BEA WebLogic Adapter for Siebel®

User Guide

Release 7.0
Document Date: December 2002

BEA WebLogic Adapter for Siebel

Part Number	Date	Software Version
N/A	December 2002	7.0

Contents

7. Using Siebel Business Components and Siebel Business Services

A. Sample Files

B. Creating Siebel Workflows

About This Document

The *BEA WebLogic Adapter for Siebel User Guide* is organized as follows:

- Chapter 1, "Introducing the BEA WebLogic Adapter for Siebel," explains how to execute Siebel functions, access data stored in Siebel, and use the BEA WebLogic Adapter for Siebel with the BEA Application Explorer. This section also outlines Siebel EAI architecture.
- Chapter 2, "Using Siebel Integration Objects," describes the processing of Siebel Integration Objects using Siebel XML.
- Chapter 3, "Creating Schemas for Siebel Integration Objects," explains how to create the BEA event and service schema definitions for an Integration Object, when used in conjunction with the Siebel XML access method.
- Chapter 4, "Creating and Editing Application Views," explains how to create an Application View.
- Chapter 5, "Adding Application View Events for Siebel Integration Objects," explains how to add events to an application view for the MQ, File, and HTTP transports. Instructions for adding and testing the events are provided.
- Chapter 6, "Adding Application View Services for Siebel Integration Objects," explains how to add services to an application view for the MQ, File, and HTTP transports. Instructions for adding and testing the services are provided.
- Chapter 7, "Using Siebel Business Components and Siebel Business Services," explains how the BEA WebLogic Adapter for Siebel enables the processing of Siebel Business Services and Business Components using the Siebel Java Data Bean and Siebel Data COM Interface.
- Appendix A, "Sample Files," provides sample schemas for Siebel Business Components and Siebel Business Services.
- Appendix B, "Creating Siebel Workflows," provides sample Siebel Workflows.

What You Need to Know

This document is written for system integrators who develop client-server interfaces between Siebel and other applications.The BEA WebLogic Adapter for Siebel provides a means to exchange real-time business data between Siebel systems and other application, database, or external business partner systems. The adapter allows for inbound and outbound processing with Siebel. It is assumed that readers have:

- General knowledge of the Siebel environment, including Siebel Server, Siebel Client, Siebel Tools, and how to configure Siebel Server tasks.
- General knowledge of Siebel EAI concepts including how to use Siebel Tools and Wizards to create and modify Siebel Business Services and Integration Components.
- Specific knowledge of Siebel business applications.
- Knowledge of Siebel processes and data models for the required application area.
- General knowledge of WebLogic Integration architecture.
- General knowledge of client-server concepts.

Related Information

The following documents provide additional information for the associated software components:

- *BEA WebLogic Adapter for Siebel Installation and Configuration Guide*
- *BEA WebLogic Adapter for Siebel Release Notes*
- *BEA Application Explorer Installation and Configuration Guide*

- BEA WebLogic Server installation and user documentation, which is available at the following URL:

  `http://edocs.bea.com/more_wls.html`

- BEA WebLogic Integration installation and user documentation, which is available at the following URL:

  `http://edocs.bea.com/more_wli.html`

- Siebel eBusiness Bookshelf Version 6.3. or higher. Applicable topics include:
  - Overview: Siebel eBusiness Application Integration Volume I
  - Integration Platform Technologies: Siebel eBusiness Volume II
  - Transports and Interfaces: Siebel eBusiness Application Volume III
  - Business Processes and Rules: Siebel eBusiness Application Integration Volume IV
  - Tools Guide
  - Server Administration Guide
  - Workflow Administration Guide
- Siebel eBusiness Bookshelf Version 6.0.1. Applicable topics include:
  - Siebel eBusiness Application Integration: Siebel EAI Architecture, Using Siebel Business Services, Using Siebel EAI Adapters & Transports, Creating Business Services in Siebel
  - Siebel Object Interfaces Reference
  - Siebel Tools Guide
  - Server Administration Guide
  - Siebel Workflow Guide
- BEA Application Explorer documentation

Contact Us!

Your feedback on the BEA WebLogic Adapter for Siebel documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA WebLogic Adapter for Siebel documentation.

