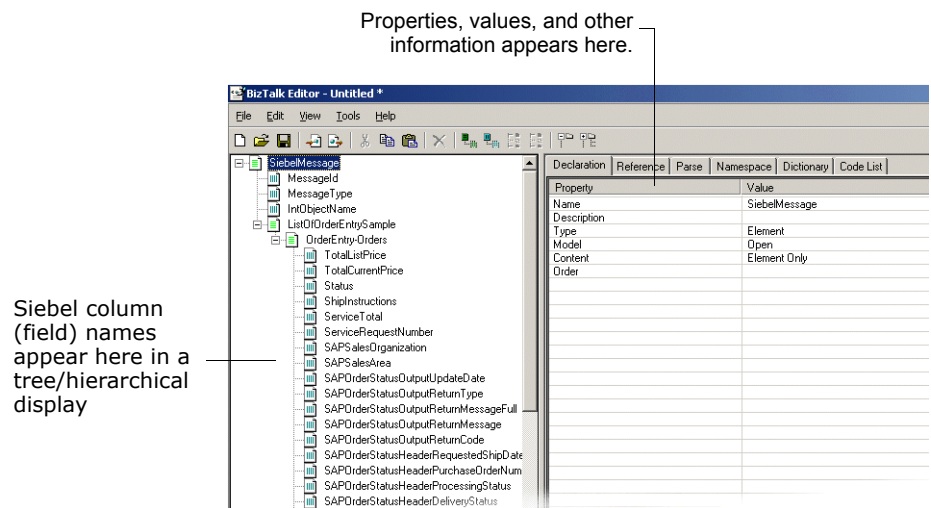
- 4 Perform the following in the Generate Schema Wizard:
 - a Select EAI XML XDR Generator from the Business Service drop-down list.
 - b Select EAI Siebel Message Envelope Service from the Envelope drop-down list.

- 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 to complete the process.
- A file (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.

To import schema into BizTalk Server to save it as a BizTalk document specification

- 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.

The columns of the integration object display in the BizTalk Server Editor, samples of which are shown in the next figure.



- 5 Choose File > Store to WebDav.
- 6 Give the file a name (such as Siebel Orders).
- 7 Click Save.

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

About Siebel Data Types Mappings in BizTalk

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

Table 24 presents data types for the Siebel Field object type and the current corresponding mapping to XDR data types.

Table 24. 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 EAI MSMQ Transport

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

The BizTalk Server Adapter uses the EAI MSMQ Transport to exchange messages with the BizTalk Server over MSMQ. See [Chapter 4, "EAI MSMQ Transport,"](#) for specific details about that transport.

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

Using EAI MSMQ Transport for Outbound Messaging

This section describes how to use the Siebel BizTalk interface to send an XML document to BizTalk Server using the EAI MSMQ Transport.

Checklist

- Set up the EAI MSMQ Transport and set up queue to send messages.

For details, see ["Configuring EAI MSMQ Transport for Various Send and Receive Scenarios"](#) on page 36.

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

For details, see ["Setting Up an MSMQ Queue for Sending Messages."](#)

- ❑ Create the workflow to process outbound documents from Siebel applications using EAI MSMQ Transport.

For details, see ["Sending Outbound Messages with EAI MSMQ Transport"](#) on page 38.

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. For details, see [Chapter 4, "EAI MSMQ Transport"](#) and the Microsoft MSMQ documentation.

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 Oracle, enter Oracle.
- 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 153](#).
- 5 Create a messaging port to your trading partner.
 - a In the Destination Organization window, select an organization, such as Oracle.
 - 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" such as, Siebel To Oracle Channel. Click Next.
 - b Select the appropriate inbound document definition, such as Siebel Order.
 - c Select the appropriate outbound document definition for your trading partner, such as Oracle Order.
 - d Select an appropriate map, if necessary.
 - e 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 From the Site Map choose Administration - Business Process > Workflow Processes.
- 2 Create a new workflow process as shown below to send a message to BizTalk.



NOTE: For information on how to set up Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.

- 3 Set up the process properties for your workflow as follows:

Name	Data Type	Value
OrderMessage	Hierarchy	-
<Value>	String	-
Object Id	String	<i>record # for an order</i>

- 4 Set up the first step after Start to use the EAI Siebel Adapter with the Query method to query the order from Siebel Database, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Process Property Type
Output Integration Object Name	Literal	Sample Order	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
OrderMessage	Output Argument	-	Siebel Message

- 5 Set up the second step to use the EAI XML Converter with the Integration Object Hierarchy to XML Document method and the following input and output arguments to convert the order object to an XML document:

Input Arguments	Type	Value	Property Name	Property Data Type
XML Character Encoding	Literal	UTF-16	-	-
Siebel Message	Process Property	-	Order Message	Hierarchy

Property Name	Type	Value	Output Argument
<Value>	Output Argument	-	XML Document

- 6 Set up the last step to use EAI MSMQ Transport with the Send method and the following input arguments:

Input Arguments	Type	Value	Property Name	Process Property Type
MsmqPhysicalQueueName	Literal	private\$\ <i>name of the queue</i>	-	-
MsmqQueueMachineName	Literal	Machine Name that owns the MSMQ queue	-	-
LargeMessageSupport	Literal	False	-	-
Message Text	Process Property	-	<Value>	String

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.

- 7 Save your workflow process and test it using Workflow Process Simulator.

NOTE: For details on setting up and using Siebel EAI MSMQ Transport, see [Chapter 4, "EAI MSMQ Transport."](#)

Using EAI MSMQ Transport for Inbound Messages

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

Checklist

-
- ❑ Set up Siebel EAI MSMQ Transport and set up 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 36.](#)

 - ❑ Set up the organization, port, and channel (collectively known as BizTalk configuration objects) for use with MSMQ Transport.
For details, see [“Setting Up an MSMQ Queue for Receiving Messages.”](#)

 - ❑ Create the workflow to process inbound documents from Siebel using MSMQ transport.
For details, see [“Receiving Messages from MSMQ” on page 45.](#)
-

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 [Chapter 4, “EAI MSMQ Transport”](#) 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 Oracle, enter Oracle.
- 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 153.](#)
- 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, such as `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 in [Step 5](#).
 - a Create a new channel "From an Organization" such as Oracle To Siebel MSMQ Channel. Click Next.
 - b Select the appropriate inbound document definition, such as Oracle Order.
 - c Select the appropriate outbound document definition for your trading partner, such as Siebel Order.
 - d Select an appropriate map, if necessary.
 - e If required, configure the Advanced properties in the final Channel Configuration page to provide authentication information.
 - f 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](#).

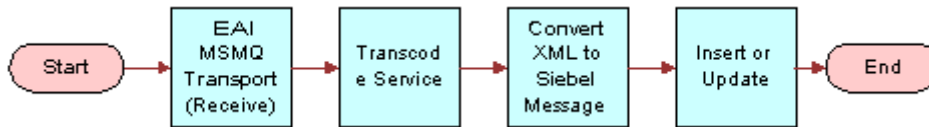
NOTE: This example uses File transport to receive documents from trading partners for the sake of illustration. In your actual business situations, a trading partner can deliver messages to a Siebel application over any supported transport.

- 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 and provide the information.
 - Name.** The name of the File receive function, such as Oracle to Siebel MSMQ.
 - Comment.** Add a brief description (optional).
 - Server.** Server on which the receive function runs.
 - Polling Location.** Enter c:\MSMQ.
- NOTE:** This is the location where your trading partner delivers the XML documents for the Siebel applications so the channel can pick it up from there and send it to Siebel application for processing.
- File Types To Poll For.** Type the extension of the files that BizTalk Server receives. In this case, type *.xml.
- d Click the Advanced tab and enter a Channel Name, such as Oracle to Siebel MSMQ Channel.
- 8 Click OK to finish.

To create the Siebel workflow for MSMQ Inbound

- 1 From the Site Map choose Administration - Business Process > Workflow Processes.

- 2 Create a new workflow process as shown below to receive a message.



NOTE: For information on how to set up Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.

- 3 Set up the process properties for your workflow as follows:

Name	Data Type	Value
OrderMessage	Hierarchy	-
<Value>	String	-
Object Id	String	-

- 4 Set up the first step after Start to use the EAI MSMQ Transport business service with the Receive method and the following input and output arguments:

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	<Value>	String
MsmqPhysicalQueueName	Literal	private\$\name of the queue	-	-
MsmqQueueMachineName	Literal	Machine Name that owns the MSMQ queue	-	-
LargeMessageSupport	Literal	False	-	-
IgnoreCorrelationId	Literal	True	-	-

Property Name	Type	Value	Output Argument
<Value>	Output Argument	-	Message Text

NOTE: Large Message Support must be disabled and Ignore Correlation Id must be set to True for Siebel applications 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. For details on setting up and using the Siebel EAI MSMQ Transport, see [Chapter 4, "EAI MSMQ Transport."](#)

- Set up the next step to use the Transcode Service business service to convert the incoming data to UTF-8 format.

NOTE: XML documents are sent in UTF-16 format from BizTalk Server to Siebel applications. In order for the XML Converter to correctly process, the incoming document must be converted to UTF-8 format through the Transcode Service.

Input Arguments	Type	Value	Property Name	Property Data Type
Conversion Mode	-	EncodingToEncoding	-	-
Source Encoding	-	UTF-16	-	-
Target Encoding	-	UTF-8	-	-
<Value>	Process Property	-	-	<Value>

Property Name	Type	Value	Output Argument
<Value>	Output Argument	-	<Value>

- Set up the third step to use the EAI XML Converter with the XML Document to Integration Object Hierarchy method using the following input and output arguments. This step converts the XML document to a Siebel object.

Input Arguments	Type	Value	Property Name	Property Data Type
XML Document	Process Property	-	<Value>	

Property Name	Type	Value	Output Argument
OrderMessage	Output Argument	-	Siebel Message

- Set up the last step to use the EAI Siebel Adapter with the Insert or Update method to update the Siebel Database, using the following input argument.

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Siebel Message	Hierarchy

- Save your workflow process and test it using Workflow Process Simulator.

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

Connecting to BizTalk Using COM and 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 (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 148.](#)

-
- 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 162.](#)

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, such as 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, such as BiztalkInterchange.MSI and BizTalkInterchange.MSI.cab.

- 9 Copy these files to the Siebel client machine and then, on the Siebel client machine, 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 applications 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 Siebel Workflow. You would run the workflow as fits your needs, using any of the usual mechanisms for running workflows in Siebel applications: Workflow Process, eScripting, and so on.

Checklist

- Set up the Siebel COM Outbound Transport.

For details, see ["Setting Up the COM Outbound Transport" on page 162](#).

- 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 164](#), and ["COM Outbound Transport Parameters" on page 163](#).

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

For details, see ["To create the Siebel workflow for COM Outbound" on page 165](#).

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 25 presents the outbound parameters.

Table 25. 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.

