Transports and Interfaces: Siebel *e*Business Application Integration Volume III

Version 7.5.3

July 2003

12-FRWMT3

Contents

Introduction

This guide explains the details of the transports and interfaces technology of Siebel eBusiness Application Integration (Siebel eAI), including MQSeries, MSMQ, OLE DB Provider, BizTalk interface, and Java Data Beans.

Although job titles and duties at your company may differ from those presented in the following table, the audience for this guide consists primarily of employees in these categories:

Business Analysts	Persons responsible for analyzing application integration challenges and planning integration solutions at an enterprise.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel Integration Developers	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution at an enterprise for Siebel applications.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.
System Integrators	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution at an enterprise for specific applications and or to develop custom solutions.

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

Product Modules and Options

This *Siebel Bookshelf* 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 Bookshelf. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

How This Guide Is Organized

This book is organized in a way that presents information on discrete components of the Siebel eAI transports and interfaces—such as MQSeries, BizTalk, J2EE, and so on—as individual chapters. Additional information, as applicable, can be found in the appendixes.

This book is Volume 3 of a five-volume set. The full set includes:

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

Additional Resources

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

- *Application Services Interface Reference* serves as a reference if your integration involves Application Services Interfaces.
- *Siebel Tools Online Help* serves as a reference if you plan on using COM, CORBA, or the ActiveX Plug-ins to accomplish integration and also as a reference for Siebel business objects and components.
- *Siebel Business Process Designer Administration Guide*.
- *Siebel Enterprise Integration Manager Administration Guide* serves as a reference if you perform bulk loading or unloading of data.
- The Connector books provide specifics on each of the associated connectors.

Revision History

Transports and Interfaces: Siebel eBusiness Application Integration Volume III

Version 7.5.3

Table 1. Changes Made in Version 7.5.3

Topic	Revision
"Exposing MQMD Headers as Properties."	New for 7.5.3: Added a section on how to expose MQMD Headers as properties in a property set.
"About Transport Headers and HTTP Response Headers."	New for 7.5.3: Added a section on how transport headers and HTTP response headers work with HTTP Transport (outbound) to form a cookie handling system.

Version 7.5, Rev. A

Table 2. Changes Made in Version 7.5, Rev. A

Topic	Revision
"EAI MQSeries Transport Named Subsystems."	Updated the examples under the EAI MQSeries Transport named subsystems section.
"Message Id Tracking for an Inbound Message."	Added a new section on Message Id Tracking for an Inbound Message.
"Invoking a Workflow Process Using MQSeries Server Receiver."	Provided more examples for invoking a workflow process using MQSeries server receiver section.
Chapter 3, "EAI MSMQ Transport."	Updated the examples.
"About EAI HTTP Transport."	Updated the description for Send and SendReceive methods.
"Sending and Receiving Messages with the EAI HTTP Transport."	Added a new example on sending and receiving messages using EAI HTTP Transport.

Table 2. Changes Made in Version 7.5, Rev. A

Topic	Revision
"Accessing an URL Protected by Basic Authentication."	Added a section on authentication.
"Sending and Receiving Messages through HTTP."	Expanded the section on using Messages returned from an external system.
"Creating a DLL to Call a Function in an External DLL."	Added a section on creating a DLL to call a function in an external DLL.
"Troubleshooting OLE DB."	Enhanced the Troubleshooting section.
Chapter 7, "Interfacing with Microsoft BizTalk Server."	Updated the examples.
Chapter 8, "Integrating with J2EE Application Server."	Updated the content throughout chapter.

EAI Transports and Interfaces Overview 1

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

- **EAI Transports.** For information, see "EAI Transports" on page 15.
- **Object Interfaces.** For information, see "Object Interfaces" on page 26.
- **Database Level Interface and Batch Loading.** For information, see "Database Level Interfacing" on page 26.

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 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 2, “EAI MQSeries Transports.”
- EAI MSMQ Transport. For information on this transport, see Chapter 3, “EAI MSMQ Transport.”
- EAI HTTP Transport. For information on this transport, see Chapter 4, “EAI HTTP Transport.”

- EAI DLL Transport and EAI File Transport. For information on these transports, see Chapter 5, "EAI DLL and EAI File Transports."

NOTE: EAI MQSeries Server/AMI Transport, EAI MSMQ Transport, and EAI File Transport business services are not re-entrant. For more information on transport re-entrance, see "EAI MQSeries Transport Re-Entrance" on page 38.

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 4 on page 22, as well as transport-specific parameters that are discussed in the respective chapter for each transport.

Outbound Methods

Available outbound methods depend on the transport business service in use, such as EAI MQSeries AMI Transport or EAI MSMQ Transport. The business service sends messages from 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 (with the exception of the EAI HTTP Transport which expects a correct HTTP response) request method, 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

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

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

Receive	Receives an inbound request message and returns it to the caller of the transport.
Receive and Execute (ReceiveDispatch)	Receives an inbound request message and calls another service with the inbound message as input. This called service is known as the Dispatch Service, and the method that is called is known as the Dispatch Method.
Receive, Execute, Send (ReceiveDispatchSend)	This is a request/response method. Receives an inbound request message, calls another service with the inbound message as input, and then sends the output of the called service as a response. 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 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 Server 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 3. The data handling parameters are presented in Table 4. 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 3. 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 4. Common Data Handling Parameters for Transport Methods

Parameter Name	Description
CharSetConversion	■ 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. 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.
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.
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.

Table 4. Common Data Handling Parameters for Transport Methods

Parameter Name	Description
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.
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 application-level menu, choose View > Site Map > Server Administration view > Enterprise Configuration.

2. Click the Enterprise Component Groups tab.

   A list of receivers appears in the Enterprise Component Groups applet, for example, MQSeries AMI Receiver.

3. Select Components from the Server Administration view.
4. Click the Component Parameters tab.
5. Select the MQSeries AMI Receiver in the Server Components List applet.
6. Query for the Default Tasks parameter in the Component Parameters applet.
7. 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 application-level menu, choose View > Site Map > Server Administration view > Enterprise Configuration.
2. Click the Enterprise Profile Configuration tab.
3. In the Component Profiles List applet, create a new Named Subsystem record.
   - **a** Enter MyAMISubsys in the Named Subsystem Alias column.
   - **b** Select MQSeriesAMISubsys as the type.
   - **c** In the Enterprise Profile Configuration applet, specify the values for each of the connection parameters for the AMI receiver, such as MqLocalHostFileName.

To create a data handling subsystem

1. From the application-level menu, choose View > Site Map > Server Administration view >Enterprise Configuration.
2. Click on the Enterprise Profile Configuration tab.
3. In the Component Profiles List applet, create a new Named Subsystem record.
   - **a** Enter MyDispSubsys in the Named Subsystem Alias column.
   - **b** Select EAITransportDataHandlingSubsys as the type.

4. In the Enterprise Profile Configuration applet, specify the values for each of the data handling parameters for the AMI receiver, such as Dispatch Service.

To start the Receiver Server task

1. From the application-level menu, choose View > Site Map > Server Administration view > Servers.
2. Click the Server Components tab.
3. In the Server Components 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 application-level menu, choose View > Site Map > Business Process Administration > Workflow Processes.
2. Query and select the workflow process named MyMQOutbound.
3. Double-click the MQ Send step.
4. In the Input Arguments Applet, insert a new record to specify the Connection subsystem, and specify the value as MyAMISubsys.
5. Insert another input argument to specify the Data Handling Subsystem, and specify the value as MyDispSubsys.
6. Insert a third record to specify the Message Text.

Object Interfaces

Object Interfaces allow integration between 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 6, "Using Siebel OLE DB Provider."
- Siebel eBusiness Applications interfacing with Microsoft BizTalk. For information, see Chapter 7, "Interfacing with Microsoft BizTalk Server."
- Siebel eBusiness Applications that produce Java Data Beans to support J2EE applications. For information, see Chapter 8, "Integrating with J2EE Application Server."

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.

EAI MQSeries Transports 2

This chapter discusses the Siebel EAI MQSeries Transports.

Siebel EAI MQSeries Transports

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

Siebel EAI provides two MQSeries transports:

- EAI MQSeries Server Transport
- EAI MQSeries AMI Transport

NOTE: The Siebel business service EAI MQSeries Transport, which was provided in Siebel 6.x releases, is not available in the Siebel 7.x release. Customers of previous Siebel versions using 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 EAI MQSeries Server Transport, the named subsystem type is MqSeriesServerSubsys. For 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 created named subsystem and start a receiver.

```
create named subsystem MyMqSrvrSubsys for subsystem
MQSeriesServerSubsys with
MqPhysicalQueueName=Receiver,MqRespPhysicalQueueName=Sender,MqQu
eueManagerName=myQueueMgr

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

start task for comp MqSeriesSrvRcvr with
ReceiverConnectionSubsystem=MyMqSrvrSubsys,ReceiverDataHandlingS
ubsystem=SiebelEcho,ReceiverMethodName=ReceiveDispatchSend

create named subsystem  MqAMIConnSubsys for subsystem
MQSeriesAMISubsys with
MqPolicyName=MY.POLICY,MqReceiverServiceName=MY.RECEIVER,MqTrace
=true,MqTraceLocation=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,ReceiverDataHandling
Subsystem=SiebelEcho,ReceiverMethodName=ReceiveDispatchSend
```

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

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 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" on page 17 and "Inbound Methods" on page 18.

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, 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 4 on page 22, the EAI MQSeries Server Transport also uses the parameters shown in Table 5. 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 5. 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.

Table 5. EAI MQSeries Server Transport Specific Parameters

Argument	Display Name	Description
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.
MqQueueManagerName	Queue Manager Name	Name of the MQSeries queue manager. If this parameter is not specified, the default Queue Manager Name, as specified in the MQSeries configuration, is used.
MqRespModelQueueName	Response Model Queue Name	Name of model queue for response connection. The Response Queue Manager is the same as MqQueueManagerName.
MqRespPhysicalQueueName	Response Physical Queue Name	Name of physical queue for response connection.
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 EAI MQSeries Server Transport.

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

Table 6. MQMD Message Headers

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

Table 6. MQMD Message Headers

Field	DataType	Description	Input/Output Property
AccountingToken	MQBYTE32	Accounting token.	Output.
ApplIdentityData	MQCHAR32	Application data relating to identity.	Output.
PutApplType	MQLONG	Type of application that put the message.	Output.
PutApplName	MQCHAR28	Name of application that put the message.	Output.
PutDate	MQCHAR8	Date when message was put.	Output.
PutTime	MQCHAR8	Tine when message was put.	Output.
ApplOriginData	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 form 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.

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 and

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

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.
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.

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.

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 mqseriesserversubsys
with MQQueueManagerName=QueueMgr, MQPhysicalQueueName=LocalQueue
```

Command to start a MqSeriesSrvrRcvr

```
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.

EAI MSMQ Transport 3

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

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.

Sending Messages Methods

- Send
- Send and Receive Response

Receiving Messages Methods

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

Messages from a Siebel Application to an External System

You configure EAI MSMQ Transport using the Siebel 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 1, “EAI Transports and Interfaces Overview.”
- For more information on named subsystems, see *Siebel Server 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
- 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
- MSMQ Client
- As many MSMQ Queues as needed
- Relevant ODBC driver
- Siebel Server
- Siebel Gateway
- Siebel HTML Connected Client

NOTE: MSMQ Server can reside on either machine. This functionality is independent of the underlying database platform. You can use any of the supported database platforms, including IBM UDB, IBM 390, Oracle, and Microsoft SQL Server.

NT Server Setup

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

NT Server Setup Prerequisites

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

To set up the MSMQ Primary Enterprise Controller (PEC)

1. Choose Start > Programs > Windows NT 4.0 Option Pack Setup.
2. Choose Add/Remove Components.
3. Click Microsoft Message Queue, uncheck all other options, and click Next.
4. Click Primary Enterprise Controller to install MSMQ Server.
   a. Specify a name for the Enterprise to be created, such as Siebel.

  - **b** Specify the site to be created, such as Sales.
  - **c** Leave all other default MSMQ parameters as is.

5. Click Connected Network as the machine on which MSMQ PEC is installed.

To set up the MSMQ Client

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

Configuring EAI MSMQ Transport for Various Send and Receive Scenarios

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

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.
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

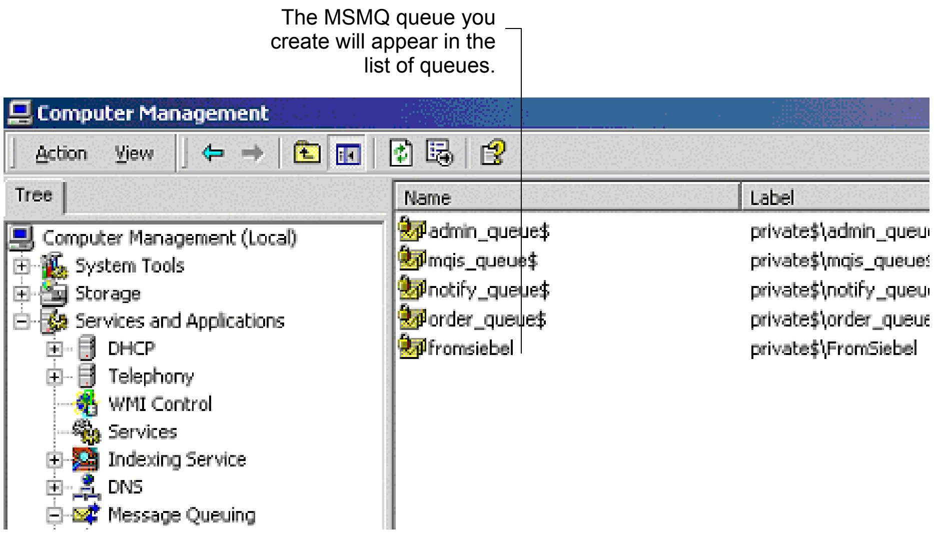
The following procedure describes how to set up your system to send a message to an external system using EAI MSMQ Transport.

To send messages from a Siebel application to MSMQ

1 Access the Windows Computer Management tool.

- On Windows NT, choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer.
- On Windows 2000, choose Start > Programs > Administrative Tools > Computer Management.

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



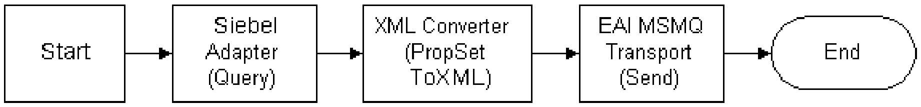
Name	Label
admin_queue$	private$\admin_queu
mqis_queue$	private$\mqis_queue
notify_queue$	private$\notify_queu
order_queue$	private$\order_queue
fromsiebel	private$\FromSiebel

3 Set the queue to be Transactional.

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

4 From the application-level menu, choose View > Site Map > Business Process Administration > Workflow Processes.

5 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.*

6 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	-
Object Id	String	In/Out	*`Row Id of an Employee record`*
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
MsmqPhysicalQueue Name	Literal	private$\FromSiebel	-	-
MsmqQueueMachine Name	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.