In your e-mail message, please indicate which version of the *BEA WebLogic Adapter for Siebel User Guide* you are using.

If you have any questions about this release of the BEA WebLogic Adapter for Siebel, or if you have problems installing and running the adapter, contact BEA Customer Support through BEA eSupport at `http://support.bea.com`. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the adapter you are using
- The version of WebLogic Integration you are using
- A description of the problem and the content of pertinent error messages.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introducing the BEA WebLogic Adapter for Siebel

This section explains how the BEA WebLogic Adapter for Siebel integrates with Siebel. This section also describes the Siebel architecture. It includes the following topics:

- Features of the BEA WebLogic Adapter for Siebel
- The Siebel Application Model
- Integrating With Siebel
- Siebel EAI Architecture
- Using the BEA Application Explorer with the BEA WebLogic Adapter for Siebel

Features of the BEA WebLogic Adapter for Siebel

The BEA WebLogic Adapter for Siebel provides a means to exchange real-time business data between Siebel systems and other application, database, or external business partner systems. The adapter enables external applications for inbound and outbound processing with Siebel.

The adapter uses WebLogic Integration and XML messages to allow non-Siebel applications to communicate and exchange transactions with Siebel.

- Applications use this capability if they require access to Siebel data only when a Siebel business event occurs in combination with WebLogic Integration application views, events, and BEA business process management workflow to receive messages from Siebel.
- Applications use this capability when they require a Siebel business event in combination with WebLogic Integration application views, services, and BEA business process management workflow to send request messages to Siebel.

  If the request is for retrieving data from Siebel, then the adapter sends the application a response message with the data.

The BEA WebLogic Adapter for Siebel:

- Supports synchronous and asynchronous, bi-directional message interactions for Siebel Business Services, Business Components, and Integration Objects.
- Provides the BEA Application Explorer, a GUI tool which uses Siebel object manager metadata to build XML schemas for application views, events, and services.
- Integrates service and event operations with Siebel.
- Supports Siebel Transports—MQSeries, File, and HTTP.

The Siebel Application Model

The Siebel Enterprise application defines a data abstraction layer that removes dependencies on the underlying database. It accomplishes this by using intermediate business components and objects that represent database structures. A business component typically represents a table in a database. A business object is a group of related business components. Each business component can have relationships with other business components. A relationship can be a parent/child relationship (Multi-Valued Link Field or one-to-many) or an association relationship (many-to-many).

From a given business component, you can “walk along” the relationships defined for that component to another component. The path you use to traverse component relationships is called the navigation path. For example, if you want to obtain all addresses for a particular account, you can traverse the parent/child relationship between Account and Address to obtain those addresses. By using navigation paths, you can traverse nearly all of the business component relationships defined in the Siebel system.

You can navigate from a top-level business component to another related component. This defines the navigation path taken to reach the selected component. All operations performed by the adapter traverse this path prior to performing the selected operation. For example, to select account addresses, select Account as the business object in the tree view. From there, navigate to the Address sub-component by expanding the Account view and selecting the Business Address Multi-Valued Link. By choosing this navigation path, you perform an operation on the Address component for a particular Account.

In Siebel, Integration Objects represent Siebel Objects to external applications. They are based on Siebel Business Objects. They also can be created to describe external tables and views. They enable Siebel data structures and external data structures to be instantiated in a string format that Siebel Business Services can handle.

Integrating With Siebel

You can use the BEA WebLogic Adapter for Siebel to invoke a Siebel business process, such as add/update account, or you can use the adapter as part of an integration effort to connect Siebel and non-Siebel systems. The adapter is bi-directional and can detect an event in Siebel by receiving a Siebel XML document. Alternatively, the adapter can cause a Siebel business event by:

- Using the Siebel Java Data Bean.
- Using the Siebel COM Data Interface.
- Passing a Siebel XML document to execute an instance of a Siebel object and its method. To send the XML document to Siebel, the adapter uses one of the Siebel Transports—MQSeries, File, or HTTP.

When integrating with Siebel using Siebel XML, the BEA WebLogic Adapter for Siebel application developer must create Siebel Integration Objects for Siebel Business Objects that are to be used as part of a Siebel Workflow. The workflow is responsible for processing inbound or outbound Siebel XML and utilizes various transports such as MQSeries, File, and HTTP to exchange transactions with external systems. The Siebel Workflow is typically created by the Siebel administrator or developer using Siebel Client for Siebel Workflow Administration screens.