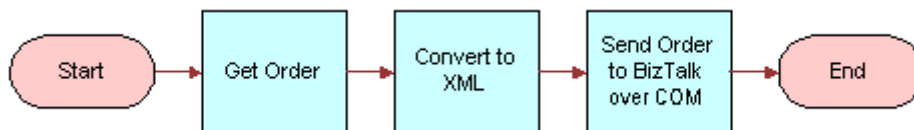
To set up BizTalk configuration objects

- 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 Oracle.
 - NOTE:** Throughout this chapter, many steps are illustrated with the use of an example. This example uses Oracle, but your actual business requirements 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, such as Siebel Order. See “To import schema into BizTalk Server to save it as a BizTalk document specification” on page 153.
- 5 Create a messaging port to your trading partner.
 - a In the Destination Organization window, select an organization, such as Oracle.

- 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" such as Siebel To Oracle Channel. Click Next.
 - b Select the appropriate inbound document definition, such as Siebel Order.
 - c Select the appropriate outbound document definition for your trading partner, such as Oracle Order.
 - d Select an appropriate map, if necessary.
 - e Complete other pages as required and select Finish to complete.

To create the Siebel workflow for COM Outbound

- 1 From the Site Map choose Administration - Business Process > Workflow Processes.
- 2 Create a new workflow process as shown below to send a message to BizTalk.



NOTE: For information on how to set up Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.

- 3 Set up the process properties for your workflow as follows:

Name	Data Type	Value
OrderMessage	Hierarchy	-
<Value>	String	-
Object Id	String	<i>record # for an order</i>

- 4 Set up the first step after Start to use the EAI Siebel Adapter with the Query method to query the order from Siebel Database, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Process Property Type
Output Integration Object Name	Literal	Sample Order	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
OrderMessage	Output Argument	-	Siebel Message

- 5 Set up the second step to use the EAI XML Converter with the Integration Object Hierarchy to XML Document method and the following input and output arguments to convert the order object to an XML document.

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Order Message	Hierarchy

Property Name	Type	Value	Output Argument
<Value>	Output Argument	-	XML Document

- 6 Set up the last step to use the EAI BTS COM Transport with the Send or the SendReceive method and the following input arguments.

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

Input Arguments	Type	Value	Property Name	Property Data Type
<Value>	Process Property	-	<Value>	String
Channel	Literal	Siebel to Oracle Channel (The channel you created in "To set up BizTalk configuration objects" on page 164.)	-	-

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 25 on page 164](#).

- 7 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 168
- ["Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server"](#) on page 170

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 168.](#)

 - ❑ 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 168.](#)

 - ❑ 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 172.](#)

 - ❑ Make sure that the Siebel application has the capability to run workflows—BizTalk Server sends named workflows to Siebel application expecting the Siebel application to be able to execute them.
For details, see ["To configure Siebel Server to run Workflow" on page 173.](#)
-

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

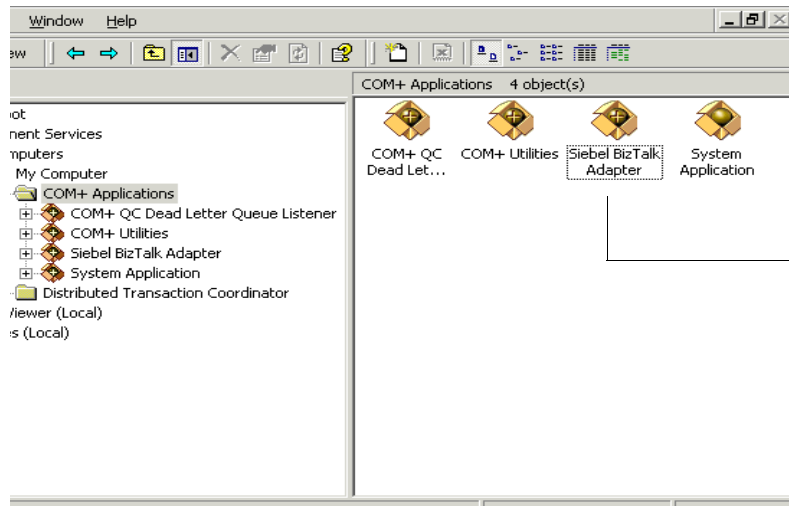
- 1** On the BizTalk machine, insert the Siebel Enterprise Server Installation CD and choose Custom Install.
- 2** Deselect other options so that only the EAI Connectors are installed.
- 3** Follow online prompts to complete the remaining pages and click Finish to install the Siebel EAI ActiveX Data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

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

- 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, such as 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” and click Next and then Finish.

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



Here is the COM+ application you just created.

- 7 Expand this newly created COM+ Server Application by double-clicking on it.

You 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.

- c Browse to the *SIEBSVR_ROOT\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 the Siebel.BizTalkHTTPAIC component registered and you should be able to see the AIC interfaces and methods.

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

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 170.](#)

 - ❑ 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 171.](#)

 - ❑ 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 171.](#)

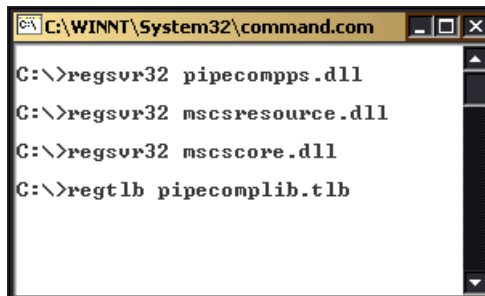
 - ❑ Copy the required Siebel Active Server Pages (ASP) to the BizTalk Server machine.
For details, see ["To configure BizTalk Server Messaging Manager with new ASP files" on page 172.](#)

 - ❑ Make sure that the Siebel application has the capability to run workflows—BizTalk Server sends named workflows to Siebel application expecting the Siebel application to be able to execute them. For details, see ["To configure Siebel Server to run Workflow" on page 173.](#)
-

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 the 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:\WINNT\System32\command.com
C:\>regsvr32 pipecompss.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.

The 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 168.

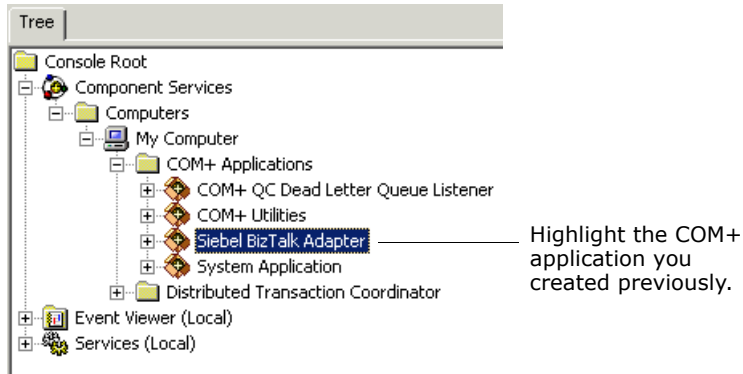
NOTE: Siebel AIC (Application Integration Components) and ADC (ActiveX Data Controls) are installed as a part of the Siebel Server or the EAI Connectors installation.

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:

SIEBSRVR_ROOT\eaiconn\bts\scripts

to BizTalk Machine:

BizTalk installation directory\MessagingManager\pipeline

To configure Siebel Server to run Workflow

- 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 application AppObjMgr_Lang (for example, SCCObjMgr_enu, for Siebel Call Center) components are installed and correctly configured.

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.

NOTE: See the *Siebel Installation Guide* for the operating system you are using and *Using Siebel Tools* 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 application AppObjMgr_Lang (for example, SCCObjMgr_enu) component, Siebel Workflow, and Siebel eBusiness Application Integration (eAI) must be enabled before following the procedure below. See the *Siebel Installation Guide* for the operating system you are using for more information.

Checklist

-
- ❑ Set up the Siebel AIC Inbound Transport.
For details, see ["Setting up the AIC Inbound Transport" on page 167](#).

 - ❑ 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" on page 173](#).

 - ❑ 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 175](#).

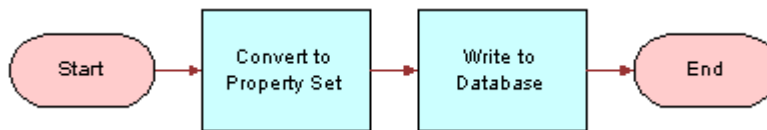
For AIC inbound connections, you define a workflow to process the inbound documents passed by the BizTalk Server using the Siebel Application Integration Component (AIC). This workflow performs the necessary steps to process the document, depending on the business requirements. For example, it could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

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

- 1 From the Site Map choose Administration - Business Process > Workflow Processes.

- 2 Set up the workflow process, as shown below, to receive and process XML document from AIC and give it a name, such as Write XML Doc.

NOTE: For details on Siebel workflows, see *Siebel Business Process Designer Administration Guide*.



- a Create the following two new process properties for your workflow:

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

NOTE: AIC passes the XML Document received from BizTalk Server to the process property with Default String set to <Value>. The same process property could 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 EAI Siebel Adapter.

- b Convert to the property set using the EAI XML Converter with the XML Document to Integration Object Hierarchy method and the following input and output arguments:

Input Argument	Type	Value	Property
MessageText	Process Property	-	InputBizdoc

Property Name	Type	Output Argument
SiebelMessage	Output Argument	Siebel Message

- c Write to the database using the EAI Siebel Adapter with the Upsert method and the following input argument:

Name	Type	Property
Siebel Message	Process Property	Siebel Message

After you create the workflow, the workflow information must be supplied to the Siebel Application Integration Component (see [Step 1 in the "To supply the workflow information for AIC Inbound" procedure](#)). 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.

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 (see [Step 2 on page 175](#)). For details, 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 the 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.

To supply the workflow information for AIC Inbound

- 1** Run the BizTalk Server Messaging Manager on the BizTalk Server machine.
 - a** Set up a Home Organization for Siebel applications, such as Siebel Sales.
 - b** Set up a Siebel Trading Partner to be the Destination Organization, such as Oracle.
 - c** Create new document specification for the XDR document definition stored in WebDav.
 - d** Set up a new BizTalk Server port to send the document to the Siebel application with the Primary Transport of Application Integration Component—BizTalk SiebelBizTalkAIC. Using BizTalk Messaging Manager, create a new port named Siebel AIC BTS.
 - e** Specify the primary transport type to be Application Integration Component.
 - f** Specify the Component Name to be the AIC component name, BizTalk.AppIntegrationSiebel.
 - g** Save your work.

BizTalk is now configured to deliver messages to Siebel BizTalk AIC.
- 2** Set up a new channel for the port created in [Step d](#).
 - 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 173](#).

- c Enter the user name, password, and connect string for the Active X Data Control to connect to Siebel Server. For example:

```
host="siebel.tcpip.none.none://SiebelServer:2321/EnterpriseServer/  
SCCObjMgr_enu" lang="ENU"
```

Where:

SiebelServer = the machine name where the Siebel Server is installed.

lang = the language of the Siebel Server. The default lang is ENU, but for other languages you have to provide this value. Also note that the quotes shown in the example are required.

CAUTION: Make sure the connect string you type is using the correct format, including the use of quotes. The correct connect string format is very important in order for ADC to correctly connect to Siebel Server.

- 3 Configure a File Receive Function from BizTalk Server Administration to poll a file location and deliver to the channel configured in [Step 2](#).

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 and provide the information.