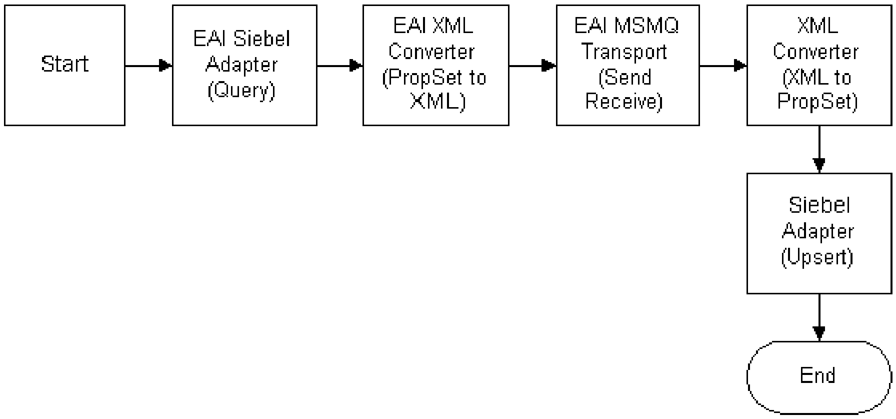
- On Windows NT, choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer.
- On Windows 2000, choose Start > Programs > Administrative Tools > Computer Management.

2 Set up an MSMQ queue to receive messages from the Siebel application, and give the queue an easy to identify name, such as fromsiebel.

3 Set up another queue to send messages to the Siebel application, and give the queue an easy-to-identify name, such as tosiebel.

4 From the application-level menu, choose View > Site Map > Business Process Administration > 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:


Start
EAI Siebel Adapter (Query)
EAI XML Converter (PropSet to XML)
EAI MSMQ Transport (Send Receive)
XML Converter (XML to PropSet)
Siebel Adapter (Upsert)
End

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.	-	-
MsmqRespQueueMachine Name	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.
   - On Windows NT, choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer.
   - On Windows 2000, choose Start > Programs > Administrative Tools > Computer Management.
2. Set up a queue to send messages to the Siebel application.
   - **a** Name the queue an easy-to-identify name, such as tosiebel.

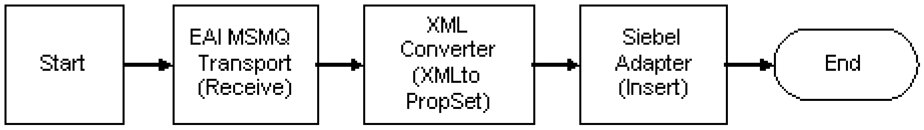
b Create a message in the queue.

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 application-level menu, choose View > Site Map > Business Process Administration > 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.

- On Windows NT, choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer.
- On Windows 2000, choose Start > Programs > Administrative Tools > Computer Management.

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

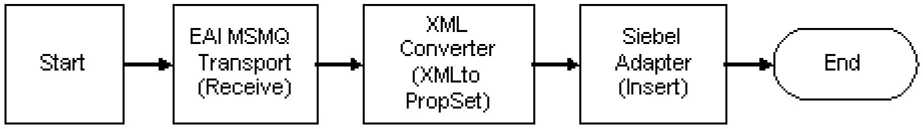
a Name the queue an easy-to-identify name, such as toSiebel.

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 application-level menu, choose View > Site Map > Business Process Administration > 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.

- On Windows NT, choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Message Queue > Explorer.
- On Windows 2000, choose Start > Programs > Administrative Tools > Computer Management.

2 Set up an MSMQ queue to receive messages from the Siebel application.

Give the queue an easy-to-identify name, such as fromSiebel.

3 Set up another queue to send messages to the Siebel application.

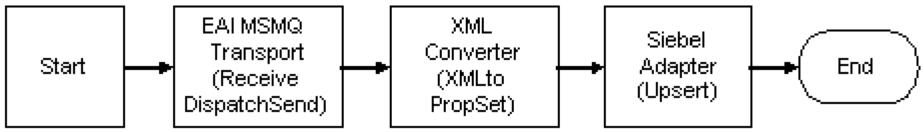
a Name the queue an easy-to-identify name, such as toSiebel.

b Create a message in 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 application-level menu, choose View > Site Map > Business Process Administration > Workflow Processes.

5 Set up a workflow process for receiving and dispatching a message from MSMQ as shown in the following figure:


Start
EAI MSMQ Transport (Receive DispatchSend)
XML Converter (XMLto PropSet)
Siebel Adapter (Upsert)
End

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
MsmqPhysical QueueName	Literal	tosiebel	-	-
MsmqQueue MachineName	Literal	Siebel2001 Machine name where the Siebel MSMQ Transport is running.	-	-
MsmqResponse MachineName	Literal	Siebel2001A Name of the machine where the queue receiving messages from Siebel application is located.	-	-
MsmqRespQueue Name	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.

EAI HTTP Transport 4

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

About 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 *CSSHTTPTransService* 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 Transports allows you to send messages across the Internet using the standard HTTP protocol. Using this transport, you can send messages to any URL. The XML document sent can then be acted upon by any Web-based application, including those written in Java, JavaScript, VBScript, or any other Web-enabled technology.

System Requirements

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

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

Selecting the Appropriate Business Service for HTTP

The business service you need to initialize to process a given XML document 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 9 identifies restrictions on these HTTP methods.

Table 9. 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 `application/xxx-form-urlencoded` 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 1, "EAI Transports and Interfaces Overview." For more information on named subsystems, see *Siebel Server Administration Guide*.

General Information on How to Send a Message

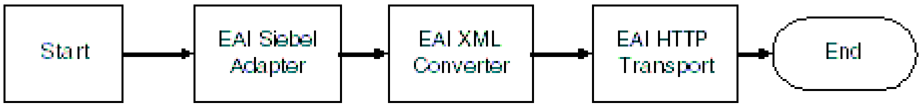
The following steps demonstrate how to send information from a Siebel application to another Web-based application using the EAI HTTP Transport.

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 the 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	Hierarchy	In/Out	-
Account XML	Binary	In/Out	-
Error Code	String	In/Out	-
Error Message	String	In/Out	-
Object Id	String	In/Out	*`Row Id of an account`*
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 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	-	Account Message	Hierarchy

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	-	Employee XML	String
Login Method	Literal	GET	-	-
Login URL Template	Literal	http://$web_address$/$login_param$	-	-
Log Off Method	Literal	GET	-	-
Log Off URL Template	Literal	http://$web_address$/$logoff_param$	-	-
Request URL Template	Literal	http://$web_address$/$request_param$	-	-
Password	Literal	mypassword	-	-

Property Name	Type	Value	Output Argument	Property Data Type
Employee 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, 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 19 in Chapter 1, "EAI Transports and Interfaces Overview."

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

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

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

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

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

Where:

transport = TCPIP or http.

encryption = none or mscrypto.

compression = none or zlib.

gateway server = the name of your Siebel Gateway server. The default port for TCP/IP is 2320, and for HTTP it is 80.

enterprise name = the name of your Siebel Enterprise Server.

object manager name = the type of Object Manager for the Siebel eBusiness application you are installing.

Siebel Server Name = the name of your Siebel Server.

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 *eSalesObjMgr*.

```
ConnectString = siebel.TCPIP.none.none://gatewayhost:2320/
siebel/eSalesObjMgr/Siebel Server name
```

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.

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.

Login Examples

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

Login HTTP Request Example 1

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

Using HTTP GET

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

Using HTTP POST

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

HTTP Body  =
SWEExtSource=source&SWEExtCmd=ExecuteLogin&UserName=username&Pas
sword=password
```

Table 10 presents each of the Login HTTP Request variables for session mode.

Table 10. Session Mode Variables

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

Example Login URL

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

Login HTTP Request Example 2

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

Using HTTP GET

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

Using HTTP POST

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

```
HTTP Body = data text
```

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 UTF-8 encoded before being URL-encoded. POST requests can send the data without URL encoding and should include the Content-Type HTTP header. The Content-Type should specify the charset of the incoming data as in Content-Type = text/xml;charset = "UTF-8".

Logoff HTTP Request

This request must include the session cookie from Login:

Using HTTP GET

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

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

Example Logoff URL

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

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

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=usernam
e&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 11 presents each of the variables for sessionless mode.

Table 11. Sessionless Mode Variables

Variable	Description
webserver	URL of the Web server that has Siebel Web Engine installed, such as www.*myserver*.com.
path	Default is eai. 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.

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:

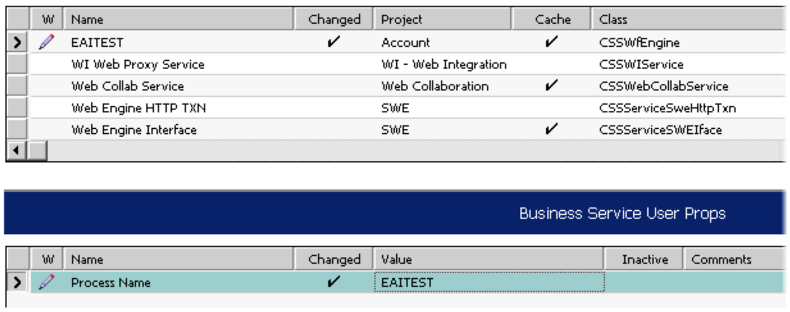
- Set up the business service for use in the workflow.
- Create the workflow.

Both steps are explained in this section.

This scenario assumes incoming XML. Your business requirements dictates 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	✓	CSSServiceSWEIface

Business Service User Props

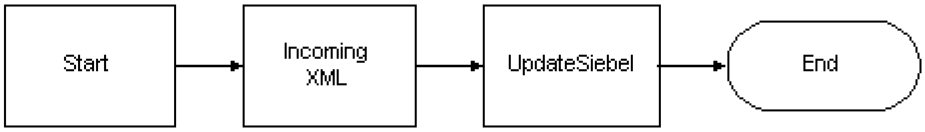
W	Name	Changed	Value	Inactive	Comments
	Process Name	✓	EAITEST		

5. Compile a new .srf file and copy it to the *SIEBSRVR_ROOT*\Object directory.
6. Restart 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 application-level menu, choose View > Site Map > Business Process Administration > 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.
< 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.

5 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

6 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.

7 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 87.
- Specifying Parameters as Subsystem Parameters on page 87.
- About Parameters as Run-Time Properties on page 88.
- About Parameters in Parameter Templates on page 88.

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 as on either the client side or the server side, depending on whether you use the client or server version.

To specify the parameters on the client side

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

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

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

4. Save the file and exit the text editor.

To specify the parameters on the server side

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

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

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

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: Any parameters specified in an input property set overrides parameters with the same name that have been specified in the .srf file or in your subsystem 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 23.

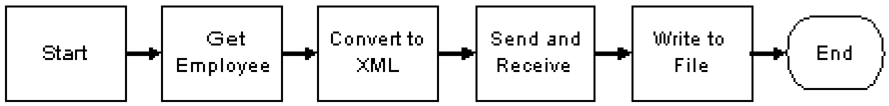
3. In your eai.cfg file, add the following line in the [HTTP Services] section:

   MyEcho = MyEchoSubsys

4. Log in to the Siebel client as an administrator connected to the server.

5. From the application-level menu, choose View > Site Map > Business Process Administration > 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&User
Name=$Username$&Password=$Password$"
```

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

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

- The following URL logs off from the server:

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

ServerPath = "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 Request Method, 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=ABCIntl&User=$Username$&pwd=$Password$"

Username = "acctuser"