When integrating with Siebel directly using the Java Data Bean or COM Data Interface, the BEA WebLogic Adapter for Siebel does not require a Siebel Workflow. Instead, it executes Siebel Business Services and Siebel Business Components directly.

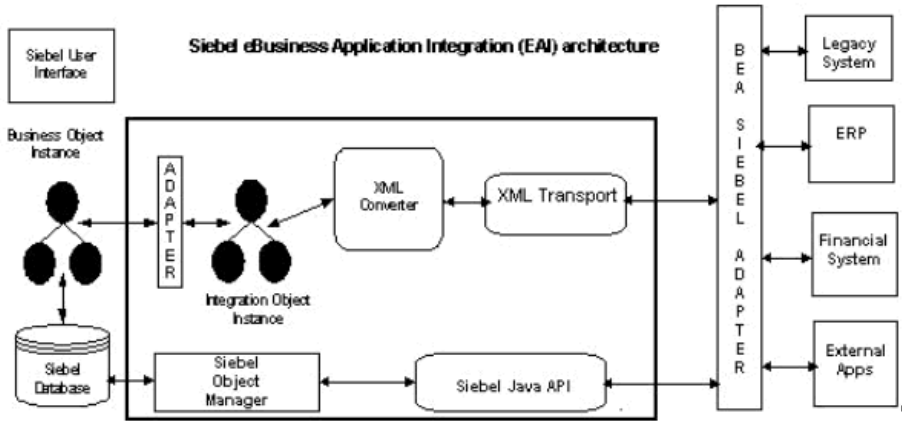
Siebel EAI Architecture

Siebel provides for integration with other applications and systems using its Siebel EAI framework and its Business Integration Manager facility. The BEA WebLogic Adapter for Siebel uses the Siebel EAI framework and leverages various integration access methods to provide the greatest amount of flexibility and functionality while working within the Siebel framework.

The BEA WebLogic Adapter for Siebel supports the following integration access methods:

- Siebel Java Data Bean for services involving Siebel Business Components or Siebel Business Services.
- Siebel COM Data Interface for services involving Siebel Business Components or Siebel Business Services.
- Siebel XML for events and services involving Siebel Integration Objects.

Figure 1-1 Siebel eBusiness Application Integration (EAI) Architecture



Using the BEA Application Explorer with the BEA WebLogic Adapter for Siebel

The BEA Application Explorer uses an explorer metaphor for browsing the Siebel system for Business Services, Business Objects, Business Components, and Integration Objects. The explorer enables the user to create service and event schemas for the associated object. When running a service using either Siebel Business Components, Business Services, or Integration Objects, you use the explorer to create BEA request and response schemas. Depending on the Siebel system release, the explorer uses either the Siebel Java Data Bean or the Siebel COM Data Interface when creating BEA schemas.

When running an event or service for Siebel Integration Objects, you use the explorer to create BEA schema definitions for an Integration Object associated with a Siebel Workflow. The Siebel Workflow is created by a Siebel developer or administrator using Siebel Client and Siebel Workflow Administration.

The steps required to create BEA schemas for events and services are illustrated in the following sections:

- Chapter 3, “Creating Schemas for Siebel Integration Objects.”
- Chapter 7, “Using Siebel Business Components and Siebel Business Services.”

CHAPTER

2 Using Siebel Integration Objects

This section describes the processing of Siebel integration objects using Siebel XML. It includes the following topics:

Overview

The BEA WebLogic Adapter for Siebel supports access to Siebel integration objects by using Siebel XML to handle both services and events. The adapter also supports the direct invocation of Siebel Business Services and business components using the Siebel Java Data Bean or the Siebel COM Data Interface. This section describes the processing of Siebel integration objects using Siebel XML.

When using Siebel XML to integrate with Siebel Integration Objects, the interface uses a Siebel Workflow. A Siebel Workflow is defined within Siebel either to emit Siebel XML in the case of an event, or to receive incoming Siebel XML in the case of a service. In either case, emitting or receiving is handled by Siebel transport services (for MQSeries, File, or HTTP). Other sections discuss the use and creation of workflows that employ the supported transport services. The creation of the workflows required for an event and service, as well as for each of the possible transports (MQSeries, File, or HTTP), is illustrated.