- Name.** The name of the File receive function, such as Oracle to Siebel AIC.
- Comment.** Add a brief description (optional).
- Server.** Server on which the receive function run.
- Polling Location.** Enter c:\AIC.

NOTE: This is the location where you copy the XML Instance you created earlier. The channel can then pick it up from there and send it to the Siebel application.

- File Types To Poll For.** Type the extension of the files that BizTalk Server receives. In this case, type *.xml.

- d Click the Advanced tab and enter a Channel Name, such as AIC Oracle To Siebel Channel.

- 4 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](#)
- [How to Use EAI HTTP Transport to Receive Documents from BizTalk on page 181](#)

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 Interchange Submit() 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 177.](#)

- ❑ 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 177.](#)

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 the 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:

```
SIEBSRVR_ROOT\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.
For details, see ["Siebel HTTP Outbound Transport" on page 177](#).

 - ❑ 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 EAI HTTP Transport.
For details, see ["To create the Siebel workflow for HTTP Outbound" on page 179](#).

To set up BizTalk configuration objects for HTTP Outbound

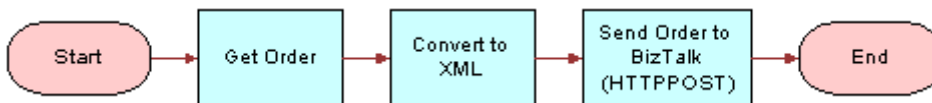
- 1 On the BizTalk Server machine, access the BizTalk Messaging Manager.
- 2 Set up your home organization as Siebel and set up applications for the home organization.
NOTE: For each BizTalk-specific step, see Microsoft BizTalk Server documentation for the details.
- 3 Create a new organization and name it as is appropriate. For example, if your trading partner is Oracle, enter Oracle.
- 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 153](#).
- 5 Create a messaging port to your trading partner.
 - a In the Destination Organization window select the organization, such as Oracle. Click Browse under Primary Transport.
 - b Select the appropriate transport for your trading partner.
 - c 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" such as Siebel To Oracle Channel. Click Next.
 - b Select the appropriate inbound document definition, such as Siebel Order.
 - c Select appropriate outbound document definition for your trading partner, such as Oracle Order.
 - d Select an appropriate map, if necessary.
 - e 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 From the Site Map choose Administration - Business Process > Workflow Processes.
- 2 Create a new workflow process as shown below to send a message to BizTalk.



NOTE: For information on how to set up Siebel Workflow, see *Siebel Business Process Designer Administration Guide*.

- 3 Set up the process properties for your workflow as follows:

Name	Data Type	Value
OrderMessage	Hierarchy	-
<Value>	String	-
Object Id	String	<i>record # for an order</i>

- 4 Set up the first step after Start to use the EAI Siebel Adapter with the Query method to query the order from Siebel Database, using the following input and output arguments:

Input Arguments	Type	Value	Property Name	Process Property Type
Output Integration Object Name	Literal	Sample Order	-	-
Object Id	Process Property	-	Object Id	String

Property Name	Type	Value	Output Argument
OrderMessage	Output Argument	-	Siebel Message

- Set up the second step to use the EAI XML Converter with the Integration Object Hierarchy to XML Document method and the following input and output arguments to convert the order object to XML document:

Input Arguments	Type	Value	Property Name	Property Data Type
Siebel Message	Process Property	-	Order Message	Hierarchy

Property Name	Type	Value	Output Argument
<Value>	Output Argument	-	XML Document

- Set up the last step to use the EAI HTTP Transport with the Send or the SendReceive method and the following input arguments.

Input Arguments	Type	Value	Property Name	Property Data Type
Message Text	Process Property	-	<Value>	String
Request Method	Literal	POST	-	-
Request URL Template	Literal	The URL to which the document will be posted—for example, <code>http://webServer/virtual directory name/SiebelBizTalkOutHTTP.asp</code> where: <i>webServer</i> is the name of your Web server and <i>virtual directory name</i> is the virtual directory you set up in "To create a virtual directory for HTTP Outbound" on page 177, such as BizTalkReceive.	-	-

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

- Save your workflow process and test it using the Workflow Process Simulator.

How to Use EAI HTTP Transport to Receive Documents from BizTalk

This section describes how to set up and use EAI 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 HTTP Transport for inbound processing 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 7, "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 170.

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 `sscaeibh.dll` file from `SIEBSRVR_ROOT\eaiconn\bts\http\` on the Siebel Server to a newly created directory, such as `c:\AIC` on the BizTalk Machine.

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:

`SIEBSRVR_ROOT\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 ["How to Use EAI HTTP Transport to Receive Documents from BizTalk"](#) on page 181.
- ❑ Set up the EAI HTTP Transport Named Subsystem and the HTTP Receive function to receive and process incoming XML documents.
For details, see ["EAI HTTP Transport Named Subsystems"](#) on page 81.
- ❑ 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 a Home Organization for Siebel applications, such as Siebel Sales.

- b** Set up a Siebel Trading Partner to be the Destination Organization, such as Oracle.
 - 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 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 the port created in [Step d](#), such as Siebel HTTP AIC Channel.

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 the 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 the 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 2](#).

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 and provide the following information.
 - Name.** The name of the File receive function, such as Siebel HTTP AIC.

- **Comment.** Add a brief description (optional).
- **Server.** Server on which the receive function runs.
- **Polling Location.** Enter `c:\http`.

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

- **File Types To Poll For.** Type the extension of the files that BizTalk Server receives. In this case, type `*.xml`.
- d** Click the Advanced tab and enter a Channel Name, such as Siebel HTTP AIC Channel.
- 5** Click OK to finish.

NOTE: If there are any errors while sending the document to the Siebel Server, they are written to the Application Event log with details.

11 Integrating with Java

This chapter discusses the integration of Java with the Siebel Application and includes the following topics:

- “About Siebel Applications and Java” on page 185
- “About the JDB Business Service API” on page 186
- “About the Siebel Code Generator” on page 187
- “About Running the Java Data Bean” on page 195
- “About the Siebel Resource Adapter” on page 200

About Siebel Applications and Java

Many enterprises, especially those involved in eBusiness, develop 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 Systems supports integration of its business services and business objects using the Siebel Java Data Bean. The Siebel Java Data Bean can be used for interaction with various kinds of Siebel application objects:

- Business objects and business components.
- Business services and property sets.
- Integration objects.

In all cases, the Java code acts as client-side proxy stub to the corresponding object on the Siebel Server. It does not implement the functionality of the object in Java.

For ease of use, the Siebel Code Generator can be used to produce Java code based on the Siebel Java Data Bean for any specific business service or integration object. This generated code has an API specific to the chosen business service or integration object.

Additionally, Siebel applications support the J2EE Connector Architecture with the Siebel Resource Adapter. The Siebel Resource Adapter supports the invocation of business services.

About the JDB Business Object API

The Java Data Bean provides an API to Siebel business objects and their business components. The API is similar in function to the API provided for other platforms, such as COM and CORBA.

Example of the Business Object and Business Component Interface

Following is a code sample demonstrating use of the business object API. The sample shows how the Java Data Bean might be used to search for a Contact with a particular login name.

The first step in using the Siebel Java Data Bean is to log in to the Object Manager of the Siebel Server. The first parameter, the connection string, specifies the protocol, server name, enterprise name, and object manager name. Once logged into the Object Manager, the methods `getBusObject` and `getBusComp` are used to obtain business objects and their business components.

The code sample activates fields to allow the query to retrieve data for the specific fields, specifies the search criteria, and executes the query. If the query is successful, the first and last name of the contact are printed to the standard output.

```
import com.siebel.data.*;

public class ObjectInterfaceExample {
    public static void main(String[] args) throws SiebelException {
        String connectString =
            "siebel://mymachine:2321/siebel/SCCObjMgr_enu";

        SiebelDataBean dataBean = new SiebelDataBean();
        dataBean.login(connectString, "USER", "PWD", "enu");
        SiebelBusObject busObject = dataBean.getBusObject("Contact");
        SiebelBusComp busComp = busObject.getBusComp("Contact");

        busComp.setViewMode(3);
        busComp.clearToQuery();
        busComp.activateField("First Name");
        busComp.activateField("Last Name");
        busComp.activateField("Id");
        busComp.setSearchSpec("Login Name", "thomas");
        busComp.executeQuery2(true,true);

        if (busComp.firstRecord()) {
            System.out.println("Contact ID: " + busComp.getFieldValue("Id"));
            System.out.println("First name: " + busComp.getFieldValue("First Name"));
            System.out.println("Last name: " + busComp.getFieldValue("Last Name"));
        }
        busComp.release();
        busObject.release();
        dataBean.logoff();
    }
}
```

If the query results in multiple records, the record set can be iterated as follows:

```
if (busComp.firstRecord()) {
    // obtain the fields/values from this record
    while (busComp.nextRecord()){
        // obtain the fields/values from the next record
    }
}
```

About the JDB Business Service API

Aside from the business object and business component API, the primary point of integration with the Siebel application is by using business services.

There are several ways to invoke a business service. The simplest way is using the Siebel Java Data Bean directly, as shown in the following example. Alternatively, Siebel Tools provides a Code Generator which creates, for any business service, Java classes that invoke the business service. The generated code may invoke the business service either using the Siebel Java Data Bean or using the Siebel Resource Adapter. The creation and use of generated code is described in the next section. The Siebel Resource Adapter is part of the J2EE Connector Architecture, which is described in [“About the Siebel Resource Adapter” on page 200](#).

Following is an example of invoking a business service directly using the Siebel Java Data Bean.

```
import com.siebel.data.SiebelDataBean;
import com.siebel.data.SiebelException;
import com.siebel.data.SiebelPropertySet;
import com.siebel.data.SiebelService;

public class BasicDataBeanTest {
    public static void main(String[] args) throws SiebelException {
        SiebelDataBean dataBean = new SiebelDataBean();
        dataBean.login("siebel://mymachine:2321/siebel/SCCObjMgr_enu", "USER", "PWD",
            "enu");

        SiebelService businessService = dataBean.getService("Workflow Utilities");
        SiebelPropertySet input = new SiebelPropertySet();
        SiebelPropertySet output = new SiebelPropertySet();
        input.setValue("Please echo this");
        businessService.invokeMethod("Echo", input, output);