Password = "123456789abcdefg"
```

Accessing an 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/.`"

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 a 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 12.

- 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 12. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
`<Value>`	*User-Defined Message Text*			Input and Output data passed as a string. This is the value stored in the Value field of the property set, either input or output. If you specify the HTTPRequestBodyTemplate, the < Value > parameter is ignored and the HTTPRequestBodyTemplate parameter is used instead.
HTTPRequestURLTemplate	Request URL Template	■	■	Template for the request URL, which is the address to which the data is sent or from which a response is requested.
HTTPRequestMethod	Request Method	■	■	HTTP method to use with the data request, 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.

Table 12. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
HTTPLoginBodyTemplate	Login Body Template	■		Specifies the HTTP request body that should be used when HTTPLoginURLMethod is POST. By putting login information into the HTTP body (as opposed to putting it into the URL) for sending, this method provides stronger security than sending the login information in the URL. 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.
HTTPLogoffMethod	Log Off Method	■		Defaults is HTTPLoginMethod. HTTP method to be used for logging off.
HTTPAccept	HTTP Accept			Default is *text/**. 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 application/xxx-form-urlencoded. 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 Mozilla/4.0. 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.

Table 12. EAI HTTP Transport Send and SendReceive Arguments

Parameter	Display Name	S	SL	Description
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.
HTTPAllowPersistentCookies	Allow Persistent Cookies			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. All session cookies persist in memory only as long as the current session. Session cookies are not written to disk. 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.
HTTPIsSecureConn	Is Secure Connection			Default is N. If set to N then the security mode defaults to whatever the URL specifies, either HTTP or HTTPS. Setting this parameter to Y enables the Secure flag for SSL communications; thus, it forces the use of secure mode. If you choose to use SSL encryption, you must establish valid certificates and SSL capabilities on both the client and server. Using the *HTTPS:* designation in an URL, by default enables the Secure flag and specifies *clear text* unencrypted communications.
HTTPNoAutoRedirect	No Auto Redirect			Default is N. This means auto-redirect is enabled. Setting this parameter to Y disables auto-redirection of messages to other URLs.

Table 12. 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.

EAI DLL and EAI File Transports 5

This chapter discusses the EAI DLL and EAI File Transports.

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 1, "EAI Transports and Interfaces Overview," the EAI DLL Transport takes the parameters presented in Table 13.

Table 13. 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 13 on page 102.

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.

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:

Sending Messages Methods

- Send
- Send and Receive Response

Receiving Messages Methods

- 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 14 on page 107.

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 1, "EAI Transports and Interfaces Overview," the EAI File Transport takes the parameters presented in Table 14. These parameters can be specified as service method arguments, subsystem parameters, or user properties.

Table 14. EAI File Transport Parameters

Display Name	Argument	Description
Append To File	AppendToFile	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	The name of the file to be received by the file transport. 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. 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.
File Text	FileText	Indicates data type of file. If the argument is set to True, the file is opened as a text file. If the argument is set to False, the file is opened as a binary file.
Response File Name	RespFileName	Name of the file containing the response when using the SendRecieve Method.
Sleep Time	FileSleepTime	The timeout interval on receive calls, in milliseconds. This specifies the maximum amount of time that the service waits for a response. Default is 20000 milliseconds.

Using Siebel OLE DB Provider 6

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

Microsoft OLE DB

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

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

Siebel OLE DB Provider

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

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

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

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

Software Architecture

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

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

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

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

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

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

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

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

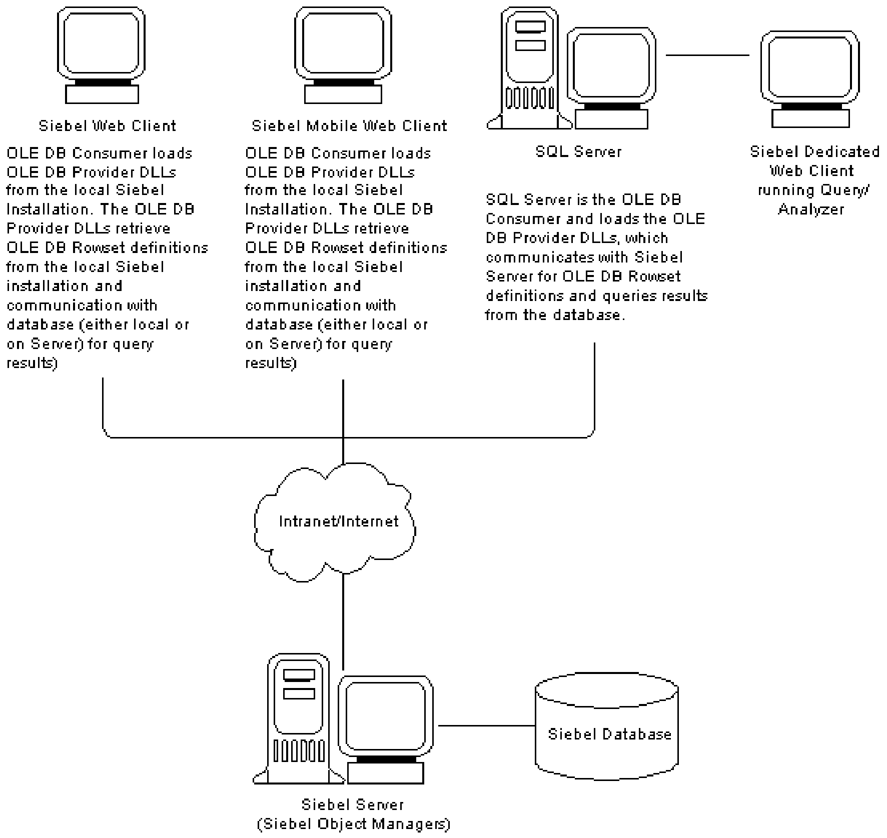

Siebel Web Client
OLE DB Consumer loads OLE DB Provider DLLs from the local Siebel Installation. The OLE DB Provider DLLs retrieve OLE DB Rowset definitions from the local Siebel installation and communication with database (either local or on Server) for query results)
Siebel Mobile Web Client
OLE DB Consumer loads OLE DB Provider DLLs from the local Siebel Installation. The OLE DB Provider DLLs retrieve OLE DB Rowset definitions from the local Siebel installation and communication with database (either local or on Server) for query results)
SQL Server
SQL Server is the OLE DB Consumer and loads the OLE DB Provider DLLs, which communicates with Siebel Server for OLE DB Rowset definitions and queries results from the database.
Siebel Dedicated Web Client running Query/ Analyzer
Intranet/Internet
Siebel Server (Siebel Object Managers)
Siebel Database

Figure 1. OLE DB Providers and Consumers of Siebel Data

About 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 *Siebel System Requirements and Supported Platforms* for further information.

This section covers the following topics:

- Siebel OLE DB Provider Configuration for Testing on page 114
- Multiple Language Considerations on page 117
- About Primary and Foreign Key Relationships on page 118
- Viewing OLE DB Provider Events on page 118
- Viewing OLE DB Information on page 119

Siebel OLE DB Provider Configuration for Testing

You configure and test Siebel OLE DB Provider using the siebel.udl file that is installed in the `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 15 presents the necessary connection properties.

Table 15. Siebel OLE DB Provider Connection Properties

Property	Description
Data Source Name	The connection string as explained in *Siebel Tools Online Help*. 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.

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.

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 16 below defines the syntax.

Table 16. Data Source Syntax

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

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

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

```
QueryType=OLEDB
Version=1

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

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

Multiple Language Considerations

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

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

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://gtwyhost/siebel/SCCObjMgr/srvr
  ```

- 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://gtwyhost/siebel/SCCObjMgr/srvr,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 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 OLE DB Provider events

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

   There should be two information events:

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

Viewing OLE DB Information

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

To view OLE DB information

1. Using Windows Explorer, right-click on each of the following OLE DB Provider DLLs.

   - bin\ssceolpr.dll
   - bin\ssceolwr.dll
   - bin\language\ssceolrs.dll

     where: *language* 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.

Siebel OLE DB and Multivalue Fields

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

For instance, if you create a rowset based on the Opportunity BC and you add the Product MVF to the rowset, when you use the rowset in MS Excel only one record will display for each opportunity. The 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 17 shows the required properties for an Excel query file.

Table 17. Excel Query Properties

Property	Value/Description
QueryType	OLEDB.
Version	1.

Table 17. Excel Query Properties

Property	Value/Description
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: ■ 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';'Pa
ssword',Siebel OLE DB Provider query text)
```

For example:

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

'SADMIN';'SADMIN',

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

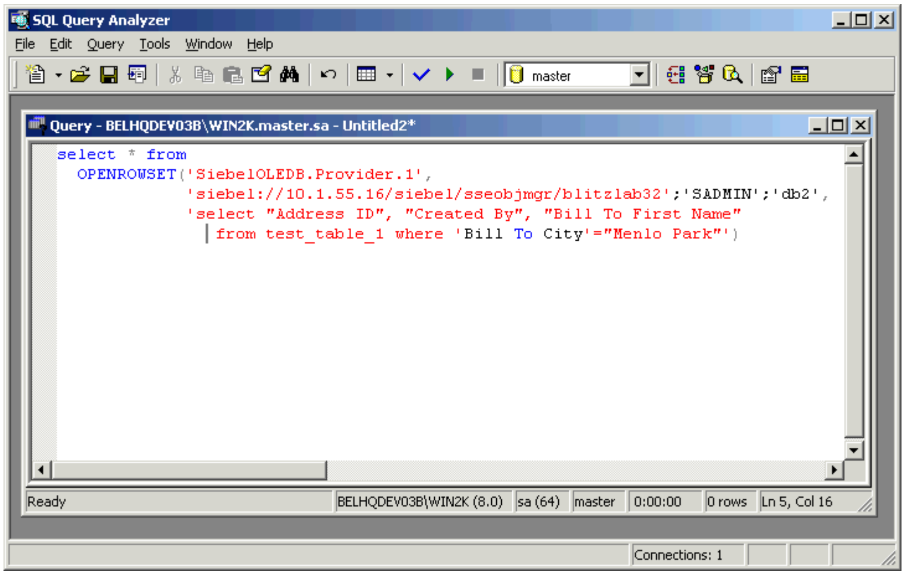
NOTE: For more information about Microsoft SQL Server distributed queries, please refer to Microsoft SQL Server 7.0 documentation.

Database administrators (DBAs) 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.



```
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"')
```

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 135

Writing an OLE DB Consumer

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

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

OLE DB Object Support in Siebel OLE DB Provider

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

- DataSource
- Session
- Command

- Rowset

Siebel OLE DB DataSource Object

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

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

- IDBCreateSession
- IDBInitialize
- IDBProperties
- IPersist
- ISupportErrorInfo

You must specify DataSource properties to successfully initialize and authorize the connection to the Siebel environment, as shown in Table 18.

Table 18. 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.

Siebel OLE DB Session Object

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

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

Siebel OLE DB Command Object

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

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

Siebel OLE DB Provider Command Syntax

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

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

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

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

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

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

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

Siebel OLE DB Rowset Object

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

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

Retrieving Siebel Data Using VB and ASP

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

To view Siebel data using Visual Basic

1 Start Microsoft Visual Basic.

2 Enter code similar to the following example (providing your actual ID, password, selection criteria, and so on):

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

Dim Fso, File



'Setup program parameters.

ProviderString = "SiebelOLEDB.Provider.1"

DataSourceString = "siebel://MyGateway/MyEnterprise/MyObjMgr/
MyServer"

UserIdString = "MyUserId"

PasswordString = "MyPassword"

OutFileString = "output.txt"



'Build the connection string.

ConnectString = "Provider=" & ProviderString & ";User Id=" &
UserIdString & ";Password=" & PasswordString & ";Data Source=" &
DataSourceString & ";"



'Ask the user if they are ready to establish a connection and
retrieve the data.
```

```
Message = "Ready to connect using" & Chr(13) & Chr(10) &
ConnectString & Chr(13) & Chr(10) & "Do you want to continue?"

Continue = MsgBox(Message, vbYesNo, "Ready to Connect")


If Continue = vbYes Then

   'Create the output file for storing the data.

   Set Fso = CreateObject("Scripting.FileSystemObject")

   Set File = Fso.OpenTextFile(OutFileString, 2, True)


   'Establish a connection.

   Set Connection = CreateObject("ADODB.Connection")

   Connection.Open ConnectString


   'Execute a query to create a record set.

   'Retrieve all accounts involved in any electrical related
   business.

   QueryString = "Select * from Account_Account_1 where 'Line of
   Business' = 'Electrical*'"

   Set RecordSet = Connection.Execute(QueryString)


'If 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")
```

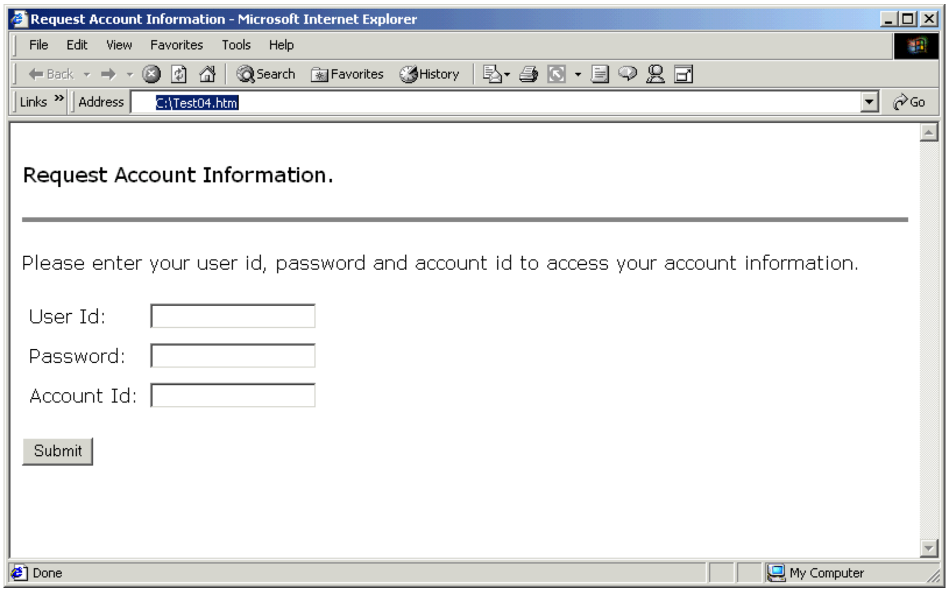
```
ConnectString = "Provider=SiebelOLEDB.Provider.1;User Id=" +
UserId + ";Password=" + Password + ";Data Source=siebel://
MyGateway/MyEnterprise/MyObjMgr/MyServer;"

Connection.Open ConnectString

If Len(AccountId) = 0 Then

  Query = "Select * from Account"

Else

 Query = "Select * from Account where Id = '" + AccountId + "'"

End If

Set RecordSet = Connection.Execute(Query)

If Not RecordSet.EOF Then

%>

  This is your current account information.

  <br><br>

  <!-- Build a table to display the data -->

  <TABLE CELLPADDING=4>

  <!-- BEGIN column header row -->

  <TR>

  <%For Each Field in RecordSet.Fields%>

  <TH><FONT SIZE=2><%Response.Write Field.Name%></FONT></TH>

  <%Next%>

  </TR>

  <%Do While Not RecordSet.EOF%>

    <TR>

    <%For Each Field in RecordSet.Fields%>
```

```
     <TD><FONT SIZE=2><%Response.Write Field.Value%></FONT></
TD>

    <%Next%>

    </TR>

<%

    RecordSet.MoveNext

  Loop

%>

  </TABLE>

<%

End If
```

' Clean up variables.