Siebel Workflows

A Siebel Workflow is a series of Siebel Business Services linked together to accomplish a business task. You create workflows using the Siebel Client Workflow Administration windows that are invoked through one of the following methods:

- Using a workflow policy
- Using a run-time event (Siebel Event)
- Using a script (eScript or Siebel VB)

The following topic briefly describes how to invoke the workflow through a policy condition. For more information on policy and other methods, see the *Siebel Bookshelf* documentation.

Using a Policy to Invoke a Siebel EAI Workflow

A workflow policy is defined by a set of conditions that executes a set of defined actions. A Siebel Workflow policy consists of:

- Conditions that define circumstances, based on changes in the state of a Siebel database.
- Actions that define steps taken when conditions are fulfilled.

Creating a policy to invoke a workflow as an action involves the following steps:

1. Define an action to be executed after a policy is triggered. Use the Run Integration Process program.
2. Create a policy by setting conditions and selecting appropriate policy groups and actions.
3. Activate the policy by choosing an activation date.
4. Run Generate Triggers server task from Server Administration windows to set the conditions to be monitored.
5. Start the Workflow Monitor agent after editing with the appropriate policy group (to which your policy belongs) to evaluate whether to perform an action.
6. Start Workflow Action Agent server task from Server Administration windows to perform the action.

For more information on the above steps, see the *Siebel Bookshelf* documentation.

Events

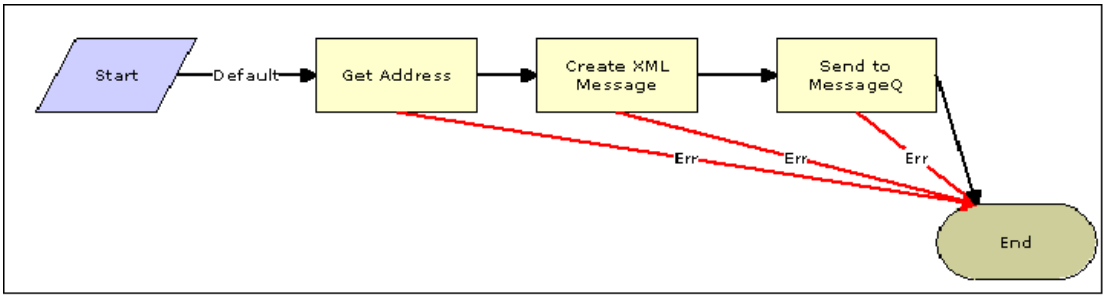
In the case of an event, a Siebel Workflow based on a Siebel policy, run-time, or script (eScript or Siebel VB) for a Siebel business event, the end result is the generation of a Siebel XML document. The document is placed on one of the Siebel transports. For example, when you add a new account in the Siebel Call Center application, you can design and configure a workflow to be triggered on the account transaction. You can design the workflow to extract the data for the new record, convert it to Siebel XML, and then place it on an IBM MQSeries message queue.

In this example, the Siebel Workflow process executes a series of Siebel Business Services as follows:

1. Calls the Siebel EAI Siebel Adapter, which queries for the newly updated account record, and places the data in its original internal structure into memory.
2. Calls the Siebel EAI XML Converter, which converts the data into an XML message.
3. Calls the Siebel EAI MQSeries Transport, which places the newly created XML message into the appropriate MQSeries message queue.

   After the message is placed in the message queue, it is retrieved by the BEA WebLogic Adapter for Siebel 6.3 and higher. The above steps are illustrated by the following workflow sequence.