        System.out.println("Output: " + output.toString());
    }
}
```

About the Siebel Code Generator

JavaBeans for invoking a particular business service can be generated using the Siebel Code Generator. These JavaBeans provide a uniform mechanism for interacting with the Siebel application from a Java or J2EE application. The JavaBean for a particular business service provides facilities for creating inputs and invoking methods. The JavaBean representing a business service can be based on either the Siebel Java Data Bean or on the Siebel JCA Resource Adapter.

For business services whose methods have integration objects as input or output, JavaBeans representing the integration objects must be generated separately. These beans provide facilities for creating the integration objects and setting their fields.

The business services most commonly used for integration are EAI Siebel Adapter and various ASI business services based on the data sync service. The methods of these business services typically have inputs and outputs that are property sets of a special type called *integration objects*. Siebel Java integration provides special support for working with integration objects.

Invoking the Siebel Code Generator

This section describes how to invoke the Siebel Code Generator to create JavaBeans for either a Siebel business service or a Siebel integration object.

To invoke the Siebel Code Generator

- 1 Start Siebel Tools.

NOTE: For information on how to use Siebel Tools consult *Using Siebel Tools*.

- 2 Select Business Service (or Integration Object) from the Types tab on the Object Explorer.

NOTE: If Integration Object is not present, add it by checking Integration Object on the Object Explorer tab of the Development Tools Options window opened by selecting View >Options.

- 3 Select the desired business service (or integration object), as shown in the following illustration for an integration object.

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.

Synchronize Generate Schema Generate Code				
W	Name	Changed	Project	Base Object Type
>	Sample Account		EAI Test	Siebel Business Object
	Sample Account LIV		EAI Test	Siebel Business Object
	Sample Contact		EAI Test	Siebel Business Object
	Sample Employee		EAI Test	Siebel Business Object
	Sample Internal Division		EAI Test	Siebel Business Object
	Sample Order		EAI Test	Siebel Business Object
	Sample Order from Ariba		EAI Demo	Siebel Business Object
	Sample Quote		EAI Test	Siebel Business Object

- 4 Click Generate Code.

- 5 Complete the Code Generator wizard:

- a Leave the business service as is (there is only one available: the Siebel Code Generator).
- b Select either Java(JDB) (Java Data Bean) or Java(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 services or integration objects is stored in subdirectories there, as explained next.
- d Click Finish.

The code is generated and the wizard closes, returning you to the Business Service or Integration Object form.

Code Generated for a Business Service

The code generated for a business service includes a class representing the business service itself as well as classes representing inputs and outputs of its methods. These classes are described in detail in this section.

ASI business services based on the data sync service have integration objects as part of the input or output of their methods. The JavaBeans representing these integration objects must be generated separately from the business service.

The classes for a given business service reside in a package in one of the following:

- `com.siebel.service.jdb.business service name` or
- `com.siebel.service.jca.business service name`

Depending on whether the beans are based on the Java Data Bean or the Siebel JCA Resource Adapter. For example, generated JDB code for the EAI Siebel Adapter resides in the package `com.siebel.service.jdb.eaisiebeladapter`.

The Code Generator creates the standard Java directory structure reflecting the package structure. As shown in [Figure 7](#), a subfolder named `com` is created in the folder specified during the generation process. Below the `com` folder is a folder named `siebel` containing a folder named `service`, containing a folder named `jdb` (or `jca`), containing a folder named for the business service. This last folder contains the classes for the business service. Each class is defined in its own file.

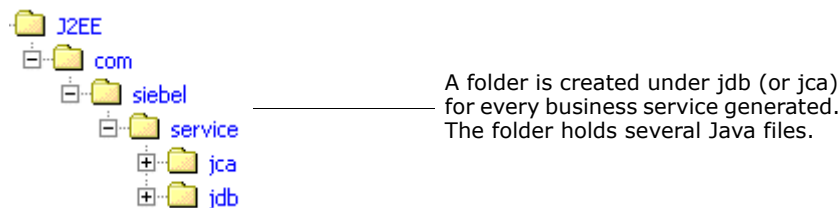


Figure 7. Directory Structure Created to Contain Java Code for Business Services

One Java class is generated to represent the business service itself. The name of the class is the name of the business service with all special characters replaced by underscores (`_`) and `BusServAdapter` appended to the end. For example, the class representing EAI Siebel Adapter is `EAI_Siebel_AdapterBusServAdapter`.

The Java class has one method for each method of the business service. Its name is the name of the method with `m` prepended. For code based on the Java Data Bean, the class is a subclass of `com.siebel.integration.adapter.SiebelJDBAdapterBase`. For code based on the Siebel Resource Adapter, the class is a subclass of `com.siebel.integration.adapter.SiebelJCAAdapterBase`.

Additionally, for each method of the business service defined in Siebel Tools, one Java class is created for the method's input and one for the method's output. The name of the class is the name of the method with *Input* or *Output* appended. The class encapsulates all input (or output) arguments for the method. Each argument is represented as a field whose name is that of the argument with *f* prepended. For each field, public set and get methods are provided Java methods for reading and writing their values.

For example, the business service CC XML Converter, which has two methods, PropSetToXML and XMLToPropSet, generates the following four classes:

- CC_XML_Converter BusServiceAdapter
- PropSetToXMLInput
- PropSetToXMLOutput
- XMLToPropSetInput

The first class, CC_XML_Converter BusServiceAdapter, represents the business service as a whole; it has methods mPropSetToXML and mXMLToPropSet. The other three classes represent the input or output parameters of the two methods. (Notice there is no class XMLToPropSetOutput because that method has no outputs.) Those three classes each have methods to read and write the individual parameters, as well as methods to convert to and from a com.siebel.data.SiebelPropertySet.

About Methods of Java Classes Generated for a Business Service

Table 26, Table 27, and Table 28 describe methods that are present in the generated Java code for every business service. Generic names (for example, GenericService, GenericMethod) are substituted for the actual names of the business service, methods, and arguments.

Table 26. Methods of the Java Class `com.siebel.service.jdb.GenericServiceBusServAdapter` Generated for an Example Business Service, `GenericService`, Having the Business Service Method `GenericMethod`

Method	Description
<code>GenericServiceBusServAdapter()</code>	Constructor that uses the default properties file, <code>siebel.properties</code> .
<code>GenericServiceBusServAdapter(SiebelDataBean)</code>	Constructor that reuses the resources of an existing <code>SiebelDataBean</code> .
<code>GenericServiceBusServAdapter(String)</code>	Constructor taking the name of the properties file to use.
<code>GenericServiceBusServAdapter(String, String, String)</code>	Constructor taking the username, password, and connect string.
<code>GenericServiceBusServAdapter(String, String, String, String)</code>	Constructor taking the username, password, connect string, and language.
<code>mGenericMethod(GenericMethodInput)</code>	Invokes the specified business service method.

Table 27. Methods of the Java Class `com.siebel.service.jdb.GenericMethodInput` Generated for an Example Business Service Method, `GenericMethod`

Method	Description
<code>GenericMethodInput()</code>	Constructor.
<code>GenericMethodInput(SiebelPropertySet)</code>	Constructor that sets its fields from the given property set.
<code>fromPropertySet(SiebelPropertySet)</code>	Copies fields values from the given property set.
<code>toPropertySet()</code>	Returns a <code>SiebelPropertySet</code> with the properties and values corresponding to the fields of this object.
<code>getfGenericArgument()</code>	Returns the value of business service method argument.
<code>setfGenericArgument(String)</code>	Sets the value of a business service method argument.

Table 28. Methods of the Java class `com.siebel.service.jdb.GenericMethodOutput` Generated for an Example Business Service Method, `GenericMethod`

Method	Description
<code>GenericMethodOutput()</code>	Constructor.
<code>GenericMethodOutput(SiebelPropertySet)</code>	Constructor that sets its fields from the given property set.
<code>fromPropertySet(SiebelPropertySet)</code>	Copies fields values from the given property set.
<code>toPropertySet()</code>	Returns a <code>SiebelPropertySet</code> with the properties and values corresponding to the fields of this object.
<code>getfGenericArgument ()</code>	Returns the value of business service method argument.
<code>setfGenericArgument ()</code>	Sets the value of a business service method argument.

About the Code Generated for an Integration Object

Integration objects are special kinds of property sets that are the input and output of business services based on the data sync service. JavaBeans based on integration objects are designed to be used with those business services or with the EAI Siebel Adapter and may be used to query, delete, upsert, and synchronize information in the Siebel Server's database.

The integration object, and each of its components, has its own Java class, stored in the package `com.siebel.local.Integration Object Name`. The class for the integration object has `IO` appended to the end, and the class for an integration component has `IC` appended. The Code Generator creates the standard Java directory structure reflecting the package structure. In the selected folder, a subfolder named `com` is created, containing a subfolder `siebel`, containing a subfolder `local`, which contains one subfolder for each integration object that was generated. The Java files are stored in the lowest directory. This structure is shown in [Figure 8](#).

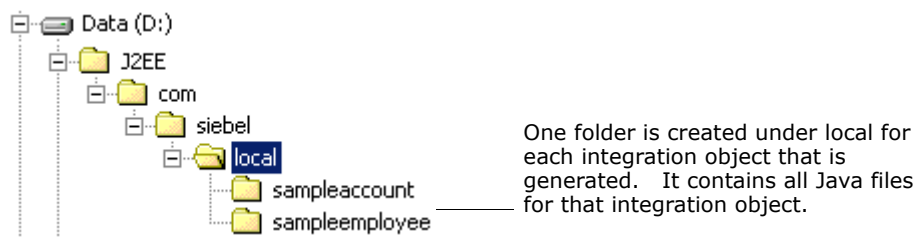


Figure 8. Directory Structure Created of Java Code Generated for Integration Objects

For example, the integration object Sample Account; which has five components Account, Account Attachment, Account_Organization, Business Address, and Contact; generates the following six classes:

- Sample_AccountIO
- AccountIC
- Account_AttachmentIC
- Account_OrganizationIC
- Business_AddressIC
- ContactIC

The first class, suffixed with IO, represents the entire integration object. It has methods to construct the object, to read and write fields, to add integration object components, and to convert to and from a SiebelPropertySet. The other five classes, suffixed with IC, represent the individual integration object components and provide methods that are for constructing the component to read and write fields and to convert to and from a SiebelPropertySet.

Methods of Java Classes Generated for an Integration Object

Table 29 describes methods that are present in the generated Java code for every integration object.

Table 29. Methods of the Java Class com.siebel.local.GenericIntObjIO Generated for an Example Integration Object, GenericIntObj

Object	Description
addfintObjInst(SiebelHierarchy)	Adds an integration object component object to the integration object.
clone	Returns a copy of the integration object.
equals(Object)	Determines whether integration object has the same data as the integration object passed.
fromPropertySet(SiebelPropertySet)	Copies the data from the given property set to the integration object.
getfIntObjectFormat	Returns a String containing the format of the integration object.
getfIntObjectName	Returns the integration object name property.
getfintObjInst	Returns a Vector representation of the integration object.
getfMessageId	Returns the MessageId property of the integration object.
getfMessageType	Returns the MessageType property of the integration object.

Table 29. Methods of the Java Class com.siebel.local.GenericIntObjIO Generated for an Example Integration Object, GenericIntObj

Object	Description
getfOutputIntObjectName	Returns the OutputIntObjectName property of the integration object.
Generic_ObjectIO()	Default constructor.
Generic_ObjectIO(SiebelPropertySet ps)	Creates an integration object (and its hierarchy) based on a property set.
setfIntObjectFormat	Sets the IntObjectFormat property of the integration object.
setfIntObjectName	Sets the IntObjectName property of the integration object.
setfMessageId	Sets the MessageId property of the integration object.
setfMessageType	Sets the MessageType property of the integration object.
setfOutputIntObjectName	Sets the OutputIntObjectName property of the integration object.
toPropertySet	Returns a SiebelPropertySet representation of the integration object.

Methods of Java Classes Generated for an Integration Object Component

Table 30 describes methods that are present in the generated Java code for every integration object component.

Table 30. Methods of the Java Class com.siebel.local.GenericIntCompIC Generated for an Example Integration Object, GenericIntComp Having Child Component GenericIntCompChild and Field GenericField

Object Component	Description
addfGenericIntCompChildIC(GenericIntCompChildIC)	Adds to the integration object component the given child integration object component.
clone	Returns a copy of the integration object.
equals(Object)	Determines whether the integration object component has the same data as the passed integration object component.

Table 30. Methods of the Java Class `com.siebel.local.GenericIntCompIC` Generated for an Example Integration Object, `GenericIntComp` Having Child Component `GenericIntCompChild` and Field `GenericField`

Object Component	Description
<code>fromPropertySet(SiebelPropertySet)</code>	Populates the integration object component based upon the contents of a property set.
<code>getfGenericIntCompChildIC</code>	Returns a Vector containing all child integration object components of type <code>ChildIntObjComp</code> associated with the integration object component.
<code>getfGenericField()</code>	Returns the value of the field <code>GenericField</code> .
<code>GenericIntCompIC()</code>	Default constructor.
<code>GenericIntCompIC(SiebelPropertySet)</code>	Creates an integration object component from a property set.
<code>setfGenericField(val)</code>	Sets the value of the field <code>GenericField</code> .
<code>toPropertySet</code>	Returns a property set representation of the integration object component.

About Running the Java Data Bean

Two Siebel .jar files are needed to compile and run a Java application that uses the Java Data Bean:

- `Siebel.jar`
- `SiebelJI_lang.jar` (*lang* = the installed language pack; for example, `SiebelJI_enu.jar` for English or `SiebelJI_jpn.jar` for Japanese.)

These jar files are provided with the standard Siebel installation under the directory `INSTALLED_DIR\classes`.

Documentation of individual classes is provided in the form of javadoc (`Siebel_JavaDoc.jar`), which is installed when installation option Siebel Java Integrator (a component of the Siebel Tools or the Siebel Server installer) is chosen. This .jar file contains the up-to-date javadoc for the Siebel Java Data Bean, Siebel Resource Adapter, and dependent classes.

Connect String and Credentials for the SiebelDataBean

When using the `SiebelDataBean` directly, without any generated code, three arguments must be passed to the `login` method. A fourth argument, language code, is optional.

- connect string
- Siebel username

- Siebel password
- language code (default is enu)

The connect string has the following form:

```
siebel://SiebelServerName:SCBPort/EnterpriseName/XXXObjMgr_lang
```

For example,

```
siebel://mymachine:2321/mysiebelenterprise/SCCObjMgr_enu
```

When using generated code, by default, these parameters are taken from the file `siebel.properties`, which must be in the classpath of the JVM. These properties are read from `siebel.properties` at the time an instance of the generated business service class is created using the parameterless constructor. They may be overridden by calling the methods `setConnectionString`, `setUserName`, `setPassword`, and `setLanguage` any time prior to calling `initialize()` or invoking a business service method (such as `mGenericMethod` in [Table 26](#)). This is the behavior when the default (parameterless) constructor of the generated Java class is used.

Alternatively, the generated class provides the following four constructors with arguments:

- One String argument: the name of the property file to be used.
- Three String arguments: the connect string, username, and password. No properties file is used.
- Four String arguments: the connect string, username, password, and language. No properties file is used.
- `SiebelDataBean` argument: the `SiebelDataBean` passed should already have parameters assigned and its login method executed.

Connection Parameters for the `SiebelDataBean`

Regardless of how the `SiebelDataBean` is invoked, certain parameters of the connection may be set using the properties file. These are `siebel.conmgr.txtimeout`, `siebel.conmgr.poolsize`, `siebel.conmgr.sesstimeout`, `siebel.conmgr.retry`, and `siebel.conmgr.jce`.

Other connection parameters may also be specified in the properties file, but they are used only in conjunction with generated code (subclasses of `com.siebel.integration.adapter.SiebelJDBAdapterBase` or `SiebelJCAAdapterBase`).

The following table, [Table 31](#), gives the details of these parameters.

Table 31. Properties in the siebel.properties File

Property	Description
siebel.conmgr.txttimeout	The number of milliseconds to wait after sending a request to the Siebel Server. Must be a positive integer; other values are ignored. The default is 600000 milliseconds (10 minutes).
siebel.conmgr.poolsize	For each object manager process, a pool of open connections is maintained and shared by all users of that process. This parameter specifies the maximum number of connections that will be stored in the pool. Its value must be a positive integer less than 500; other values are ignored. The default is 2.
siebel.conmgr.sesstimeout	The number of seconds the Siebel Server will wait before disconnecting an idle client session. Its value must be a nonnegative integer. The default is 2700 seconds (45 minutes).
siebel.conmgr.jce	Determines whether encryption of transmissions is done using Java Cryptography Extension or RSA (if the connection uses encryption). 1 indicates JCE; 0 indicates RSA. The default is 0.
siebel.conmgr.retry	The number of attempts to be made at establishing a connection (opening a session) before giving up. Must be a positive integer. The default is 3.
siebel.connection.string	The Siebel connect string. For information on the syntax of the connect string, see the <i>Siebel Object Interfaces Reference</i> guide.
siebel.loglevel	The level of messages to be logged. Must be a positive integer less than 6. Other values are ignored or throw an exception. 0 causes only FATAL messages to be logged; 1 ERROR; 2 WARN; 3 INFO; 4 DETAIL; 5 DEBUG. The default is 0.
siebel.logfile	The name of a file to which logging should be directed. Strings that cause a FileNotFoundException cause an error to be logged and are ignored. The default is to print to the JVM's standard output.
siebel.user.name	The Siebel username to be used for logging in to the Siebel Object Manager.
siebel.user.password	The Siebel password to be used for logging in to the Siebel Object Manager.

Table 31. Properties in the siebel.properties File

Property	Description
siebel.user.language	The language code indicating the natural language to be used for messages and other strings. Default is enu.
siebel.jdb.classname	The name of a subclass of com.siebel.data.SiebelDataBean to use instead of SiebelDataBean. Strings that do not specify a valid class or specify a class that is not a subclass of SiebelDataBean cause an error log to be logged and SiebelDataBean to be used instead.

Here is a sample siebel.properties file:

```
siebel.connection.string = siebel:// 172.20.94.55:2321/siebel/EAIObjMgr_enu
siebel.user.name        = User1
siebel.user.password    = password
siebel.user.language    = enu
siebel.user.encrypted   = false
siebel.conmgr.txtimeout = 3600
siebel.conmgr.poolsize  = 5
siebel.conmgr.sesstimeout = 300000
siebel.conmgr.retry     = 5
siebel.conmgr.jce       = 1
siebel.loglevel        = 0
```

Examples Using Generated Code for Integration Objects

The following code uses the code generation facilities provided in Siebel Tools. For more information see ["About the Siebel Code Generator" on page 187](#), 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.

The following is a code sample invoking the QueryByExample method of the Siebel Account business service. In addition to the generated code for Siebel Account (resident in com.siebel.service.jdb.siebelaccount), the sample uses the generated code for the integration object Sample Account (resident in com.siebel.local.sampleaccount).

The code invokes the method QueryByExample of the ASI business service SiebelAccount. The parameter to this method is formed from an instance of the integration object SampleAccount, which serves as the example, essentially specifying a search criterion of all Accounts that start with the letters Ai. The output integration object is converted to a Vector and iterated through to print the names of matching Accounts.

```
import com.siebel.data.SiebelDataBean;
import com.siebel.data.SiebelException;
import com.siebel.service.jdb.siebelaccount.Siebel_AccountBusServAdapter;
import com.siebel.service.jdb.siebelaccount.QueryByExampleInput;
import com.siebel.service.jdb.siebelaccount.QueryByExampleOutput;
import com.siebel.local.sampleaccount.Sample_AccountIO;
```

```

import com.siebel.local.sampleaccount.AccountIC;

public class JDBSiebelAccount {
    public static void main(String[] args) throws SiebelException {
        Siebel_AccountBusServAdapter svc = new Siebel_AccountBusServAdapter("USER",
            "PWD","siebel://mymachine:2321/siebel/SCCObjMgr_enu","enu");

        // Create the example-accounts starting with "Ai":
        AccountIC acctIC = new AccountIC();
        Sample_AccountIO acctIO = new Sample_AccountIO();
        acctIO.addfintObjInst(acctIC);
        acctIC.setfName("Ai*");
        QueryByExampleInput qbeIn = new QueryByExampleInput();
        qbeIn.setfSiebelMessage(acctIO);

        // Call QueryByExample
        QueryByExampleOutput qbeOut = svc.mQueryByExample(qbeIn);

        acctIO = new Sample_AccountIO(qbeOut.getfSiebelMessage().toPropertySet());
        Vector ioc = acctIO.getfintObjInst();

        // print the name of each account returned:
        if (!ioc.isEmpty()) {
            for(int i=0; i < ioc.size(); i++) {
                acctIC = (AccountIC) ioc.get(i);
                System.out.println(acctIC.getfName());
            }
        }
    }
}

```

The following example uses the generated code for the business service EAI Siebel Adapter. An instance is instantiated using the constructor that takes an instance of SiebelDataBean. The QueryPage method is called; its output is actually a SampleAccount integration object, but the object returned is not strongly typed and instead is used to construct a SampleAccount instance. The generated code for SampleAccount is also needed for this example.

```

import com.siebel.data.SiebelDataBean;
import com.siebel.data.SiebelException;
import com.siebel.local.sampleaccount.AccountIC;
import com.siebel.local.sampleaccount.Sample_AccountIO;
import com.siebel.service.jdb.eaisiebeladapter.EAI_Siebel_AdapterBusServAdapter;
import com.siebel.service.jdb.eaisiebeladapter.QueryPageInput;
import com.siebel.service.jdb.eaisiebeladapter.QueryPageOutput;

public class DataBeanDemo {
    public static void main(String[] args) throws SiebelException {
        SiebelDataBean m_dataBean = new SiebelDataBean();
        String conn = "siebel://mymachine:2321/siebel/SCCObjMgr_enu";
        m_dataBean.login(conn, "USER", "PWD", "enu");
        // Construct the EAI Siebel Adapter, using the data bean
        EAI_Siebel_AdapterBusServAdapter svc =
            new EAI_Siebel_AdapterBusServAdapter(m_dataBean);
        svc.initialize();
        try {
            // Set values of the arguments to the QueryPage method.

```

```

QueryPageInput qpInput = new QueryPageInput();
qpInput.setfPageSize(Integer.toString(10)); // Return 10 records.
qpInput.setfOutputIntObjectName("Sample Account");
qpInput.setfStartRowNum(Integer.toString(0)); // Start at record 0.
QueryPageOutput qpOutput = svc.mQueryPage(qpInput);

// Construct the integration object using the QueryPage output
Sample_AccountIO acctIO =
    new Sample_AccountIO(qpOutput.getfSiebelMessage().toPropertySet());

// Convert the results to a vector for processing
Vector ioc = acctIO.getfintObjInst();