```
RecordSet.Close

Connection.Close

Set RecordSet = nothing

Set Connection = nothing

%>

</FONT>

</BODY>

</HTML>
```

The form created by the HTML page prompts the user for input, as shown in Figure 2.


Request Account Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links Address C:\Test04.htm Go

Request Account Information.

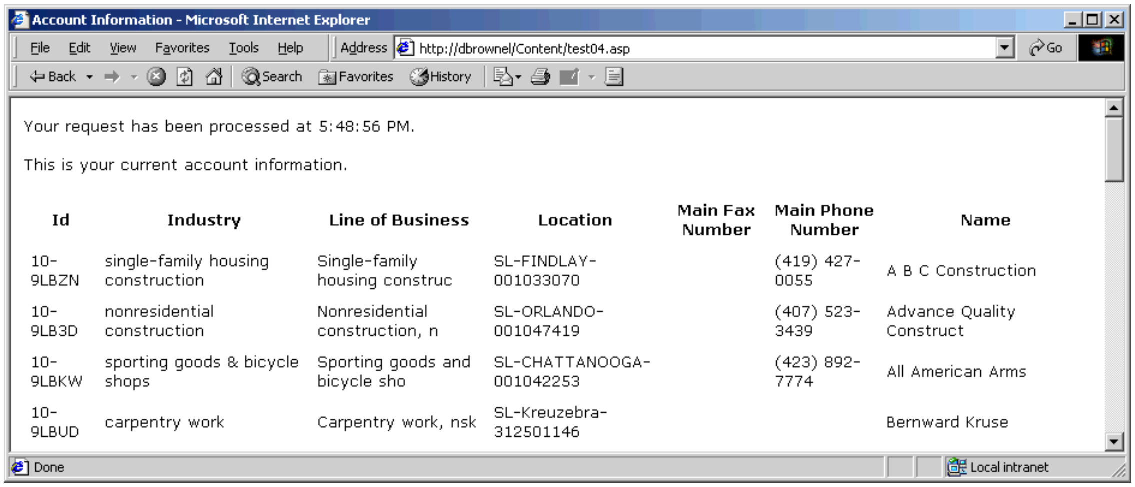
Please enter your user id, password and account id to access your account information.

User Id:	
Password:	
Account Id:	

Submit

Figure 2. Sample HTML User Input Page

The output is shown in Figure 3.


Account Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://dbrownel/Content/test04.asp
Go
Back Search Favorites History
Done
Local intranet

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

This is your current account information.

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

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 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.

Interfacing with Microsoft BizTalk Server 7

This chapter discusses the Microsoft BizTalk Server.

About Microsoft BizTalk Server

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

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

Siebel BizTalk Server Adapter

With Siebel eBusiness Applications support for Microsoft BizTalk Server, you can share Siebel data with external and internal contemporary and legacy systems by generating and exchanging XML documents through BizTalk Server, without having to create a custom solution or write custom code. This allows for interaction between business processes within a single organization, 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 *ge*t (extract) and *pu*t (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 19 shows other sources of information on associated technologies.

Table 19. Other Information Sources

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

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.

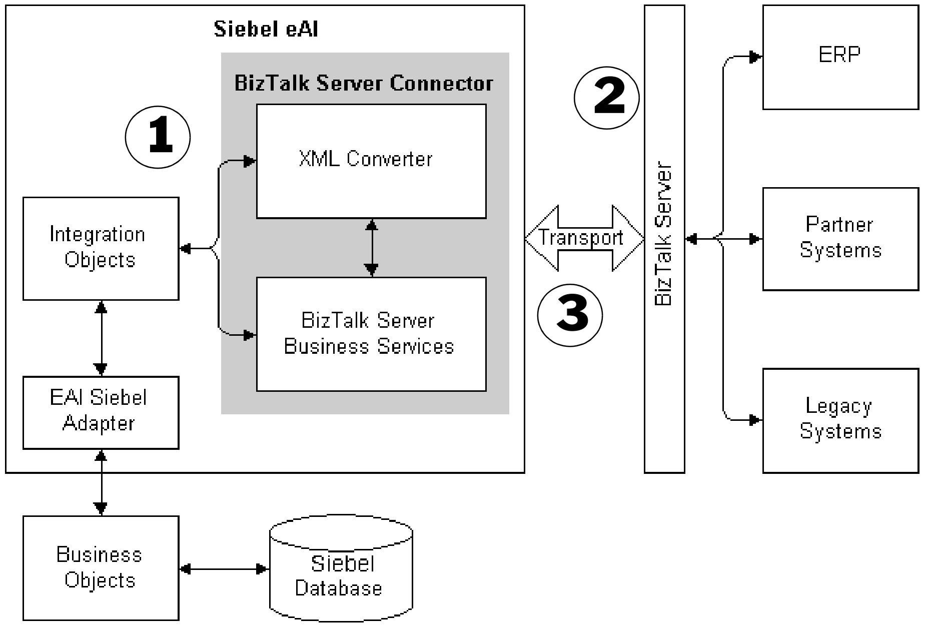


Figure 4. Siebel BizTalk Interface Architecture

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.

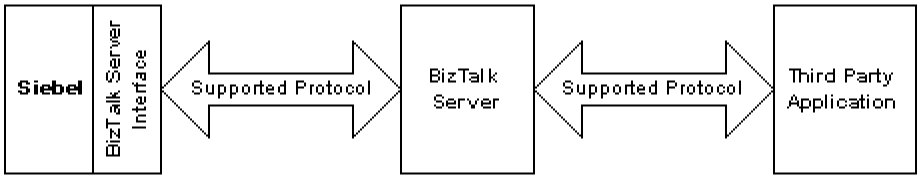


Figure 5. Siebel Application and BizTalk Inbound and Outbound Interfacing

- **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 158.
❑	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 159 and "Data Types" on page 164.

Installing and Configuring Software for Servers and Clients

Before you can establish a relationship with your trading partners though 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 *Siebel System Requirements and Supported Platforms*.

Siebel Server

- 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 Server
- SQL Server (Siebel DB)
- Siebel EAI Transports

BizTalk Server

- Microsoft BizTalk Server 2000 and 2002 requirements:
  - BizTalk Server
  - Microsoft Windows 2000 Server, or Windows 2000 Advanced Server
  - Windows 2000
  - SQL Server or SQL Server 2000
  - Microsoft Visio 2000
  - Microsoft Internet Explorer

Siebel Clients

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

Siebel Integration Objects

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

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

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.

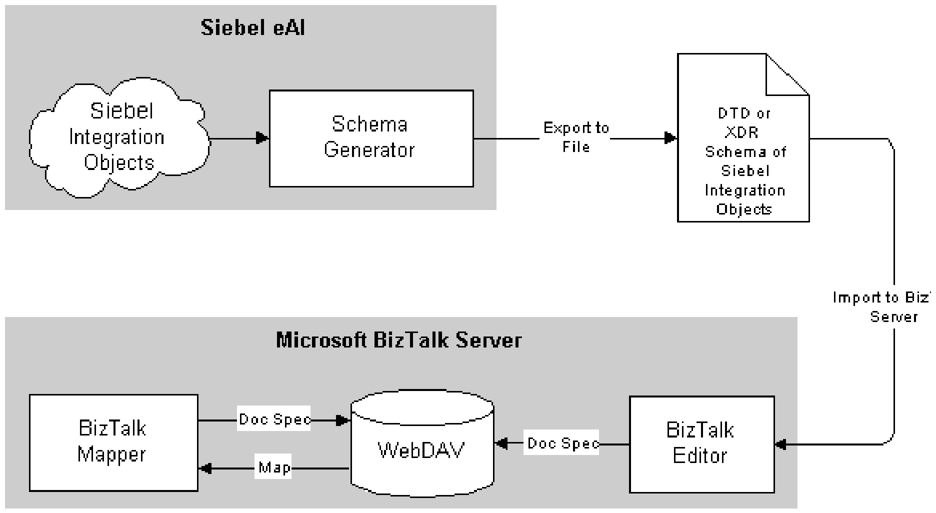


Figure 6. Exposing Siebel Integration Objects to BizTalk Server

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.

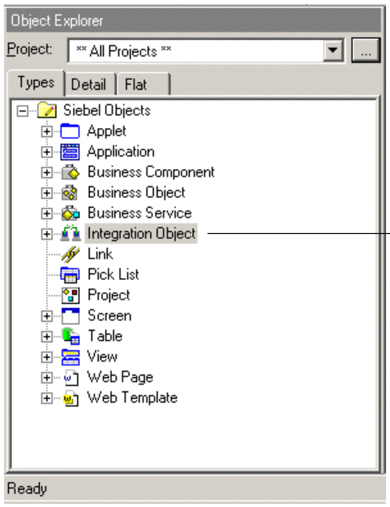
Checklist

❑	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 161.
❑	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 163.

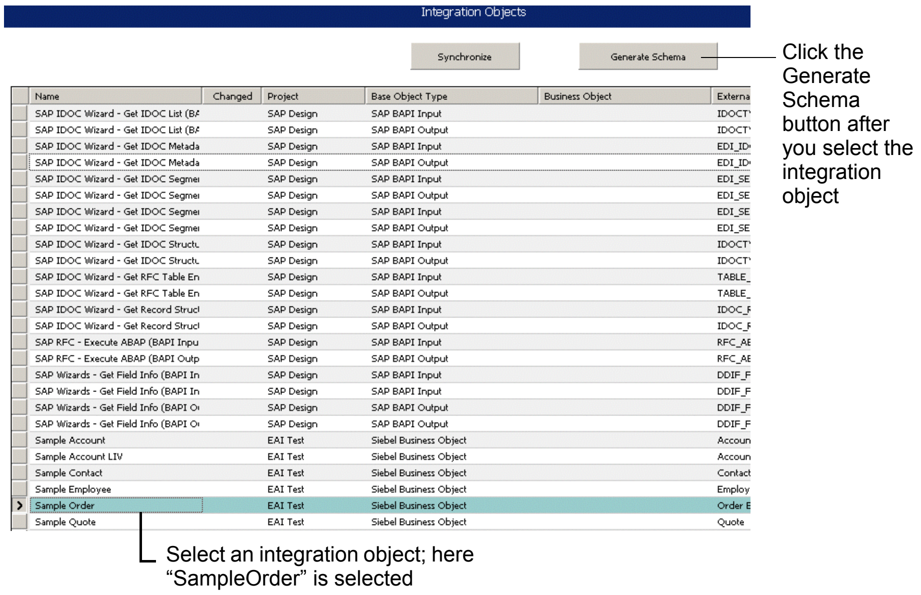
To generate a new DTD or XDR schema

1 Start Siebel Tools.

2 Select Integration Object in the Objects Explorer window as shown in the following figure.



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


Integration Objects
Synchronize
Generate Schema
Click the Generate Schema button after you select the integration object
Select an integration object; here “SampleOrder” is selected

	Name	Changed	Project	Base Object Type	Business Object	Externa
	SAP IDOC Wizard - Get IDOC List (BA		SAP Design	SAP BAPI Input		IDOCT'
	SAP IDOC Wizard - Get IDOC List (BA		SAP Design	SAP BAPI Output		IDOCT'
	SAP IDOC Wizard - Get IDOC Metada		SAP Design	SAP BAPI Input		EDI_ID
	SAP IDOC Wizard - Get IDOC Metada		SAP Design	SAP BAPI Output		EDI_ID
	SAP IDOC Wizard - Get IDOC Segme		SAP Design	SAP BAPI Input		EDI_SE
	SAP IDOC Wizard - Get IDOC Segme		SAP Design	SAP BAPI Output		EDI_SE
	SAP IDOC Wizard - Get IDOC Segme		SAP Design	SAP BAPI Input		EDI_SE
	SAP IDOC Wizard - Get IDOC Segme		SAP Design	SAP BAPI Output		EDI_SE
	SAP IDOC Wizard - Get IDOC Structu		SAP Design	SAP BAPI Input		IDOCT'
	SAP IDOC Wizard - Get IDOC Structu		SAP Design	SAP BAPI Output		IDOCT'
	SAP IDOC Wizard - Get RFC Table En		SAP Design	SAP BAPI Input		TABLE_
	SAP IDOC Wizard - Get RFC Table En		SAP Design	SAP BAPI Output		TABLE_
	SAP IDOC Wizard - Get Record Struct		SAP Design	SAP BAPI Input		IDOC_F
	SAP IDOC Wizard - Get Record Struct		SAP Design	SAP BAPI Output		IDOC_F
	SAP RFC - Execute ABAP (BAPI Inpu		SAP Design	SAP BAPI Input		RFC_AE
	SAP RFC - Execute ABAP (BAPI Outp		SAP Design	SAP BAPI Output		RFC_AE
	SAP Wizards - Get Field Info (BAPI In		SAP Design	SAP BAPI Input		DDIF_F
	SAP Wizards - Get Field Info (BAPI In		SAP Design	SAP BAPI Input		DDIF_F
	SAP Wizards - Get Field Info (BAPI O		SAP Design	SAP BAPI Output		DDIF_F
	SAP Wizards - Get Field Info (BAPI O		SAP Design	SAP BAPI Output		DDIF_F
	Sample Account		EAI Test	Siebel Business Object		Accoun
	Sample Account LIV		EAI Test	Siebel Business Object		Accoun
	Sample Contact		EAI Test	Siebel Business Object		Contact
	Sample Employee		EAI Test	Siebel Business Object		Employ
>	Sample Order		EAI Test	Siebel Business Object		Order E
	Sample Quote		EAI Test	Siebel Business Object		Quote

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

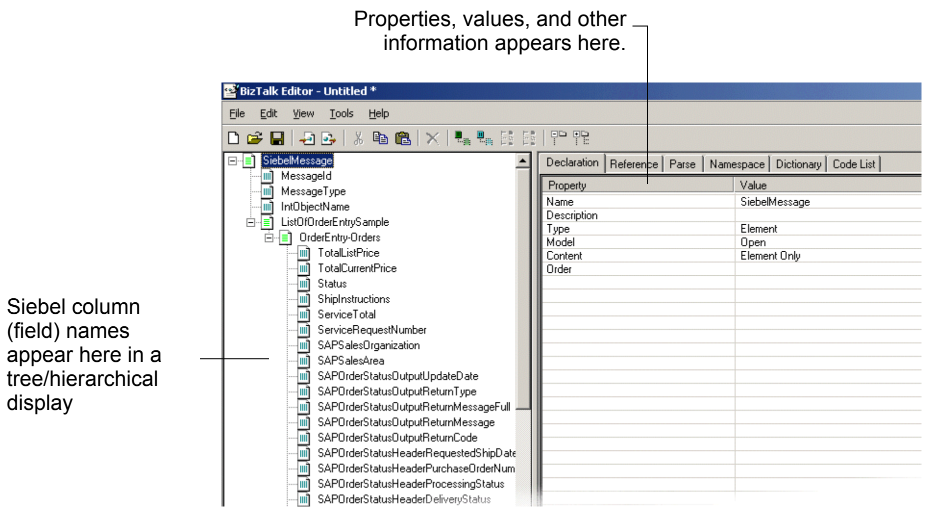
1 Open the BizTalk Server Schema Editor on the BizTalk Server machine.

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.


Properties, values, and other information appears here.
Siebel column (field) names appear here in a tree/hierarchical display
BizTalk Editor - Untitled *
File
Edit
View
Tools
Help
Declaration
Reference
Parse
Namespace
Dictionary
Code List

- SiebelMessage
  - MessageId
  - MessageType
  - IntObjectName
  - ListOfOrderEntrySample
    - OrderEntry-Orders
      - TotalListPrice
      - TotalCurrentPrice
      - Status
      - ShipInstructions
      - ServiceTotal
      - ServiceRequestNumber
      - SAPSalesOrganization
      - SAPSalesArea
      - SAPOrderStatusOutputUpdateDate
      - SAPOrderStatusOutputReturnType
      - SAPOrderStatusOutputReturnMessageFull
      - SAPOrderStatusOutputReturnMessage
      - SAPOrderStatusOutputReturnCode
      - SAPOrderStatusHeaderRequestedShipDate
      - SAPOrderStatusHeaderPurchaseOrderNum
      - SAPOrderStatusHeaderProcessingStatus
      - SAPOrderStatusHeaderDeliveryStatus

Property	Value
Name	SiebelMessage
Description	
Type	Element
Model	Open
Content	Element Only
Order	

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.

Data Types

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

Table 20 presents data types for the Siebel Field object type and the current corresponding mapping to XDR data types.

Table 20. 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 3, "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 46.
❑	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 48.

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 3, "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.

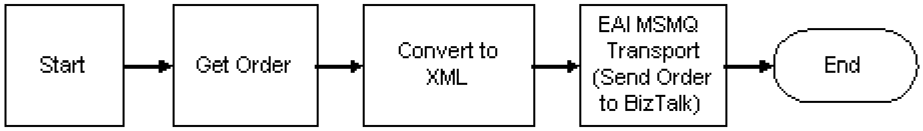
   NOTE: For each BizTalk-specific step, see Microsoft BizTalk Server documentation for the details.

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 163.
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 application-level menu, choose View > Site Map > Business Process Administration > 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	*`record # for an order`*

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$\\`name of the queue`	-	-
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 3, "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 46.
❑	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 57.

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 3, "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 163.
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`.

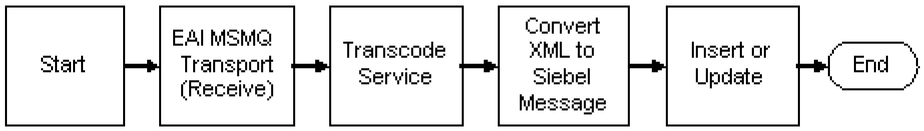
d Click the Advanced tab and enter a Channel Name, such as Oracle to Siebel MSMQ Channel.

8 Click OK to finish.

To create the Siebel workflow for MSMQ Inbound

1 From the application-level menu, choose View > Site Map > Business Process Administration > 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 3, "EAI MSMQ Transport."

5 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	-	EncodingToEndcoding	-	-
Source Encoding	-	UTF-16	-	-
Target Encoding	-	UTF-8	-	-
< Value >	Process Property	-	-	< Value >

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

6 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

7 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

8 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 156.
❑	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 177.

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 176.
❑	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 179, and "COM Outbound Transport Parameters" on page 178.
❑	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 180.

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 21 presents the outbound parameters.

Table 21. 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 *Source* 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 163.

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.

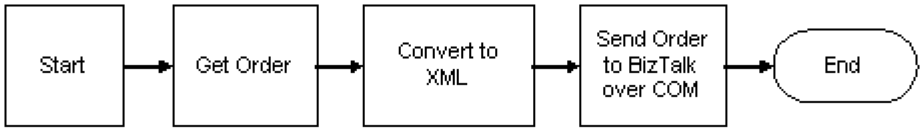
   d. Select an appropriate map, if necessary.

   e. Complete other pages as required and select Finish to complete.

To create the Siebel workflow for COM Outbound

1. From the application-level menu, choose View > Site Map > Business Process Administration > 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	*`record # for an order`*

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 179.)	-	-

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 21 on page 179.

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 184
- "Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server" on page 187

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 184.
❑	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 185.
❑	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 190.
❑	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 190.

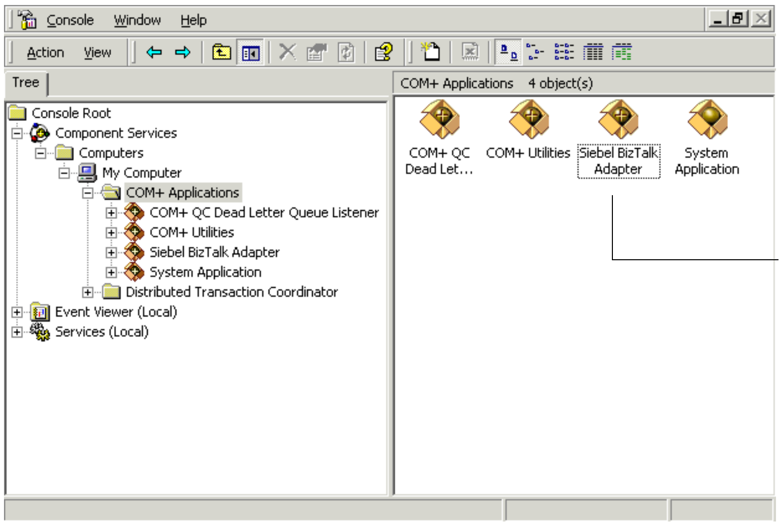
To install Siebel ActiveX Data Controls and AIC on the BizTalk Server machine

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 *SIEBSRVR_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 188.
❑	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 189.
❑	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 189.
❑	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 190.
❑	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 190.

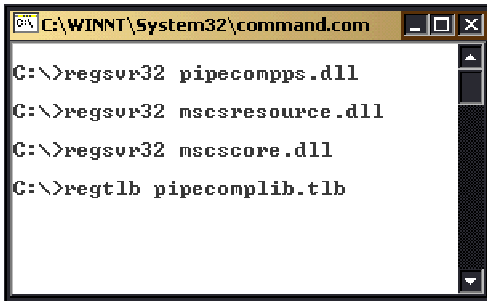
To copy BizTalk Server DLL and Type Library files to the Siebel Server

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:\>regsvr32 pipecompps.dll

C:\>regsvr32 mscsresource.dll

C:\>regsvr32 mscscore.dll

C:\>regtlb pipecomplib.tlb
```