Figure 2-1 Workflow Sequence



Services

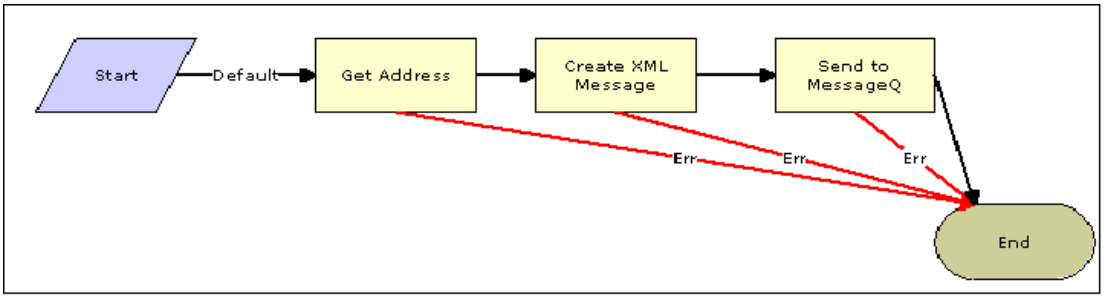
A Siebel Workflow that is triggered by an external event through a service request begins by receiving a Siebel XML document placed on one of its transports. The end result might be the update of a Siebel record using the XML as input, for example, when a new account is added in another CRM system but also must be updated in the Siebel Call Center application. You can design and configure a workflow to receive or listen on an MQSeries message queue. Upon receipt of the XML message, the workflow processes the transaction into the Siebel system to update the record.

In this example, upon receipt of the Siebel XML message in the message queue, the Siebel MQ Series Receiver server task initiates a Siebel Workflow process, which in turn executes a series of Siebel Business Services as follows:

1. Calls the Siebel EAI XML Converter, which converts the XML message into Siebel internal format.
2. Calls the Siebel EAI Siebel Adapter, which applies the newly updated account record based on the methods defined in its service.

The following is a sample of the workflow process:

Figure 2-2 Workflow Process



Next Steps

The following sections provide the information you need create WebLogic Integration application view services and events for the MQ, HTTP, and File transports:

- Chapter 3, “Creating Schemas for Siebel Integration Objects.”
- Chapter 4, “Creating and Editing Application Views.”
- Chapter 5, “Adding Application View Events for Siebel Integration Objects.”
- Chapter 6, “Adding Application View Services for Siebel Integration Objects.”

CHAPTER

3 Creating Schemas for Siebel Integration Objects

This section explains how to create the BEA event and service schema definitions for an integration object when used in conjunction with the Siebel XML access method. It includes the following topics:

- Overview
- Generating a Siebel XDR Schema
- Creating BEA Schemas from Siebel XDR Schema (Siebel XML)

Overview

When running an event or service using Siebel XML, you use the BEA Application Explorer to create the BEA schema definitions for an integration object.You have a choice to create event and service schemas for the integration object.

Generating XDR schemas using the Siebel Tools Schema Wizard is a prerequisite for setting up integration using the BEA Application View Console. After the XDR schema is generated, the BEA Application Explorer uses the XDR file to generate the following:

- Event XML schema

- Service XML request schema
- Service XML response schema

Generating a Siebel XDR Schema

This section illustrates the creation of the Siebel XDR schema. The XDR schema is then used as input to the BEA Application Explorer when browsing integration objects.

Note: For releases prior to Siebel 6.3, the Siebel Tools Schema Wizard only creates DTD schemas. First these must be transformed, manually or by using other tools, into XDR files before the explorer can use them as input for creating WebLogic Integration XSD schemas.