// Print name of each account
if (!ioc.isEmpty()) {
    for (int i = 0; i < ioc.size(); i++) {
        AccountIC acctIC = ((AccountIC) ioc.get(i));
        System.out.println(acctIC.getfName());
    }
}
} catch (SiebelException e) {}
finally {
    m_dataBean.logoff();
}
}
}

```

About the Siebel Resource Adapter

The Siebel Resource Adapter is for use within the J2EE Connector Architecture (JCA) by J2EE-based applications (EJBs, JSPs, servlets) that are deployed on containers. JCA provides clients with a standard interface to multiple enterprise information services such as the Siebel application.

The Siebel Resource Adapter implements system-level contracts that make able a standard J2EE application server to perform services such as pooling connections and managing security. This is referred to as operation within a *managed environment*. The J2EE Connection Architecture also provides for operation in a *non-managed environment*, where the client need not be deployed in a J2EE container, but instead uses the adapter directly. In this case, the client takes responsibility for services such as managing security.

The Siebel Resource Adapter has transaction support level *NoTransaction*. This means that the Siebel Resource Adapter does not support local or JTA transactions.

Using the Resource Adapter

When deploying the Siebel Resource Adapter in J2EE Application Server (BEA, WebLogic, or IBM WebSphere), the necessary Siebel JAR files are included. The Siebel JAR files that need to be added to the classpath are:

- SiebelJI.jar

- `SiebelJI_lang.jar` (*lang* = the installed language pack; for example, `SiebelJI_enu.jar` for English or `SiebelJI_jpn.jar` for Japanese.)

The resource adapter archive, or RAR file, may also be required for deployment. Refer to the documentation of the J2EE application server for more information on deploying a JCA adapter on the server.

The following sections contain code samples for both managed and non-managed environments.

About the Connect String and Credentials for the Java Connector

The Java Connector Architecture allows for credentials to be supplied using either *Container-Managed Sign-on* or *Application-Managed Sign-On*.

With Container-Managed Sign-On, the application server's container identifies the principal and passes it to the JCA adapter in the form of a JAAS Subject. Application servers provide their own system of users and roles; such a user must be mapped to Siebel user and password for the purpose of the JCA adapter. Application servers allow the specification of such mappings. With Container-Managed Sign-On, the Siebel connect string and language must be specified in the deployment descriptor of the adapter (`ra.xml`). If a Siebel username and password are present in the descriptor, they will be used by the application server only to create an initial connection to the Siebel application when the application server is started, which is not necessary.

With Application-Managed Sign-On, the client application must provide the credentials and connect string. This is done just as for the Java Data Bean, as described in ["About Running the Java Data Bean" on page 195](#), by either supplying them in `siebel.properties` or setting them programmatically using `setUserName`, `setPassword`, `setConnecString`, and `setLanguage`. If any of these parameters are supplied using Application-Managed Sign-On, then all four of them should be supplied in that manner.

NOTE: Connection parameters beginning with `siebel.commgr` are read from `siebel.properties`, whether the adapter is being used in managed or non-managed mode.

Managed Code Sample Using the Siebel Resource Adapter

The following is a code sample using the Siebel Resource Adapter in a managed environment. The sample is a servlet that makes a simple invocation to a business service using the generated JCA code. (For more information on generating code, see ["About the Siebel Code Generator" on page 187](#).)

The JCA ConnectionFactory is obtained through JNDI. Credentials are obtained at run time from the JAAS Subject passed to the servlet. The connect string and language are obtained from the deployment descriptor (`ra.xml`). Other connection parameters are obtained from the `siebel.properties` file.

```
import javax.naming.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.siebel.integration.jca.cci.SiebelConnectionFactory;
import com.siebel.service.jca.eaifiletransport.*;
```

```

public class ManagedConnectionServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        PrintWriter reply = response.getWriter();

        try {
            EAI_File_TransportBusServAdapter bs = new EAI_File_TransportBusServAdapter();
            InitialContext jndi = new InitialContext();
            SiebelConnectionFactory scf = (SiebelConnectionFactory)jndi.lookup("siebelJCA");
            bs.setConnectionFactory(scf);
            // Username and password obtained from JAAS Subject passed by server at runtime.
            // Connect string and language obtained from deployment descriptor, ra.xml.

            ReceiveInput input = new ReceiveInput();
            input.setfCharSetConversion("UTF-8");
            input.setfFileName("D:\\helloworld.txt");
            ReceiveOutput output = bs.mReceive(input);

            reply.println(output.getf_Value());
        }
        catch (Exception e) {
            reply.println("Exception:" + e.getMessage());
        }
    }
}

```

Non-Managed Code Sample Using the Siebel Resource Adapter

The following is a code sample using the Siebel Resource Adapter in a non-managed environment. The sample performs the same function as the Managed sample; it is a servlet that makes a simple invocation to a business service using the generated JCA code. (For more information on generating code, see ["About the Siebel Code Generator" on page 187.](#))

The JCA ConnectionFactory is created directly. The username, password, connect string, and language are obtained from siebel.properties or set programmatically. Other connection parameters are obtained from the siebel.properties file.

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.siebel.integration.jca.cci.notx.SiebelNoTxConnectionFactory;
import com.siebel.service.jca.eaifiletransport.*;

public class BookshelfNonManagedConnectionsSample extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        PrintWriter reply = response.getWriter();

        try {
            EAI_File_TransportBusServAdapter bs = new
            EAI_File_TransportBusServAdapter();

            bs.setConnectionFactory(new SiebelNoTxConnectionFactory());
            // Credentials, connect string, and language are read from siebel.properties,
            // which must be in the classpath of the servlet.
            // Alternatively, these may be set here programmatically:
            //bs.setUsername("USER");
            //bs.setPassword("PWD");
        }
    }
}

```

```
//bs.setConnectString("siebel://mymachine:2321/siebel/SCCObjMgr_enu");  
  
ReceiveInput input = new ReceiveInput();  
input.setfCharSetConversion("UTF-8");  
input.setfFileName("D:\\helloworld.txt");  
ReceiveOutput output = bs.mReceive(input);  
  
reply.println(output.getf_value_());  
  
}  
catch (Exception e) {  
    reply.println("Exception:" + e.getMessage());  
}  
}  
}
```


Index

Symbols

- .cfg file entries, future support of** 81
- .jar files**
 - about and list of 200
- .rpy file, about using to store query** 124

Numerics

- 6.x, EAI MQSeries Transport, about upgrading workflows** 21
- 7.x, EAI MQSeries Transport, about upgrading workflows** 21

A

Access

- ad hoc reports, about using for 125
- Siebel application data, viewing in 125
- Siebel OLE DB Provider, about accessing 116

Access XP, about support of

Active Server Pages

- BizTalk Server Messaging Manager with ASP files, configuring 172
- BizTalk Server, copying to 182
- HTTP Outbound directory, copying to 177
- Siebel OLE DB Provider, about accessing 116
- viewing Siebel application data, code sample 134

Active X Data Control, connect string

addfChildIntObjComp integration object component

- 194

addfIntObject method, about

AIC Inbound Transport

- about 167
- BizTalk Server, setting up to run locally (checklist) 168
- BizTalk Server, setting up to run on 181
- installing 168
- receiving BizTalk messages, checklist for 173
- registering as a COM+package remote communication 171
- registering as a COM+server application 168
- Siebel Server setup checklist 170
- Siebel workflow information, supplying 175
- XML documents, creating workflow for 173

AIX

- configuring with less memory 23

AMI (MQSeries Application Messaging Interface)

- See EAI MQSeries Server Transport

ANSI X12 format

- 143

Application Integration Components

Inbound Transport

- See AIC Inbound Transport

authentication with HTTP outbound

- 99

B

batch loading, about

- 19

BEA, about deploying in J2EE Application

- Server 200

BizTalk Server

- about 143
- Active Server Pages, copying to BizTalk Server 182
- BizTalk configuration objects, setting up for HTTP inbound 182
- channel, setting up 175
- configuration objects, setting up 164
- configuration objects, Siebel HTTP Inbound Transport 158
- configuration objects, Siebel HTTP Outbound Transport 178
- configuration objects, Siebel MSMQ Outbound Transport 155
- document specifications, creating 153
- exposing integration objects, checklist 151
- features 143
- formats 143
- HTTP-based AIC DDL, copying from Siebel Server 182
- HTTP-Based AIC, setting up to run on 181
- IInterchange Submit parameter 163
- information sources (table) 144
- installation and document specification checklist 149
- integration objects, about exposing 150
- interface architecture 145
- Messaging Manager, configuring 172
- receiving messages from 173
- required components for HTTP 177
- schema, about generating 146
- schemas importing 153
- Server DLL, copying to Siebel Server 170
- Siebel data type mapping 153

- Siebel HTTP Outbound Transport,
 - checklist 178
- SubmitSync parameter 163
- Type Library files, copying to Siebel
 - Server 170
- UNIX considerations 149
- BizTalk Server adapter**
 - installation and document specification
 - checklist 149
 - preparing to use 148
 - servers and clients, installing and
 - configuring 149
 - Siebel integration objects 150
- BizTalk Server, connecting to**
 - EAI MSMQ Transport, checklist for using
 - inbound messaging 158
 - EAI MSMQ Transport, checklist for using
 - outbound messaging 154
 - Siebel AIC Inbound Transport, setting
 - up 167
 - Siebel COM Outbound Transport, setting
 - up 162
 - Siebel COM Outbound Transport, using to
 - send messages 163
 - Siebel HTTP Outbound Transport,
 - checklist 177
- business objects, SQL commands for
 querying** 129
- business services**
 - EAI HTTP Transport, setting up for 90
 - parameter templates, using 95
- C**
- .cfg file entries, future support of** 81
- character conversion argument, availability
 of** 11
- CharacterSetConversion parameter, about
 setting Receive attribute** 28
- CharSetConversion parameter, about** 15
- clone integration object component** 194
- clone method, about** 193
- COM Outbound Transport**
 - BizTalk communication link, establishing 162
 - parameters (table) 163
 - sending messages, checklist 163
- COM+ package**
 - AIC, registering 171
 - BizTalk Server, registering on 168
 - remote communication, creating 171
- Command object, about and interfaces
 supported** 129
- CommandType property** 124
- connect string**
 - local client mode, Siebel OLE DB
 - Provider 118
 - Siebel OLE DB Provider, server connected
 - mode 117
 - TCP/IP example 85
- Connection property** 124
- connection subsystem, creating** 17
- ConnectionSubsystem parameter, about
 using** 13
- consumers**
 - See OLE DB Consumers
- ConverterService parameter, about** 15
- D**
- data**
 - Active Server Pages, using to view Siebel
 - data 134
 - Siebel data, using Visual Basic to view 131
- data handling parameters (table)** 15
- data handling subsystem, creating** 18
- data mapping, OLE DB consumers** 113
- data transfer, about high volume** 19
- database table, troubleshooting, could not
 be opened** 138
- DataHandlingSubsystem, about using** 13
- DataSource objects**
 - DataSource properties (table) 128
 - interfaces supported 128
 - Session object, relation to 129
- DBPROP_AUTH_PASSWORD property** 129
- DBPROP_AUTH_USERID OLE DB
 property** 129
- DBPROP_INIT_DATASOURCE property** 129
- DBPROP_INIT_PROMPT property** 129
- dead letter queue, about** 33
- Default Tasks parameter, configuring** 17
- dispatch error handling for EAI MQSeries
 Server Transport** 25
- Dispatch method, about** 13
- Dispatch parameter usage, table of** 15
- Dispatch service, about** 13
- DispatchMethod parameter, about** 16
- DispatchRuleSet parameter, about** 16
- DispatchService parameter, about** 16
- DispatchWorkflowProcess parameter,
 about** 16
- distribution list, role in BizTalk
 transport** 147
- DLL properties, viewing** 120
- DLLs, external**
 - DLL, making 108
 - EAI DLL Transport, using to call a
 - function 108

methods, supported 107
 parameters, about 107

document specifications

checklist for creating 149
 importing schema 153

Document Type Definition (DTD) format

importing into BizTalk, overview 146
 new schema, generating 152

E

EAI BTS COM Transport, about using 167

EAI Connectors, BizTalk Server integration 149

EAI DLL Transport

DLL, making 108
 external DLL, calling a function 108
 methods, supported 107
 parameters, about 107