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

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

To register AIC as a COM+ package remote communication

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

  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 185.

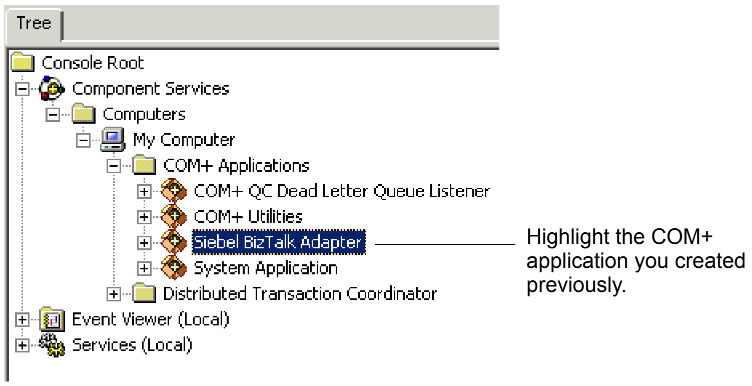
NOTE: Siebel AIC (Application Integration Components) and ADC (ActiveX Data Controls) are installed as a part of the Siebel Server or the EAI Connectors installation.

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 *Siebel Server Installation Guide for Microsoft Windows* and *Siebel Tools Online Help* 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 *Siebel Server Installation Guide for Microsoft Windows* for more information.

Checklist

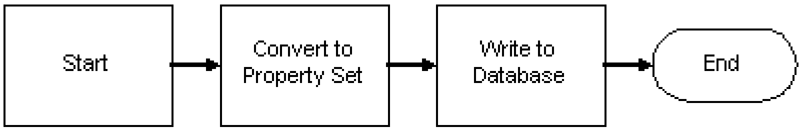
❑	Set up the Siebel AIC Inbound Transport. For details, see "Setting up the AIC Inbound Transport" on page 183.
❑	Create the workflow to process inbound documents from BizTalk using AIC. For details, see "To create a workflow to receive inbound XML documents over AIC from BizTalk" on page 192.
❑	After 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 194.

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 application-level menu, choose View > Site Map > Business Process Administration > 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 195). 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 192.
- **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://HOST/EnterpriseServer/
SCCObjMgr_enu/SiebelServer" lang="ENU"
```

  Where:

  HOST = the machine name where the Siebel Gateway 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 204

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 198.
❑	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 198.

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 197.

❑	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 200.

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 163.
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.

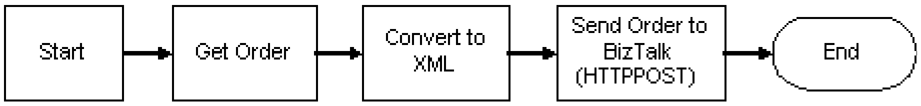
  - **a** Search for `sChannel` in the file.
  - **b** Remove the comment designator from this statement:

```
sChannel = Channel Name you just created
```

To create the Siebel workflow for HTTP Outbound

1 From the application-level menu, choose View > Site Map > Business Process Administration > 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	*`record # for an order`*

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 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 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, `http://`*`WebServer/virtual directory name/`*`SiebelBizTalkOutHTTP.asp` where: *`WebServer`* is the name of your Web server and *`virtual directory name`* is the virtual directory you set up in "To create a virtual directory for HTTP Outbound" on page 198, such as BizTalkReceive.	-	-

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

7 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 4, "EAI HTTP Transport."

Setting Up HTTP-Based AIC to Run on BizTalk Server

This section explains how to set up the required components for running the HTTP-based AIC Inbound Transport on the BizTalk Server machine. Although the AIC can run either on the Siebel Server machine or on the BizTalk Server machine, it is recommended that you run the AIC on the BizTalk Server machine locally for ease of configuration. If you would like to run the AIC on the Siebel Server, see "Setting Up the AIC Inbound Transport to Run Remotely on the Siebel Server" on page 187.

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 205.

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 204.

❑ 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 69.

❑ 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.

Integrating with J2EE Application Server 8

Many enterprises, especially those involved in eBusiness, develop and implement Java applications to meet a variety of business requirements. Typically, these applications combine existing enterprise information systems with new business functions to deliver services to a broad range of users.

Siebel applications generate JavaBeans for existing integration objects and business services by means of the Siebel JavaBean Wizard. By generating JavaBeans that represent Siebel integration objects or business services, Siebel applications help facilitate the creation of J2EE applications that need to interface with Siebel eBusiness Applications. The Siebel Code Generator produces JavaBeans for integration objects, business services, and their related components, allowing you to identify and use the Java code you need for your application.

As illustrated in Figure 7, in addition to the generation of JavaBeans, Siebel applications support the J2EE Connector Architecture with the Siebel Resource Adapter. Siebel applications also provide a range of tools for integrating with J2EE environments, supporting both loosely coupled and tightly coupled integration architectures using HTTP and IBM MQSeries.

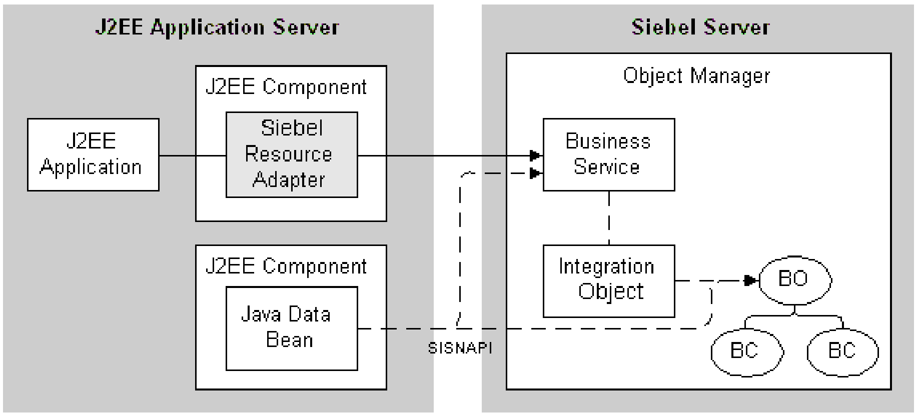


Figure 7. J2EE Application Server Integration Architecture

The Siebel JavaBean Wizard

To ease integration between the Siebel application and J2EE Application Servers, Siebel applications provide a wizard that generates JavaBeans for business services and integration objects. In Siebel Tools, you generate JavaBeans for the Siebel objects you want to interface with and then include the generated JavaBeans in the J2EE application or component you are constructing. The generated JavaBeans act as client-side proxy stubs to the corresponding objects on the Siebel Server and are not the actual business service logic written in Java.

Siebel Tools ships with a wizard for creating JavaBeans based on Siebel business services or Siebel integration objects. Creating JavaBeans based on business services or integration objects provides a uniform mechanism to interact with these objects from within a J2EE component or application.

Using these beans, you can develop J2EE components that retrieve data and content from within Siebel applications and trigger events within Siebel applications depending upon the nature of the interaction. The generation of these beans reduces the complexity of the integration, while providing a standard, component-based interface to Siebel applications. JavaBeans generated by the Wizard may be used in conjunction with either the Siebel Java Data Bean, or the Siebel Resource Adapter.

For more information about the Siebel Java Data Bean, see *Siebel Tools Online Help*. For more information about the Siebel Resource Adapter, see "The Siebel Resource Adapter" on page 238.

Generating JavaBeans for Integration Objects

This section describes how to generate JavaBeans for Siebel integration objects, as well as the methods that are generated for each JavaBean. For information on how to generate JavaBeans for Siebel business services, see "Generating JavaBeans for Business Services" on page 220.

You use the JavaBean Wizard to produce the JavaBean based upon Siebel integration objects. JavaBeans based upon integration objects are designed to be used with EAI Siebel Adapter and may be used to query, delete, upsert, and synchronize information maintained within Siebel's database. The structure of the JavaBeans generated for an integration object and their components is shown in Figure 8. The integration object and its components each have their own Java class, stored in the com.siebel.local.*`Integration Object Name`* package.

The Siebel Java Data Bean provides a Java-based programming interface to Siebel eBusiness Applications. Using the Siebel Java Data Bean, you can build Java or J2EE components that interact with the Siebel application object, business objects, business components, business services and property sets.

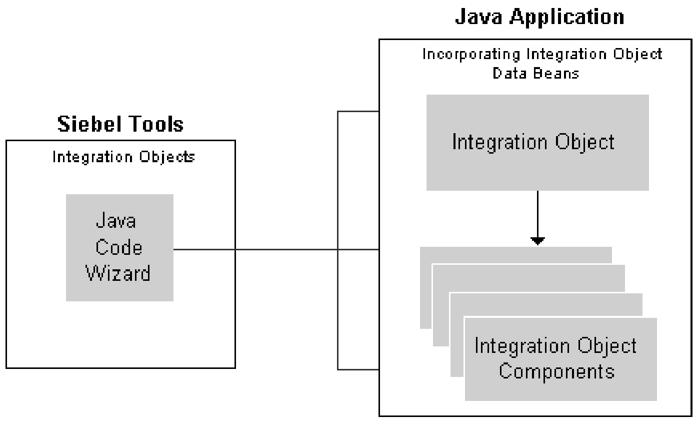


Figure 8. Generate Code Wizard Creates Java for Siebel Integration Objects

To generate Java code for a Siebel Integration Object

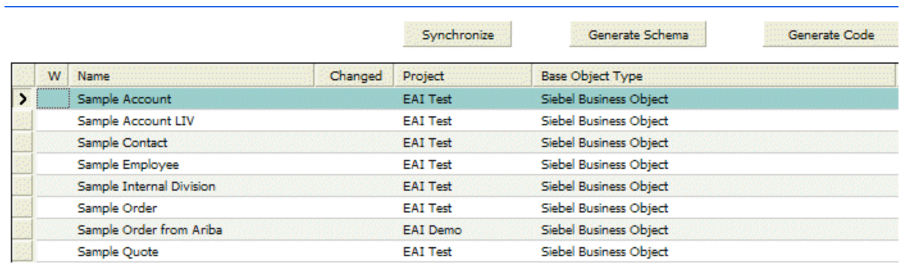
1 Log in to Siebel Tools and lock the project you are going to work on.

> **NOTE:** For information on how to use Siebel Tools and how to lock projects and so on, consult *Siebel Tools Online Help*.

2 Select Integration Objects from the Types tab on the Object Explorer.

3 Select the integration object for which you want to create Java code, as shown in the following illustration.

On the right top corner of the Integration Object list, there is a set of three buttons. The following illustration shows the Sample Account integration object highlighted.



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 the Generate Code button.

5 Complete the wizard:

a Leave the business service as is (there is only one applicable: the Siebel Code Generator).

b Select either JDB (Java Data Bean) or JCA (J2EE Connector Architecture/ Siebel Resource Adapter) for the Supported Language.

c Browse to select an existing folder as the output folder. Your Java code for the selected integration objects are stored in subdirectories there, as explained next.

d Click Finish.

The code is generated and the wizard closes, returning you to the Integration Objects form.

About the Folders and Files Created for Integration Object Java

A new folder named *com* is created as the first subfolder under the folder you specified in the wizard as your root folder. Beneath it is a *siebel* directory, and below that a *local* directory. Here is where you find separate folders for the Java code generated for any integration object for which you have generated Java code, as shown in Figure 9.

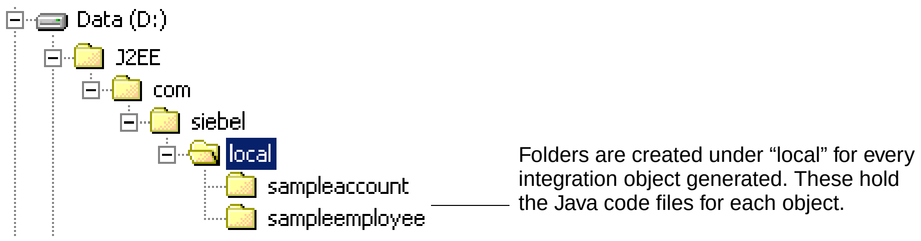

Data (D:)
J2EE
com
siebel
local
sampleaccount
sampleemployee
Folders are created under “local” for every integration object generated. These hold the Java code files for each object.

Figure 9. Directory Structure Created to Contain Java Code for Integration Objects

The files are created under the last leaf of the directory as shown in Figure 10.

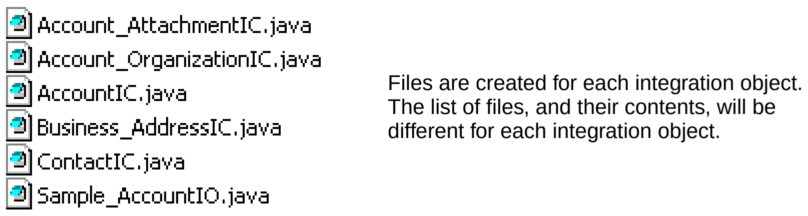

Account_AttachmentIC.java
Account_OrganizationIC.java
AccountIC.java
Business_AddressIC.java
ContactIC.java
Sample_AccountIO.java
Files are created for each integration object. The list of files, and their contents, will be different for each integration object.

Figure 10. Java Bean Files for Sample Account Integration Object

Generated JavaBean for the Sample Account Integration Object

Table 22 presents a description of the Java classes generated for the Sample Account integration object. The files the JavaBean Wizard generates varies for each integration object definition. The table presented here is designed to provide you with an idea of the structure of the Java classes that are generated for a typical integration object, as well an explanation of what each file represents.

Table 22. Java File List for Sample Account Integration Object

Java File	Description
Account_AttachmentIC.java	Java class that corresponds to the Account Attachment integration object component. Provides a set of methods that allow Java developers to construct the Account Attachment integration object component, convert to and from a property set, and to define relationships between integration object components.
Account_OrganizationIC.java	Java class that corresponds to the Account Organization integration object component. Provides a set of methods that allow Java developers to construct the Account Organization integration object component, convert to and from a property set, and to define relationships between integration object components.
AccountIC.java	Java class that corresponds to the Account integration object component. Provides a set of methods that allow Java developers to construct the Account integration object component, convert to or from a property set, and to define relationships between integration object components.
Business_AddressIC.java	Java class that corresponds to the Business Address integration object component. Provides a set of methods that allow Java developers to construct the Business Address integration object component, convert to or from a property set, and to define relationships between integration object components.

Table 22. Java File List for Sample Account Integration Object

Java File	Description
ContactIC.java	Java class that corresponds to the Contact integration object component. Provides a set of methods that allow Java developers to construct the Contact integration object component, convert to or from a property set, and to define relationships between integration object components.
Sample_AccountIO.java	Java class that corresponds to the Sample Account integration object. Provides a set of methods that allow Java developers to construct an integration object, add integration object components to the integration object and convert to or from a property set.

Default Java Methods for Integration Objects

Table 23 presents default methods generated by Siebel Tools for integration objects for Java code.

Table 23. Default Java Methods for Integration Objects

Object	Description
addfIntObject(IntObjComp ioc)	Adds an integration object component object to the integration object.
clone	Makes a copy of the integration object.
equals(Object)	Helper method to determine if two Integration Object objects are equal.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an integration object from a property set.
getfIntObjectFormat	Access method that returns a String containing the format of the integration object.
getfIntObjectName	Access method that returns the integration object name property.
getfintObjInst	Access method that returns a Vector representation of the integration object.

Table 23. Default Java Methods for Integration Objects

Object	Description
getfMessageId	Access method that returns the MessageId property of the integration object.
getfMessageType	Access method that returns the MessageType property of the integration object.
getfOutputIntObjectName	Access method that returns the OutputIntObjectName property of the integration object.
Integration_ObjectIO()	Default constructor.
Integration_ObjectIO(SiebelPropertySet ps)	This method creates an integration object (and its hierarchy) based on a property set.
setfIntObjectFormat	Access method that sets the IntObjectFormat property of the integration object.
setfIntObjectName	Access method that sets the IntObjectName property of the integration object.
setfMessageId	Access method that sets the MessageId property of the integration object.
setfMessageType	Access method that sets the MessageType property of the integration object.
setfOutputIntObjectName	Access method that sets the OutputIntObjectName property of the integration object.
toPropertySet	Method that serializes the object into a Siebel property set.

Default Java Methods for Integration Object Components

Table 24 presents the default methods generated by Siebel Tools for integration object components for Java code.

Table 24. Default Java Methods for Integration Object Components

Object Component	Description
addfChildIntObjComp(ChildIntObjComp)	This method creates a child integration object component of type ChildIntObjComp and associate with the parent integration object component.
clone	Makes a copy of the integration object.
equals(Object)	Helper method to determine if two integration object objects are equal.
fromPropertySet(SiebelPropertySet)	This method populates the integration object component based upon the contents of a property set.
getfChildIntObjComp	Returns a Vector containing all child integration object components of type ChildIntObjComp associated with the parent integration object component.
getfFieldName()	Access method that returns the value of FieldName.
IntObjCompIC()	Default constructor.
IntObjCompIC(SiebelPropertySet)	This method creates an integration object component from a property set.
setfFieldName(val)	Access method that sets class property FieldName to val.
toPropertySet	This method converts an integration object component to a property set.

Generating JavaBeans for Business Services

This section describes how to generate JavaBeans for Siebel business services. For information on how to generate JavaBeans for Siebel integration objects, see "Generating JavaBeans for Integration Objects" on page 213.

You use the JavaBean Wizard to generate JavaBeans based on Siebel business services. The wizard produces JavaBeans for the business service and the input and output parameters for each business service method defined in Siebel Tools, as shown in Figure 11. If methods of the business service take an integration object as a parameter, the JavaBeans for the integration objects are not generated. In order to generate JavaBeans for integration objects, see "Generating JavaBeans for Integration Objects" on page 213. The business service, input and output parameters for each method have their own Java class, stored in the com.siebel.jdb.service.*`Business Service Name`* or com.siebel.jca.service.*`Business Service Name`* package.

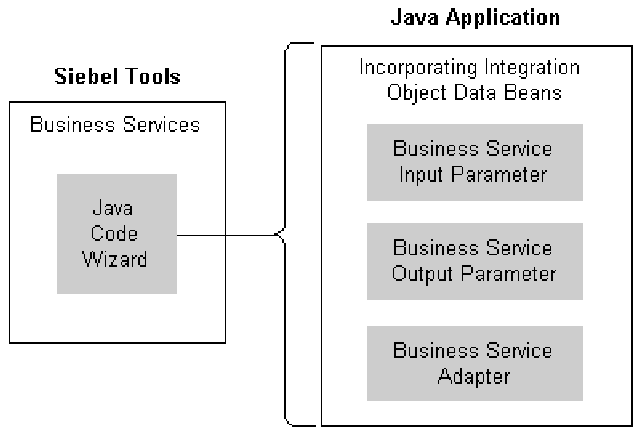

Java Application
Siebel Tools
Business Services
Java Code Wizard
Incorporating Integration Object Data Beans
Business Service Input Parameter
Business Service Output Parameter
Business Service Adapter

Figure 11. Generate Code Wizard Creates Java for Siebel Business Services

To generate Java code for a Siebel Business Service

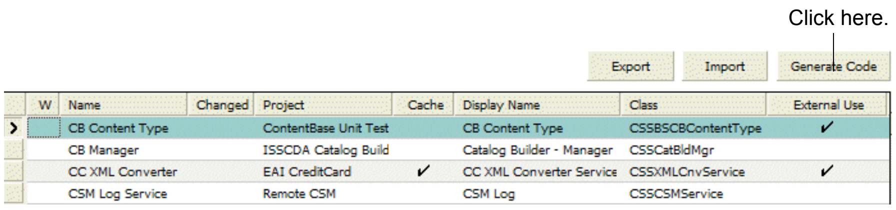
1. Log in to Siebel Tools and lock the project you are going to work on.

   NOTE: For information on how to use Siebel Tools and how to lock projects and so on, consult *Siebel Tools Online Help*.

2. Select Business Service from the Types tab on the Object Explorer.

3 Select the business service for which you want to create Java code, as shown in the following illustration.

On the right top corner of the list, there is a set of three buttons. The following illustration shows the CC XML Converter business service highlighted.



W	Name	Changed	Project	Cache	Display Name	Class	External Use
	CB Content Type		ContentBase Unit Test		CB Content Type	CSSBSCBContentType	✓
	CB Manager		ISSCDA Catalog Build		Catalog Builder - Manager	CSSCatBldMgr	
	CC XML Converter		EAI CreditCard	✓	CC XML Converter Service	CSSXMLCnvService	✓
	CSM Log Service		Remote CSM		CSM Log	CSSCSMService	

4 Click the Generate Code button.

NOTE: If the business service you selected is not available to be exported, you see an error message rather than the wizard.

5 Complete the wizard:

a Leave the business service as is (there is only one applicable: the Siebel Code Generator).

b Select either JDB (Java Data Bean) or JCA (J2EE Connector Architecture/ Siebel Resource Adapter) for the Supported Language.

c Browse to select an existing folder as the output folder. Your Java code for the selected business service will be stored in subdirectories there, as explained in the next section.

d Click Finish.

About the Folders and Files Created for Business Service Java

As shown in Figure 12, a new folder named *com* is created as the first subfolder under the folder you specified in the wizard as your root folder. Beneath it are the *siebel\eai* directories. Finally, under the *service* directory you find separate folders for the Java code generated for any business service for which you have generated Java code.

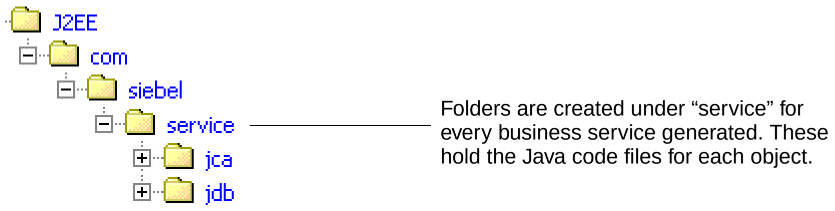

J2EE
com
siebel
service
jca
jdb
Folders are created under "service" for every business service generated. These hold the Java code files for each object.

Figure 12. Directory Structure Created to Contain Java Code for Integration Objects

The files are created under the last leaf of the directory as shown in Figure 13.

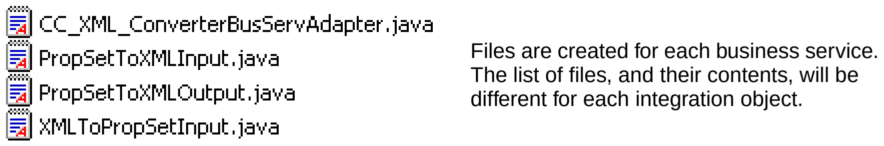

CC_XML_ConverterBusServAdapter.java
PropSetToXMLInput.java
PropSetToXMLOutput.java
XMLToPropSetInput.java
Files are created for each business service. The list of files, and their contents, will be different for each integration object.

Figure 13. JavaBean Files for Sample Account Integration Object

Naming Convention for Generated Code

Following are the naming conventions for generated code:

Package names

Start with *com.siebel.local* for integration object and *com.siebel.service* for business service and end with a modified primary class name with all special characters removed and all in lowercase.

Business service example, "Siebel Code Generator" -- > "com.siebel.service.siebelcodegenerator"

Integration object example, "Siebel Way Cool Interface" -- > "com.siebel.local.siebelwaycoolinterface"

Class names All special characters are replaced with under bars ("_"). Business services will have "BusServAdapter" appended to the end, and integration objects will have "IO" appended to the end and the integration components will have "IC" appended to the end. Business service's method inputs will be the name of the method, defined in Tools, with either Input or Output appended to it respectively.

Business service example, "Siebel Code Generator" --> "SiebelCodeGeneratorBusServAdapter"

Integration object example, "Siebel Way Cool Interface" --> "SiebelWayCoolInterfaceIO"

Integration Component example, "Siebel Way Cool Interface Component" --> "SiebelWayCoolInterfaceComponentIC"

Other naming conventions All special characters are replaced with under bars ("_"). All variables such as integration component fields or user properties will have a "f" pre-pended to the name. All business service methods will have a "m" pre-pended to the name.

Business service method example, "GenerateCode" --> "mGenerateCode"

Field example, "My Field" --> "fMy_Field"

Generated Java Files for a Typical Business Service

Table 25 provides an explanation for the Java classes generated for the CC XML Converter business service. JavaBeans generated for each business service vary based upon the method defined and the Input/Output arguments. The table is presented to give you an idea of the Java classes generated for a typical business service and an explanation for what each class represents. Please note that for the CC XML Converter business service, the XMLToPropSet method does not have any Output parameters, which is why the output class for this method was not generated. Table 26, Table 27, and Table 28 show more specific information about the default methods.

Table 25. Java File List for CC XML Converter Business Service

Java File	Description
CC_XML_Converter BusServiceAdapter.java	Java class that represents to the CC XML Converter business services. Provides a set of methods to construct the Business Service Adapter Class as well as invoke the business service methods.
PropSetToXMLInput.java	Java class that represents the input parameters of the PropSetToXML method of the CC XML Converter business service. Provides a set of access methods to set and get the method input parameters as well as convert to and from a property set.
PropSetToXMLOutput.java	Java class that represents the output parameters of the PropSetToXML method of the CC XML Converter business service. Provides a set of access methods to set and get the method output parameters as well as convert to and from a property set.
XMLToPropSetInput.java	Java class that represents the input parameters of the XMLToPropSet method of the CC XML Converter business service. Provides a set of access methods to set and get the method input parameters as well as convert to and from a property set.

Table 26. Java Methods for the Business Service Adapter Class

Method	Description
BusServAdapter()	Constructor.
BusServAdapter(SiebelDataBean)	Constructor.
BusServAdapter(String, String, String)	Constructor.
mBusSvcMethodName(methodInput)	Method that invokes the specified business service method.

Table 27. Default Java Methods for Business Service Method Input Parameters

Method	Description
methodInput()	Constructor.
methodInput(SiebelPropertySet)	Constructor.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an integration object from a property set.
toPropertySet()	Method that serializes the object into a Siebel property set.
getfMethodArgName()	Access method to get the value of business service method argument.
setfMethodArgName()	Access method to set the value of a business service method argument.

Table 28. Default Java Methods for Business Service Method Output Parameters

Method	Description
MethodOutput()	Constructor.
methodOutput(SiebelPropertySet)	Constructor.
fromPropertySet(SiebelPropertySet)	Method that allows a developer to define an integration object from a property set.

Table 28. Default Java Methods for Business Service Method Output Parameters

Method	Description
toPropertySet()	Method that serializes the object into a Siebel property set.
getfMethodArgName()	Access method to get the value of business service method argument.
setfMethodArgName()	Access method to set the value of a business service method argument.

Using the Java Data Bean

The Siebel Java Data Bean provides a Java-based programming interface to Siebel eBusiness Applications. Using the Siebel Java Data Bean, you can build Java or J2EE components that interact with the Siebel application object, business objects, business components, business services and property sets.

For additional information on the interfaces exposed by the Siebel Java Data Bean, see *Siebel Tools Online Help*.

The Development Environment

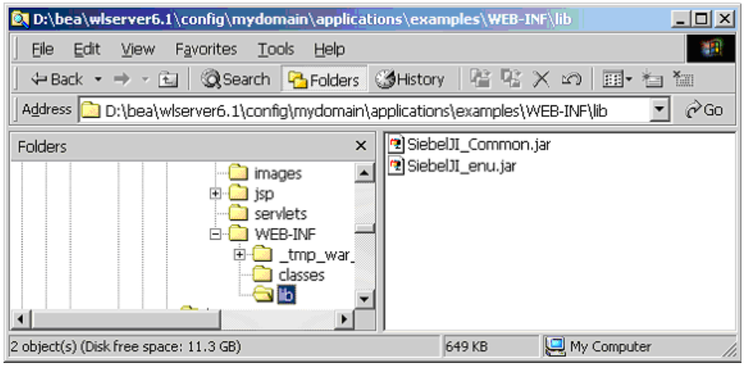
Three Siebel `.jar` files are needed when you compile your Java application, and run time. The files are:

- `SiebelJI.jar`
- `SiebelJI_common.jar`
- `SiebelJI_`*`lang`*`.jar`.

  Where *`lang`* is the installed language pack; for example, SiebelJI_enu.jar for English or SiebelJI_jpn.jar for Japanese.

The .jar files may be installed through the Siebel Enterprise Server install, or by installing Siebel Tools. For more information please refer to the *Siebel Server Installation Guide for UNIX* or *Siebel Server Installation Guide for Microsoft Windows*. To deploy the J2EE applications using the Siebel Java Data Bean or JavaBeans generated by the Java Wizard, include these `.jar` files, so that the J2EE application can access the Java Data Bean class files.

To deploy in WebLogic the sample Servlet/JSP application, for example, place the `.jar` files in the `lib` subdirectory of WEB-INF corresponding to the application, as shown in Figure 14.


D:\bea\wlserver6.1\config\mydomain\applications\examples\WEB-INF\lib
File Edit View Favorites Tools Help
Back
Search
Folders
History
Address
Go
Folders
images
jsp
servlets
WEB-INF
_tmp_war_
classes
lib
My Computer

D:\bea\wlserver6.1\config\mydomain\applications\examples\WEB-INF\lib

SiebelJI_Common.jar
SiebelJI_enu.jar

2 object(s) (Disk free space: 11.3 GB) 649 KB

Figure 14. Example of .jar Files Location

NOTE: For other application servers, refer to the application server's documentation for information on where to place the `.jar` files.

To install the Siebel .jar files

- Copy the `.jar` files to your working directories and make sure the three Siebel jar files are in your CLASSPATH.

To compile source files using Siebel Data Bean

- You combine source files using the command line executable, `javac.exe.` The Siebel files can also be used directly in your development environment. To compile using command line, enter one of the following strings, depending on which type of Java you are using. The command string is entered on a single line.

  For Sun's JVM:

  ```
  javac -classpath
  .;SiebelJI_Common.jar;SiebelJI_lang.jar;SiebelJI.jar <source
  files>
  ```

To run applications containing Siebel Java Data Beans

- Run class files that include Siebel Data Beans or JavaBeans generated by the Java Wizard by entering the following at the command line:

  ```
  jre -cp .;SiebelJI.jar;SiebelJI_Common.jar;SiebelJI_lang.jar
  <class files>
  ```

Using Java Code Generated by Siebel eBusiness Applications

Set up the development environment as described in "Using the Java Data Bean" on page 227. If you are using an Integrated Development Environment (IDE), be sure to include the `.jar` files in the classpath or input classes for your project. Refer to your IDE documentation on how to include external class/jar files in to your project.

NOTE: A javadoc jar file (Siebel_JavaDoc.zip) is installed when you choose to install the Siebel Java Integrator option (a component of the Tools or Siebel Server installer). This jar file is in the same location as the SiebelJI* jar files under the *INSTALLED_DIR*\classes directory. This jar file contains the up-to-date java doc for the Siebel Java Data Bean, Siebel Resource Adapter, and dependent classes.

Sample Java Code 1

The following is a code sample that describes the use of the Siebel Java Data Bean in a Servlet. The Java Data Bean can be used in a similar way in Java Server Pages (JSPs) or in Enterprise JavaBeans (EJBs).

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

...

SiebelDataBean m_dataBean = null;

SiebelBusObject sbo_busObject = null;

SiebelBusComp sbc_busComp = null;


try {

  boolean result;

...

  String userLoginName =...

  String strJdbLoginUser =...

  String strJdbLoginPwd =...

...

  m_dataBean = new SiebelDataBean();

  result = m_dataBean.login("siebel://GatewayServer/
EnterpriseName/AppObjMgr/SiebelServer",strJdbLoginUser,
strJdbLoginPwd, "enu");

  SiebelBusObject sbo_busObject
= m_dataBean.getBusObject("Contact");

  SiebelBusComp sbc_busComp =
sbo_busObject.getBusComp("Contact");


  sbc_busComp.setViewMode(3);

  sbc_busComp.clearToQuery();

  sbc_busComp.activateField("First Name");

  sbc_busComp.activateField("Last Name");

  sbc_busComp.activateField("Id");
```

```
    sbc_busComp.setSearchSpec("Login Name",userLoginName);

    sbc_busComp.executeQuery2(true,true);

    "

  if (sbc_busComp.firstRecord()) {

     System.out.println("Contact ID:
     "+sbc_busComp.getFieldValue("Id")+" ; First
     Name:"+sbc_busComp.getFieldValue("First Name")+ " ;Last
     Name:"+sbc_busComp.getFieldValue("Last Name"));

   }

  sbc_busComp.release();

  sbo_busObject.release();

  m_dataBean.logoff();

   } catch (SiebelException se) {

             System.err.println("Error:"+se.getErrorMessage());

             se.printStackTrace();

   }
```

About the Code Sample

The code sample above shows a simple use of the Java Data Bean from a Java Servlet to search for a particular login name in the Contact business component. The first step in using the Java data bean is to log into the Object Manager of the Siebel application. The first parameter, the connection string, specifies the protocol, machine name, enterprise name, object manager name, and the server name. After you are logged in to the Object Manager, you can use the getBusObject and getBusComp methods to obtain the business object and business component objects.

NOTE: You may have to change this based on your login permissions.

The code sample activates fields to allow the query to retrieve data for the specified fields, specifies the search criteria, and executes the query. For additional details on these methods, see *Siebel Object Interfaces Reference*. If the query is successful, the first and last name of the contact is printed to the System.out.

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

```
if (sbc_busComp.firstRecord()) {

//obtain the fields/values from this record

...

while (sbc_busComp.nextRecord()){

      //obtain the fields/values from this record

     ...

    }

   }
```

Code Sample Using Integration Object JavaBeans

The following is a code sample invoking the QueryByExample method of the Siebel Account business service. The JDBSiebelAccount class, which is described in the section below, provides code examples that show how to perform the following tasks:

- How to invoke a business service method, passing an integration object as a method parameter.
- Query for a list of Accounts that start with the letters *Ai*.
- Convert the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information see "The Siebel JavaBean Wizard" on page 212, for both business services and integration objects. By using the code generation facilities, many of the complexities of the Siebel property sets and business service interfaces have been abstracted, providing a standards-based JavaBean interface.

```
/* import declaration here */
```

```
public class JDBSiebelAccount {

  public static void main(String[] args) {

    Siebel_AccountBusServAdapter svc   = null;

    QueryByExampleInput qbeIn   = new QueryByExampleInput();

    QueryByExampleOutput qbeOut = new QueryByExampleOutput();

    AccountIC acctIC = new AccountIC();

    Sample_AccountIO acctIO = new Sample_AccountIO();

    Vector ioc   = null;


    /* query on accounts that start with 'Ai'. */

    acctIC.setfName("Ai*");

    acctIO.addfintObjInst(acctIC);


    qbeIn.setfSiebelMessage(acctIO);


    try {

        /* initialize the Siebel Account service using the user
name, password, connect stringm language constructor. The logoff
method will be invoked when class destroyed*/

        svc = new Siebel_AccountBusServAdapter("username",
"password","siebel://namesrvr/entserver/objectmanager/
siebelserver","lang");


    /* invoking QueryById method */

    qbeOut = svc.mQueryByExample(qbeIn);

      acctIO = null;
      acctIC = null;

      acctIO = new Sample_AccountIO(qbeOut.getfSiebelMessage()
toPropertySet());

      ioc    = acctIO.getfintObjInst();


      /* loop through all accounts returned */

       if(!ioc.isEmpty()) {

          for(int i=0; i < ioc.size(); i++) {

            acctIC = ((AccountIC)ioc.get(i));

            System.out.println(i + "  " + acctIC.getfName() + " "
      + acctIC.getfLocation());

            }/* for */

       }/* if */

    }catch(SiebelException se) {

    System.out.println(se.getDetailedMessage());

    }/* try-catch */

   }/* main */

}/* public class JCASiebelAccount */
```

Code Sample Using the JavaBean Wizard

The following is a code example using the Java Data Bean and JavaBeans based upon the Sample Account integration object and EAI Siebel Adapter business service. The DataBeanDemo class, implemented below, provides code examples that show how to perform the following tasks:

- Initialize a business service using the Java Data Bean Constructor.
- Invoke the QueryPage method.
- Convert the results of the QueryPage method into an integration object.

The following code uses the code generation facilities provided in Siebel Tools. For more information, see "The Siebel JavaBean Wizard" on page 212 for both business services and integration objects. By using the code generation facilities, many of the complexities of various Siebel APIs (property set, business services) have been abstracted and encapsulated into a JavaBean-based interface.

```
/* import declaration here */


public class DataBeanDemo {


  private String   userName   = "SADMIN";

  private String   passwd     = "SADMIN";

  private String   lang       = "enu";

  private String   connStr    = "siebel://GatewayServer/
EnterpriseName/AppObjMgr/SiebelServer";

  private String   intObjName = "Sample Account";

  public static void main(String[] args) {

    DataBeanDemo demo = new DataBeanDemo();

  }

/* main */

  public DataBeanDemo() {

    SiebelDataBean     m_dataBean  = null;

    EAI_Siebel_AdapterBusServAdapter  svc    = null;

    QueryPageInput     qpInput    = new QueryPageInput();

    QueryPageOutput    qpOutput   = new QueryPageOutput();

    int     rowNum    = 0;

    int     pageSize  = 10;

    Sample_AccountIO  acctIO  = null;
    Vector     ioc    = null;


    try {

      /* instantiate the Siebel Data Bean */

      m_dataBean = new SiebelDataBean();



      /* login to Siebel using connect string, user name,
password, lang constructor */

      m_dataBean.login(connStr, userName, passwd, lang);



      /* construct the EAI Siebel Adapter, using the data bean
constructor */

      svc = new EAI_Siebel_AdapterBusServAdapter(m_dataBean);

      svc.initialize();



      /* set QueryPage parameters to be used by the Siebel
Adapter.QueryPage method. The request will return 10 records,
starting at record 0, using the Sample Account integration
object*/

    qpInput.setfPageSize(java.lang.Integer.toString(pageSize));

    qpInput.setfOutputIntObjectName(intObjName);

    qpInput.setfStartRowNum(java.lang.Integer.toString(rowNum));

    qpOutput = svc.mQueryPage(qpInput);



      /* construct the integration object using the output of the
QueryPage method, and the property set constructor*/

      acctIO = new
Sample_AccountIO(qpOutput.getfSiebelMessage().toPropertySet());
```

```
      /* convert the results of the integration object to a vector
for processing*/

      ioc = acctIO.getfintObjInst();


      /* Loop through the result set, printing out account name
and location */

      if (!ioc.isEmpty()) {

        for (int i = 0; i < ioc.size(); i++) {

          AccountIC acctIC = ((AccountIC) ioc.get(i));

          System.out.println(acctIC.getfName() + " ------- " +
acctIC.getfLocation());

        }/* for */

      }/* if */


      /* logout once the request has been completed */

      m_dataBean.logoff();

    } catch (SiebelException e) {

      e.printStackTrace();

      try {

        /* if an error occurs, make sure that the connection is
closed */

        m_dataBean.logoff();

      } catch (SiebelException obj) {

        System.out.println(obj.getErrorMessage());

      }/* try-catch */

    }/* try-catch */
```

```
  }/* public DataBeanDemo */
}/* public class DataBeanDemo */
```

The Siebel Resource Adapter

The Siebel Resource Adapter uses the Siebel JavaBean to call business services and functions as a point of contact between application components, application servers, and enterprise information systems, like Siebel eBusiness Applications. This adapter provides support for the J2EE Connector Architecture through system-level contracts. These contracts specify how a system external to the J2EE platform can integrate with it by supporting basic functions that are handled by the J2EE container.