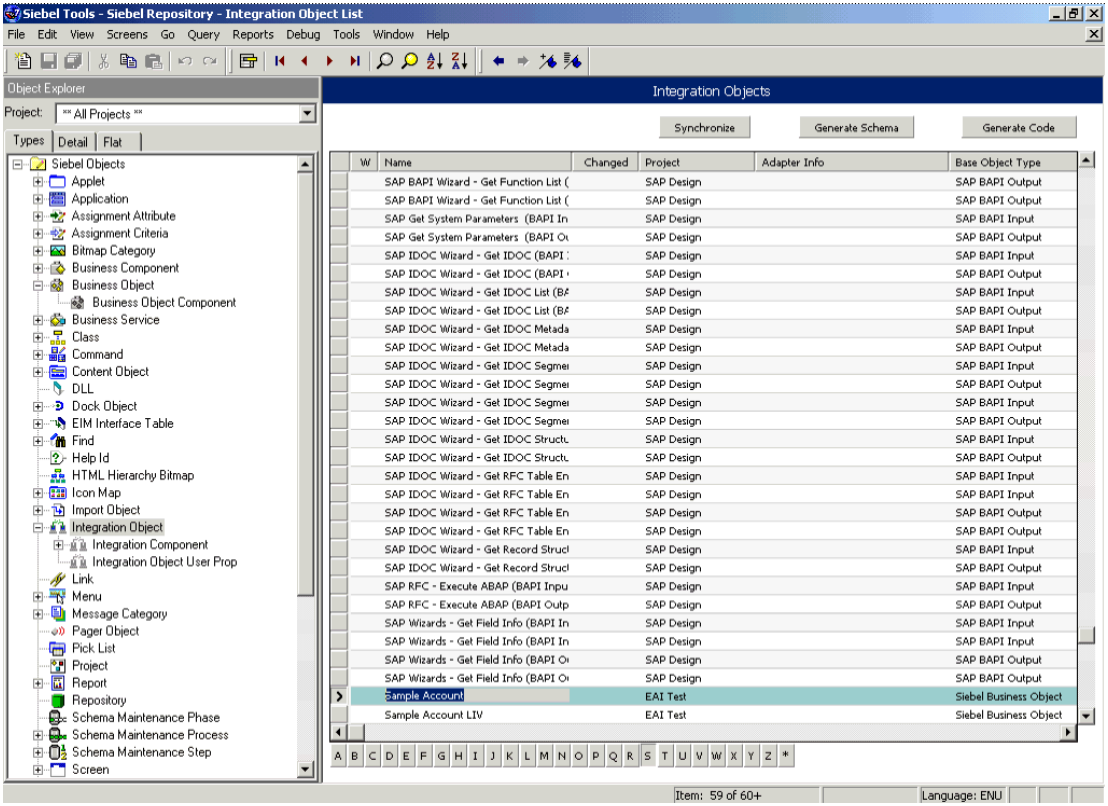
Figure 3-1 Siebel Logon Window

To generate a Siebel XDR schema:

1. Log on to Siebel Tools.

   The Siebel Tools Window opens with Integration Objects appearing in the right pane.

Figure 3-2 Siebel Tools Window - Integration Objects List

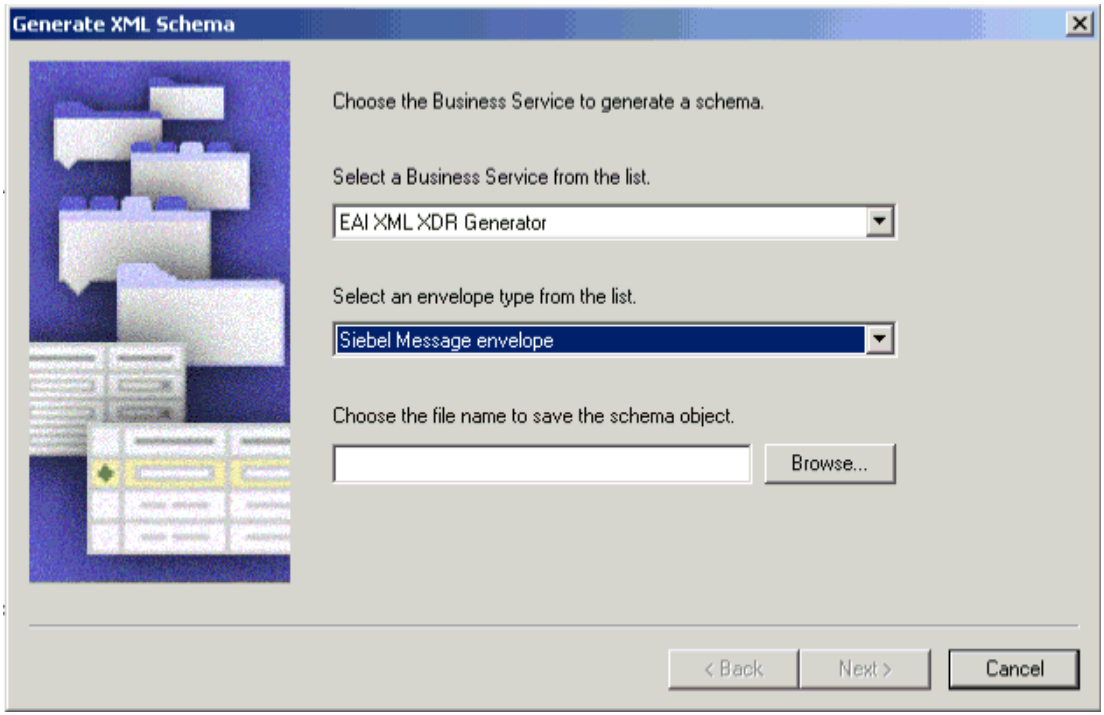


2. Select the integration object, Sample Account in the right pane.

3. Click the Generate Schema button.

   The Generate XML Schema wizard opens.