EAI File Transport

about 109
 EAI XML Read from File, compared to 109
 file names, generating 110
 parameters (table) 111
 supported modes (table) 109
 workflow, creating 110

EAI HTTP Transport

about and methods 79
 business service, selecting 80
 external system, using messages returned from 100
 HTTP outbound, basic authentication with 99
 HTTP request in session mode 98
 inbound messaging, about 84
 inbound messaging, specifying HTTP parameters 84
 network (HTTP protocol), login HTTP request example 1 86
 network (HTTP protocol), login HTTP request example 2 87
 network (HTTP protocol), logoff HTTP request 88
 POST and GET methods, about and restrictions 80
 send and receive messages, creating workflow process 96
 Send and SendReceive arguments (table) 101
 sending messages, process overview 81
 sessionless mode, secure request in 99
 sessionless mode, using in 88
 Siebel Server, setting configuration parameters 85
 system requirements 80

EAI HTTP Transport, inbound

business service, setting up 90
 error handling 93
 usage checklist 90
 workflow process, creating 91

EAI HTTP Transport, outbound

HTTP parameter templates 95
 HTTP parameters as run-time properties 95
 parameters, about specifying 93
 server-side parameters, specifying 94

EAI MQSeries AMI Transport

configuring 28
 parameters (table) 28
 ReceiveDispatchSend method, about using with AMI 29

EAI MQSeries Server Transport

See also inbound messages
 AIX environment, configuring with less memory 23
 configuring 22
 dispatch error handling 25
 EAI MQSeries AMI Transport, configuring 28
 MQMD headers, exposing 25
 MQSeries Server Receiver, about using 25
 parameters (table) 24
 re-entrance process, about using 29
 using about 21

EAI MSMQ Outbound, setting up BizTalk configuration objects 155

EAI MSMQ Transport

See also BizTalk Server, connecting to; MSMQ Transport; MSMQ Transport Server, configuring
 about 33
 inbound messages, receiving 45
 integration objects, defining 37
 messages, receiving, dispatching, and sending 49
 MSMQ, receiving messages from 45
 MSMQ, sending literal to and receiving a response 41
 outbound messages, sending with 38
 parameters (table) 36
 prerequisites, about 36
 receiving and executing messages 47
 Siebel application, sending messages from 38

EAI XML Read from File, compared to EAI File Transport 109

EDIFACT format 143

Electronic Data Exchange (EDI) format 143

Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) format 143

equals integration object component 194
equals method, about 193

error

dispatch service error, receiver shuts down
 (troubleshooting) 13
 workflow, capturing error in workflow 30

error handling

EAI HTTP Transport 93
 EAI MQSeries Server Transport, for 25

Excel

query properties (table) 124
 Siebel application data, viewing in 124
 Siebel OLE DB Provider, about accessing 116

external OLE DB-enabled applications

See Microsoft OLE DB

external system

messages, using returned from 100
 Siebel application, sending messages
 from 34
 Siebel application, sending to 34

F

file names, generating 110

foreign keys, using 119

fromPropertySet method

integration object 193
 integration object component 195

G

GET method

about and restrictions (table) 80
 Session Cookie mode login example 86

**getfChildIntOb integration object
 component** 195

**getfFieldName integration object
 component** 195

getfIntObjectFormat method, about 193

getfIntObjectName method, about 193

getfIntObjInst method, about 193

getfMessageId method, about 193

getfMessageType method, about 193

**getfOutputIntObjectName method,
 about** 194

H

**HTTP inbound, setting up for BizTalk
 configuration objects** 182

HTTP Session Cookie mode

login examples 86

I

IAccessor interface

Command object 129

Rowset object 130

IBM MQSeries

See EAI MQSeries Server Transport

**IBM MQSeries client, about connecting
 to** 22

**IBM WebSphere, about deploying in J2EE
 Application Server** 200

IColumnsInfo interface

Command object 129

Rowset object 130

ICommand interface 129

ICommandProperties interface 129

ICommandText interface 129

ICommandWithParameters interface 129

IConvertType interface

Command object 129

Rowset object 130

IDBCreateCommand interface 129

IDBCreateSession interface 128

IDBInitialize OLE DB interface 128

IDBProperties OLE DB interface 128

IDBSchemaRowset interface 129

IGetDataSource interface 129

**IgnoreCharSetConvErrors parameter,
 about** 16

IIS

See Microsoft Internet Information Server
 (IIS)

**Inbound EAI HTTP Transport business
 service**

See EAI HTTP Transport, inbound

inbound messages

See *also* EAI MQSeries Server Transport
 about 84

EAI Transport, receiving 45

HTTP parameters, specifying 84

Message Id tracking 30

Siebel Server, setting configuration
 parameters 85

inbound methods

about 13

dispatch service error, receiver shuts down
 (troubleshooting) 13

list of 13

InputBizdoc process property 174

integration objects

about exposing 150

checklist for creating 149

EAI MSMQ Transport, defining for 37

exposing, checklist for 151

Java code directory structure 191

Java code, generating 188

new schema, generating 152

- OLE DB differences in 123
- OLE DB rowsets 120
- OLE DB rowsets, relationship to 114
- Sample Account integration object, generated
 - JavaBean for 192
- Siebel OLE DB Rowset object, creating 121
- Siebel Server workflow, configuring for
 - for trading partners 151

Integration_ObjectIO method, about 194

IntObjCompIC

- integration object component, default Java
 - methods 195
- integration object component, default method
 - (SiebelPropertySet) 195

IOpenRowset interface 129

IPersist OLE DB interface 128

IRowset interface 130

IRowsetInfo interface 130

ISessionProperties interface 129

ISupportErrorInfo interface

- Command object 129

- Rowset object 130

ISupportErrorInfo OLE DB interface 128

J

J2EE Connector Architecture, about support for Siebel Resource Adapter 200

Java code, integration objects

- directory structure 191

- generating 188

L

language, support of multilingual operation 118

local client mode, about connecting to Siebel OLE DB Provider 118

login

- HTTP request example 1 86

- HTTP request example 2 87

logoff HTTP request 88

M

Message Id tracking for inbound messages 30

Message queuing API (MQI)

- See EAI MQSeries Server Transport

messages

- EAI MSMQ Transport, receiving, dispatching, and sending messages 49

- external system, sending messages to 34

- inbound messages, receiving with EAI MSMQ Transport 45

- Message Id tracking for inbound

- messages 30

- outbound messages, sending with EAI MSMQ Transport 38

- sending and receiving messages, methods for 34

- Siebel application to an external system, sending 34

Microsoft Access

- ad hoc reports, about using for 125

- Siebel application data, viewing in 125

- Siebel OLE DB Provider, about accessing 116

Microsoft Access XP, about support of 121

Microsoft ADO version 2.1 116

Microsoft BizTalk Server 2000

- See BizTalk Server

Microsoft Excel

- query properties (table) 124

- Siebel application data, viewing in 124

- Siebel OLE DB Provider, about accessing 116

Microsoft Excel 2000, OLE DB connection to 117

Microsoft Internet Information Server (IIS)

- BizTalk Server 177

- OLE DB Consumer 116

Microsoft Message Queuing Transport

- See MSMQ Transport

Microsoft OLE DB

- See *also* Siebel OLE DB Provider

- about 113

- supported foundation version 116

Microsoft SQL Server, distributed queries 126

Microsoft Visual Basic, using to view Siebel data 131

Microsoft Windows Client, about using to query Siebel OLE DB rowsets 114

Microsoft Windows Event Viewer, using 119

Microsoft Windows Server, about using to distribute Siebel OLE DB rowsets 114

model queue, about sending to 24

MQI (Message queuing API)

- See EAI MQSeries Server Transport

MQMD headers

- about exposing 25

- message headers (table) 26

MQSeries AMI Receiver server component

- connection subsystem, creating 17

- data handling subsystem, creating 18

- Default Tasks parameter, configuring 17

- Receiver Server task, starting 18

- workflow, specifying names subsystems in 18

MQSeries Application Messaging Interface

(AMI)

See EAI MQSeries Server Transport configuring 28

MQSeries Server Receiver

using, about 25
workflow process, invoking 30

MSMQ Client

configuring 35

MSMQ transport

See *also* EAI MSMQ transport
about 33
EAI MSMQ Transport, about 33
sending and receiving messages, methods for 34

MSMQ Transport Server, configuring

See *also* EAI MSMQ Transport
MSMQ Primary Controller, about configuring 35
Regional Enterprise Server and MSMQ Client, configuring 35

MSSQL, unable to query SODP using 140**multivalue fields (MVs), about creating** 120**multilingual operation, support of** 118**MVs (multivalue fields), about creating****N****named subsystems**

See *also* named subsystems. configuring data handling parameters (table) 15
Dispatch parameter usage (table) 15
EAI Transport parameters 15
object interfaces, about and support of 19
parameter specification precedence rules 14
parameters, about specifying in business service 14

named subsystems, configuring

See *also* named systems
connection subsystem, creating 17
data handling subsystem, creating 18
Default Tasks parameter, configuring 17
object interfaces, about and support of 19
Receiver Server task, starting 18
workflow, specifying named subsystems in 18

network

HTTP protocol, using in sessionless mode 88
login HTTP request example 1 86
login HTTP request example 2 87
logout HTTP request 88

O**object interfaces**

about and support of 19

OLE DB

See Microsoft OLE DB

OLE DB Consumers

applications supported 120
defined 113
multivalue fields, use of 120
OLE DB objects supported, list of 128
retrieving data, about 120
writing, about 128, 134

OLE DB objects

Siebel OLE DB Command object, about and interfaces supported 129
Siebel OLE DB Command object, syntax 130
Siebel OLE DB DataSource object, about and interfaces supported 128
Siebel OLE DB Rowset object, about and interfaces supported 130
Siebel OLE DB Session object, about and interfaces supported 129

OLE DB Provider

configuration testing 116
IDBCreateCommand interface 129
IDBCreateSession 128
IDBInitialize 128
IDBProperties interface 128
IPersist interface 128
ISupportErrorInfo interface 128
software architecture 114

OLE DB rowsets

See rowsets

Outbound EAI HTTP Transport business service

See EAI HTTP Transport, outbound

outbound messages, sending with EAI MSMQ Transport 38**outbound methods, about and list of** 12**P****parameter templates, about parameters** 95