Using the Resource Adapter

When deploying the Siebel Resource Adapter in J2EE Application Server (BEA, WebLogic, or IBM WebSphere), the necessary Siebel and J2EE Connector JAR files are included in the server classpath. The Siebel JAR files that need to be added to the classpath are:

- SiebelJI.jar
- SiebelJI_*lang*.jar
- SiebelJI_Common.jar

Once you have added the necessary Siebel and J2EE Connector JAR files to server classpath, see the J2EE application server documentation with respect to deploying a Resource Adapter.

NOTE: Some client applications can require the Siebel Resource Adapter .jar file (Siebel_JCA_ra.jar) to be included in the classpath. The Siebel resource Adapter can be extracted from the resource archive (Siebel_JCA.rar) using the jar.exe utility provided with your Java Virtual Machine.

Java Data Bean/Siebel Resource Adapter and the siebel.properties File

The siebel.properties file, which is located in your classpath, can be used to provide default parameters for client applications connecting to Siebel application using the Java Data Bean.

Table 29 shows the properties in the siebel.properties file.

Table 29. Properties in the siebel.properties File

Property Type	Property	Description
Request timeout	siebel.conmgr.txtimeout	Default is 2700 seconds. Indicates the transaction timeout in seconds on the server side.
Poolsize	siebel.conmgr.poolsize	Default is 2 with maximum of 500. Indicates the connection pool size. Connection pool maintains a set of connections to a specific server process.
Session timeout	siebel.conmgr.sesstimeout	Default is 600 seconds. Indicates the transaction timeout in seconds on the client side.
Encryption	siebel.conmgr.jce	Indicates the usage of Java Cryptography Extension. You can set this to 1 for jce usage and 0 for nonusage.
Connect String	siebel.connection.string	Specifies the Siebel connection string.
User Name	siebel.user.name	Specifies the user name to be used for logging in to Object Manager.
Password	siebel.user.password	Specifies the password to be used for logging in to Object Manager.
Language	siebel.user.language	Specifies the user's preferred language.
Boolean	siebel.user.encrypted	Specifies whether the username and the password is encrypted with com.siebel.extra.MangleString.
String	siebel.jdb.classname	Specifies the name of the Siebel DataBean to use. For example, you may want to use a bean wrapper such as com.siebel.data.wrapper.SiebelDataBean RetryWrapper class which is a SDB wrapper for retries.
Java System Properties	file.encoding	Indicates the code page on the client side, such as cp1252, utf8, unicodeBig, or cp942.

NOTE: Java System Properties are System Properties, not Siebel Properties.

Here is a sample siebel.properties file:

```
siebel.connection.string  = siebel.tcpip.rsa.none://
test.siebel.com/siebel/sseobjmgr_enu/test

siebel.user.name          = User1

siebel.user.password      = password

siebel.user.language      = enu

siebel.user.encrypted     = false

siebel.conmgr.txtimeout   = 3600

siebel.conmgr.poolsize    = 5

siebel.conmgr.sesstimeout = 300000

siebel.conmgr.retry       = 5

siebel.conmgr.jce         = 1
```

Using Java Code Generated by Siebel eBusiness Applications

The following two sections contain code samples for both managed and non-managed environments.

Managed Code Sample

The following is a code sample using the Siebel Resource Adapter in a managed environment. The SiebelAccountBean class, which is described in this section, provides code examples that show how to perform the following tasks:

- Configure the Siebel Resource Adapter by invoking methods on the Siebel_Account_BusServAdapter class.
- Query for a list of Accounts that start with the letters *Ai*.
- Convert the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information, see "The Siebel JavaBean Wizard" on page 212 for both business services and integration objects. By using the code generation facilities, many of the complexities of the J2EE Connector specification (Interactions, ConnectionSpec, CCI) have been abstracted and encapsulated into a consistent, JavaBean-based interface.

```
/* import declaration here */

import com.siebel.service.jca.siebelaccount.Siebel_AccountBusServAdapter;

import com.siebel.service.jca.siebelaccount.QueryByExampleInput;

import com.siebel.service.jca.siebelaccount.QueryByExampleOutput;

import com.siebel.local.accountinterface.*;


import com.siebel.local.contactinterface.Account_InterfaceIO;

import com.siebel.local.contactinterface.AccountIC;


import com.siebel.integration.jca.cci.SiebelConnectionFactory;

import com.siebel.data.SiebelException;

/* ... */

..."


  /* Get account list using query parameter*/

  public Vector getAccountList(String queryParam) throws
RemoteException {

  QueryByExampleInput   qbeIn  = new QueryByExampleInput();

  QueryByExampleOutput  qbeOut = new QueryByExampleOutput();

  Account_InterfaceIO  acctIO  = new Account_InterfaceIO();

  AccountIC             acctIC = new AccountIC();

  Vector                ioc    = null;


  /* query on accounts based upon the inputs provided */
  acctIC.setfName(queryParam);

  acctIO.addfintObjInst(acctIC);


  qbeIn.setfSiebelMessage(acctIO);


  try {

      /* establish connection  */

      getConnectionFactory(jndiName);


      /* initialize business service using default constructor */

      Siebel_AccountBusServAdapter svc = new
Siebel_AccountBusServAdapter();


      /* set connection specific user properties. if not
specified, values specified in siebel.properties will be used. */

      svc.setConnectString(CONNECT_STRING);

      svc.setUserName(USER_NAME);

      svc.setPassword(PASSWORD);

      svc.setLanguage(LANGUAGE);


      /* set the connection factory to be used for the request. */

      svc.setConnectionFactory(m_cf);


      /* invoke QueryByExample method */

      qbeOut = svc.mQueryByExample(qbeIn);
      acctIO = null;


      /* process output and return to client as a Vector */

      acctIO = new
Sample_AccountIO(qbeOut.getfSiebelMessage().toPropertySet());

      ioc = acctIO.getfintObjInst();


      return ioc;

     }catch(Exception e) {

        return ioc;

     }/* try-catch */

   }/* getAccountList(...) */


   /* function to obtain a connectionFactory reference */

   private void getConnectionFactory(String name) throws
Exception {

   SiebelConnectionFactory   connectionFactory  = null;

   InitialContext            context            = null;

   Object                    reference          = null;


   try {

        context = new InitialContext();

       }catch(Exception e) {

        System.out.println("Could not acquire the initial
context");

      }/* try-catch */
   try {
      reference  = context.lookup(name);
      connectionFactory =
(SiebelConnectionFactory)PortableRemote
      Object.narrow(reference, ConnectionFactory.class);
      }catch(Exception e) {
      System.out.println("Unable to obtain intial reference to
      connection factory");
      }/* try-catch */

      m_ctx = context;
      m_cf  = connectionFactory;
   }/* getConnectionFactory(...) */

   public javax.ejb.SessionContext getSessionContext() { return
mySessionCtx; }

   public void setSessionContext(javax.ejb.SessionContext ctx) {
mySessionCtx =  ctx; }

   public void ejbActivate() {  }

   public void ejbCreate() throws javax.ejb.CreateException {  }

      public void ejbPassivate() {  }
```

```
        public void ejbRemove() {   }


        protected SiebelConnectionFactory  m_cf          = null;

        protected InitialContext           m_ctx         = null;

        private javax.ejb.SessionContext   mySessionCtx  = null;

  }/* public class SiebelBean implements SessionBean */
```

Non-Managed Code Sample

The following is a code sample using the Siebel Resource Adapter in a non-managed environment. The JCASiebelAccount class (implemented below) provides code examples that show how to perform the following tasks:

- Configure the Siebel Resource Adapter by invoking methods on the Siebel_Account_BusServAdapter class.
- Query for a list of Accounts that start with the letters *Ai*.
- Convert the result into a vector.

The following code uses the code generation facilities provided in Siebel Tools. For more information, see "The Siebel JavaBean Wizard" on page 212 for both business services and integration objects. By using the code generation facilities, many of the complexities of the J2EE Connector specification (Interactions, ConnectionSpec, CCI) have been abstracted and encapsulated into a consistent, JavaBean-based interface.

```
package com.siebel.service.jca.siebelaccount;


/* import required classes here */


public class JCASiebelAccount {

public static void main(String[] args) {
```

```
Siebel_AccountBusServAdapter svc = new

Siebel_AccountBusServAdapter("siebel.properties");

QueryByExampleInput  qbeIn  = new QueryByExampleInput();

QueryByExampleOutput  qbeOut = new QueryByExampleOutput();

AccountIC  acctIC = new AccountIC();

Account_InterfaceIO  acctIO  = new Account_InterfaceIO();

SiebelConnectionFactory  m_cf = null;

Vector     ioc = null;


/* query on accounts that start with 'Ai'. */

acctIC.setfName("Ai*");

acctIO.addfintObjInst(acctIC);

qbeIn.setfSiebelMessage(acctIO);

try {

   /* establish connection factory and connection */

   m_cf   = new SiebelNoTxConnectionFactory(new
   SiebelNoTxManagedConnectionFactory());

   svc.setConnectionFactory(m_cf);

   /* invoking QueryById method */

   qbeOut = svc.mQueryByExample(qbeIn);

   acctIO = null;

   acctIC = null;

   acctIO = new
   Sample_AccountIO(qbeOut.getfSiebelMessage().toPropertySet());

   ioc    = acctIO.getfintObjInst();

   /* loop through all accounts returned */
```

```
            if(!ioc.isEmpty()) {

               for(int i=0; i < ioc.size(); i++) {

                 acctIC = ((AccountIC)ioc.get(i));

                  System.out.println(i + "  " + acctIC.getfName() + " "
               +acctIC.getfLocation());

               }/* for */

            }/* if */

            }catch(SiebelException se) {

            System.out.println(se.getDetailedMessage());

            }/* try-catch */

         }/* main */

         }/* public class JCASiebelAccount */
```

Siebel Java/XML Framework

To further simplify client-side integration, Siebel applications provide a Java Framework designed to receive XML requests sent by a Siebel application over HTTP or MQSeries. The Java/XML Framework provides a uniform way to receive and process Siebel application requests within a J2EE environment.

CAUTION: You may use this code only to receive XML requests from Siebel applications. You may extend this code solely for use in object code form and solely for the purpose of integrating Siebel applications with non-Siebel applications. However, any modification or extension of this code is outside of the scope of Maintenance Services and voids all applicable warranties.

Requests initiated from within Siebel applications are transmitted to the appropriate J2EE Application Server using Siebel's eAI integration infrastructure, as illustrated in Figure 15.

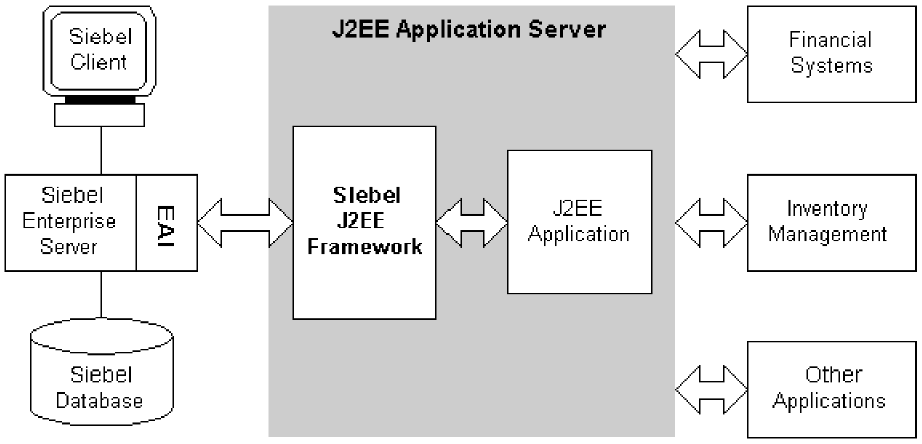


Figure 15. JavaBean Architecture

The Java/XML Framework consists of a Servlet to receive HTTP requests and an MQSeries Base Server designed to retrieve messages from an MQSeries queue. When implementing the Java/XML Framework, you need to implement a single interface (ProcessRequest) responsible for understanding the contents of the incoming request and dispatching it to the appropriate Java component.

During application design, you can choose the transport that best achieves your integration requirements, as well as the structure of the XML document to be sent to the J2EE Application Server.

The Java/XML Framework is installed with Siebel Tools as part of the Siebel Java Integration Component (Custom installation). The HTTP Servlet and MQSeries Base Server are installed in the `*Siebel Tools*\CLASSES` directory, and are packaged in the SiebelJI.jar file.

Sample implementations of the Siebel HTTP Base Servlet and the MQSeries Base Server may be found in these directories:

```
Siebel Tools\CLASSES\SRC\COM\SIEBEL\INTEGRATION\MQ
```

or

```
Siebel Tools\CLASSES\SRC\COM\SIEBEL\INTEGRATION\SERVLET
```

Setting Up Outbound Integrations

To communicate with an external application from Siebel applications and receive messages back, you implement a procedure that uses both an existing transport and Siebel JavaBeans within a workflow.

To send and receive data to and from an external application

1. Start the Siebel client and create a new workflow process.
2. Edit the workflow so that the Siebel application initiates a request to a J2EE application server by sending out an XML message.
   - **a** Select the transport adapter to use, either HTTP or MQSeries Server Transport (depending upon the target transport protocol)
   - **b** Select the Synchronous or Asynchronous method:
     - ❑ Use the SendReceive method to require a response in XML format to your request from the external application.
     - ❑ Use the Send method to send a simple request that requires no feedback from the external application.

The J2EE application server receives messages by extending a Siebel Servlet or Server, depending on the transport mechanism (HTTP or MQSeries). The Siebel Servlet or Server provides all the necessary code required to receive and pre-process a request. A processRequest method (that need to be implemented) is invoked to allow the service provider to process the request.

The processRequest method takes a single Input XML string and returns an Output XML string. The process method can only throw a process exception upon exception. The processRequest method is defined as:

```
public String processRequest(Object request, String inputXML)
throws ProcessException
```

Index