Figure 3-3 Generate XML Schema Wizard



4. Save the XDR schema file in a directory where it can be accessed by the BEA Application Explorer. For example, enter `C:\BEA\BEASCHEMAS`.

You can now use the BEA Application Explorer to generate BEA schemas as described in "Creating BEA Schemas from Siebel XDR Schema (Siebel XML)" on page 3-5.

Creating BEA Schemas from Siebel XDR Schema (Siebel XML)

After you create the Siebel XDR schema for a selected Siebel integration object, as described in "Generating a Siebel XDR Schema" on page 3-2, you can create the BEA schemas to be published in the WebLogic Integration repository. The BEA Application Explorer generates the appropriate event schema and service request and response schemas. You must supply the BEA Application Explorer with the location of the previously created Siebel XDR schema for the particular integration object.

The following topics illustrate the creation of the BEA schemas from the Siebel XDR schema for the integration object Sample Account.

Open the BEA Application Explorer.

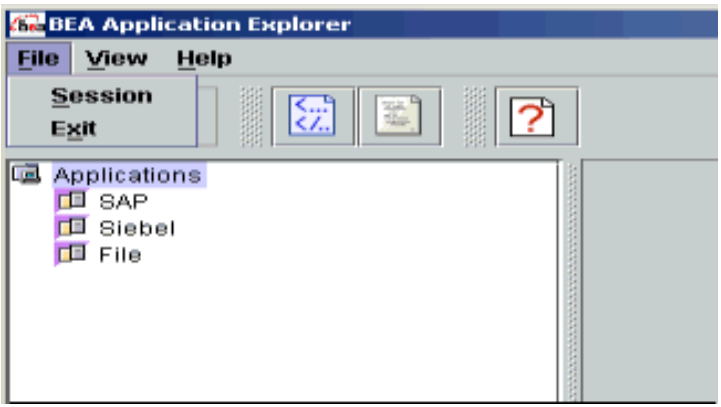
Figure 3-4 BEA Application Explorer Window

When the explorer opens, establish the directory associated with your WebLogic Integration server as described in "Establishing the Working Directory" on page 3-6.

Establishing the Working Directory

After you open the BEA Application Explorer, as described in "Creating BEA Schemas from Siebel XDR Schema (Siebel XML)" on page 3-5, you can establish the directory associated with your WebLogic Integration server to import event and service XML schemas into the application view repository.

Figure 3-5 BEA Application Explorer Window

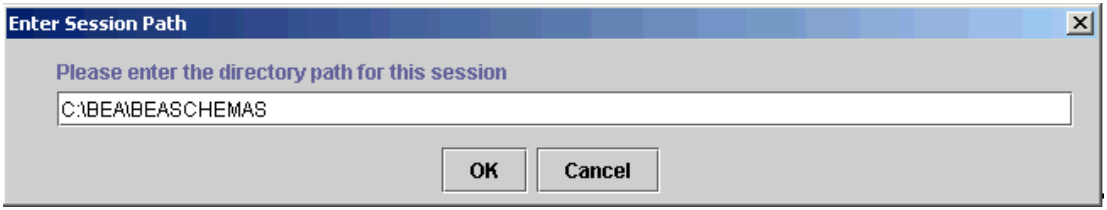


To establish the working directory:

1. From the File menu in the Application Explorer, choose Session.

   The Enter Session Path box appears.

Figure 3-6 Enter Session Path Box



2. Enter a folder name.

In this example, `C:\BEA\BEASCHEMAS` is the BEA Application Explorer working directory. This is where schemas are placed when the BEA Application Explorer generates them.

3. Click OK.

   Notice that the session path appears at the bottom of the explorer window.