COM Outbound Transport (table) 163
data handling parameters (table) 15
Dispatch parameter usage (table) 15
DLL Transport parameters (table) 107
EAI File Transport (table) 111
EAI MQSeries AMI Transport (table) 28
EAI MQSeries Server Transport parameters (table) 24
EAI MSMQ Transport parameters (table) 36
EAI Transport parameters, about 15
specification precedence rules 14
specifying as run-time properties 95

policies, about 28

POST method

about and restrictions (table) 80
Session Cookie Mode login example 86

primary keys, using 119

Q

queries

Command object, creating 129
distributed 126
Microsoft Excel query properties (table) 124
queries against one virtual table 130
query file, Siebel OLE DB Provider 124
Siebel OLE DB Provider, using 125
troubleshooting, query did not run 138

QueryText property 124

QueryType property 124

R

Receive and Execute inbound method, about 13

Receive attribute, about setting as Convert 28

Receive inbound method, about 13

Receive, Execute, Send inbound method, about 13

ReceiveDispatch inbound method, about 13

ReceiveDispatch method, about dispatch error handling 25

ReceiveDispatchSend inbound method, about 13

ReceiveDispatchSend method

AMI, using with 29
dispatch error handling, about 25

Receiver Server task, starting 18

receiving messages

external system to a Siebel application 34
external system, from a Siebel application 34
methods for 34

Regional Enterprise Server, configuring 35

Resource Adapter

See Siebel Resource Adapter

RollbackOnDispatchError parameter, about 17

Rowset object, about and interfaces supported 130

rowsets

See *also* Siebel OLE DB Rowset
integration objects, relation to 114
naming convention 122
resequencing field names 122
retrieving data, about 120

rowset object, creating 121

rowset object, modifying 123

Rowset object, support of 130

Session objects, relation to 129

visibility level 122

.rqy file, about using to store query 124

run-time properties, parameters as 95

S

Sample Account integration object, generated JavaBean for 192

SAP IDOC, BizTalk Server scenario 148

Schema Generator Wizard, about using schemas 150

BizTalk, importing into 153

formats 146

generating 152

scripts

Active Server Pages, using to view Siebel data 134

OLE DB consumer, writing 128

OLE DB objects supported, list of 128

Siebel OLE DB Provider, about using with 127

Visual Basic, using to view Siebel data 131

Send and Receive outbound method, about 13

Send method

arguments (table) 101

sending messages

methods for 34

Siebel application from an external system 34

Siebel application to an external system 34

SendReceive method

arguments (table) 101

EAI HTTP Transport, about 79

EAI Transport, using with 96

HTTP outbound, basic authentication with 99

HTTP request in session mode 98

secure request in sessionless mode 99

server connected mode, Siebel OLE DB Provider 117

Server: Msg 7357, troubleshooting 139

Server: Msg 7399, troubleshooting 139

service points, about 28

Session object, about and interfaces supported 129

setFieldName integration object component 195

setIntObjectFormat method, about 194

setIntObjectName method, about 194

setMessageId method, about 194

- setfMessageType method, about** 194
- setfOutputIntObjectName method, about** 194
- Siebel 6.x, EAI MQSeries Transport, about upgrading workflows** 21
- Siebel 7.x, EAI MQSeries Transport, about upgrading workflows** 21
- Siebel ActiveX Data Controls, installing** 168
- Siebel AIC, Siebel Server Workflow configuration** 173
- Siebel application data**
 - Microsoft Access, viewing in 125
 - Microsoft Excel, viewing in 124
 - SQL Analyzer, viewing in 126
- Siebel BizTalk adapter**
 - business scenarios 147
 - COM Outbound Siebel workflow, creating 165
 - data formats supported 146
 - inbound and outbound interfacing processes 146
 - interface file installation 149
 - new schema, generating 152
 - protocols supported 144
 - transport protocols, supported 146
 - XDR data type mappings 153
- Siebel Client, creating COM Outbound workflow** 165
- Siebel COM Outbound Transport**
 - See also* BizTalk Server, connecting to
- Siebel database, writing XML document to** 174
- Siebel EAI Connectors installation option, caution** 118
- Siebel eAI Transports**
 - See* transports; Transport methods
- Siebel HTTP Inbound Transport**
 - about 181
 - BizTalk configuration objects, setting up 158
 - BizTalk, receiving messages from (checklist) 182
 - Siebel Server, about configuring 181
 - Siebel workflow, creating 159
- Siebel HTTP Outbound Transport**
 - ASP files, copying to directory 177
 - BizTalk configuration objects, setting up 178
 - BizTalk, sending messages, checklist 178
 - checklist 177
 - Siebel workflow, creating 156
 - Siebel workflow, creating for 179
 - virtual directory, creating 177
- Siebel Integration Objects**
 - See* integration objects; Java code, integration objects
- Siebel JAR files, about and list of** 200
- Siebel JavaBean**
 - See also* Siebel Resource Adapter
- Siebel JavaBean Wizard**
 - folders and files, about created for integration object 191
 - integration objects, generating for 188
 - Sample Account integration object, generated JavaBean for 192
- Siebel OLE DB Command object**
 - about and interfaces supported 129
 - syntax 130
- Siebel OLE DB DataSource object**
 - about and interfaces supported 128
 - properties (table) 128
- Siebel OLE DB Provider**
 - See also* Microsoft OLE DB
 - about 113
 - Active Server Pages, about accessing from 116
 - architecture (diagram) 115
 - business scenarios 114
 - Command object 129
 - configuration testing 116
 - connection properties (table) 116
 - data mapping 113
 - default interface, applications 116
 - DLL properties, viewing 120
 - events, viewing 119
 - Excel or Access, about accessing from 116
 - foreign keys, using 119
 - installation, about 115
 - installation, location of 115
 - local client mode, about 118
 - local client mode, data source syntax (table) 118
 - MS SQL Server and distributed queries 126
 - multilingual operation, support of 118
 - multivalued fields, about creating 120
 - OLE DB consumer, writing 128, 134
 - OLE DB objects supported, list of 128
 - primary keys, using 119
 - read-only status 121
 - .rqy file 124
 - scripting languages supported 127
 - server connected mode, about 117
 - Session object 129
 - Siebel data, retrieving using VB and ASP 131
 - Siebel EAI Connectors installation option, caution 118
 - Siebel Mobile Client 116
 - Siebel Server 116
 - software architecture 114
 - SQL Server, about accessing from 116

- troubleshooting 138
 - visibility levels 122
 - Windows client 114
 - Siebel OLE DB Rowset**
 - See *also* rowsets
 - about 114
 - integration objects, differences in 123
 - rowset object, creating 121
 - rowset object, modifying 123
 - wizard, about 120
 - Siebel OLE DB Rowset object, about and interfaces supported** 130
 - Siebel OLE DB Session object, about and interfaces supported** 129
 - Siebel Resource Adapter**
 - about 200
 - using, about 200
 - Siebel Server**
 - AIC Inbound Transport, about 167
 - AIC Inbound Transport, setup checklist 170
 - BizTalk Server DLL, copying 170
 - COM+ remote communication, creating 171
 - configuration parameters, setting 85
 - HTTP-based AIC DDL, copying to BizTalk Server 182
 - integration component workflow, configuring for 173
 - Siebel HTTP Inbound Transport, about configuring 181
 - Type Library files, copying 170
 - Siebel Tools**
 - BizTalk Server, exposing integration objects to 145
 - BizTalk Server, integration object for 150
 - DTD schemas, role in generating 152
 - EAI MSMQ transport, about customizing 33
 - outbound HTTP Transport messages 93
 - role in specifying business service user properties 93
 - schema, about generating 146
 - Siebel OLE DB rowset object, creating 121
 - Siebel OLE DB rowset object, modifying 123
 - Siebel OLE DB rowset, role in defining 114
 - XDR schemas, role in generating 152
 - Siebel Web client, server connected mode, Siebel OLE DB Provider** 117
 - Siebel Web Engine (SWE) connectivity, checking** 84
 - Siebel Workflow Process Manager**
 - See EAI MSMQ Transport
 - Siebel workflows**
 - COM Outbound, creating 165
 - creating for Siebel HTTP Inbound Transport, creating for 159
 - creating for Siebel HTTP Outbound Transport 156
 - Siebel HTTP Outbound Transport, creating for 179
 - Siebel Server, configuring for integration objects 173
 - workflow information, supplying for AIC Inbound Transport 175
 - siebel.udl file, about using** 116
 - SiebelMessage process property** 174
 - SiebelTransactions parameter, about** 17
 - SODP, unable to query using MSSQL Query Analyzer** 140
 - SQL Analyzer, viewing Siebel Application data in** 126
 - SQL commands, supported** 129
 - SQL Query Analyzer message 7399, troubleshooting** 139
 - SQL Server, about accessing Siebel OLE DB Provider** 116
- T**
- templates, about parameter templates** 95
 - test connection**
 - button, about using 117
 - troubleshooting 139
 - testing**
 - See Siebel OLE DB Provider
 - toPropertySet**
 - integration object 194
 - integration object component 195
 - trading partners**
 - BizTalk Server scenario 147
 - integration objects for 151
 - Transport methods**
 - See *also* named subsystems; named subsystems, configuring; transports
 - about 12
 - data handling parameters (table) 15
 - inbound methods 13
 - outbound methods 12
 - transports**
 - See *also* named subsystems; named subsystems, configuring; Transport methods
 - about and list of 12
 - communication connectivity, about 11
 - Dispatch parameter usage (table) 15
 - parameters, about 15
 - role of 11
 - Transport methods 12
 - troubleshooting**
 - database table could not be opened 138

initialization failed 138
 query did not run 138
 Server: Msg 7357 139
 SiebelOLEDB.Provider.1 supplied inconsistent
 metadata for a column message 140
 SODP, unable to query using MSSQL Query
 Analyzer 140
 SQL Query Analyzer error message 139
 test connection failure 139

U

UNIX, BizTalk Server considerations 149
URL, checking connectivity to SWE 84

V

Version property 124
virtual tables, querying 130
Visual Basic, using to view Siebel data 131

W

**WebDAV BizTalk Repository, about
 using** 151
**WebLogic, about deploying in J2EE
 Application Server** 200
**WebSphere, about deploying in J2EE
 Application Server** 200
**Windows Client, about using to query Siebel
 OLE DB rowsets** 114

Windows Event Viewer, using 119
**Windows Server, about using to distribute
 Siebel OLE DB rowsets** 114

workflows

EAI File Transport, using to create 110
 EAI HTTP Transport, creating to receive
 messages 91
 EAI HTTP Transport, sending messages 81
 EAI HTTP Transport, setting up for 96
 error, capturing in workflow 30
 HTTP outbound, basic authentication with 99
 HTTP request in session mode 98
 messages, using returned from external
 system. 100
 MQSeries Server Receiver, invoking a process
 using 30
 named subsystems, specifying 18
 sessionless mode, secure request in 99

X**XDR format**

BizTalk, importing into overview 146
 data type mappings 153
 new schema, generating 152

XML

BizTalk Server, role in 144
 workflow for inbound documents 173