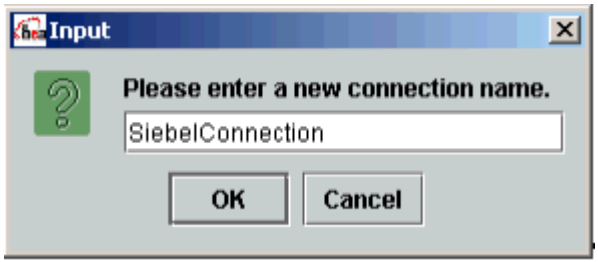
Establishing a Connection to Siebel

To establish a connection to Siebel, open the BEA Application Explorer window.

Figure 3-7 BEA Application Explorer Window

1. Right-click Siebel and choose New Connection from the shortcut menu.

Figure 3-8 Input Connection Name Box

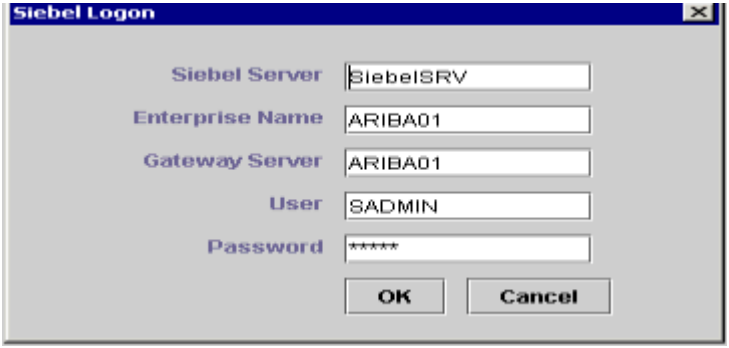


2. Enter a name for the Siebel connection, for example, SiebelConnection.

3. Click OK.

   The Siebel Logon box appears.

Figure 3-9 Siebel Logon Box



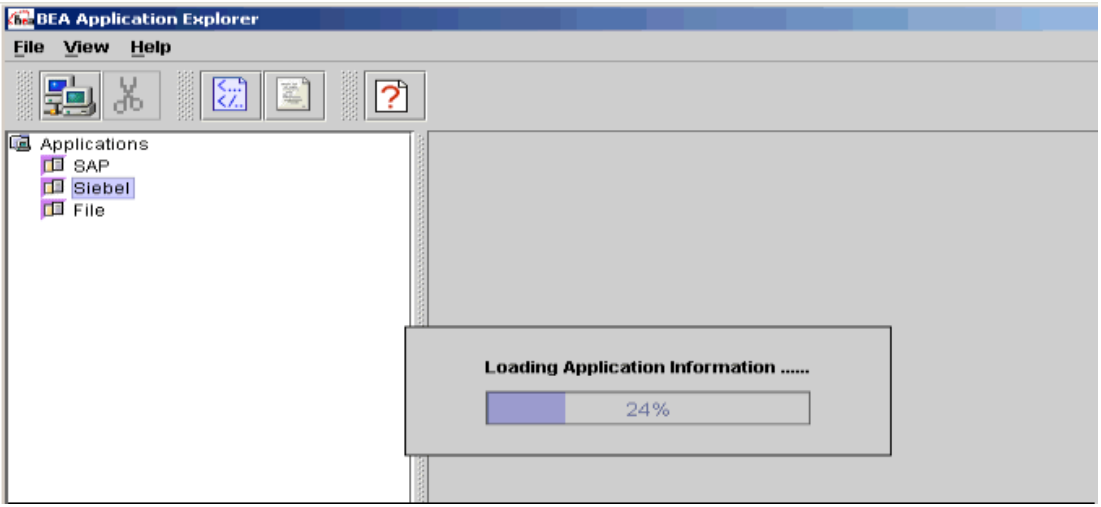
4. Enter the parameters required to establish the connection to the Siebel system:
   - Siebel Server
   - Enterprise Name
   - Gateway Server
   - User
   - Password

   The configuration parameters to be supplied are those used by Siebel Client applications for connecting to the Siebel system. For more information about these parameters, see your *Siebel Bookshelf* or ask your Siebel system administrator.

5. Click OK.

   The Loading Application Information indicator appears with progress information.

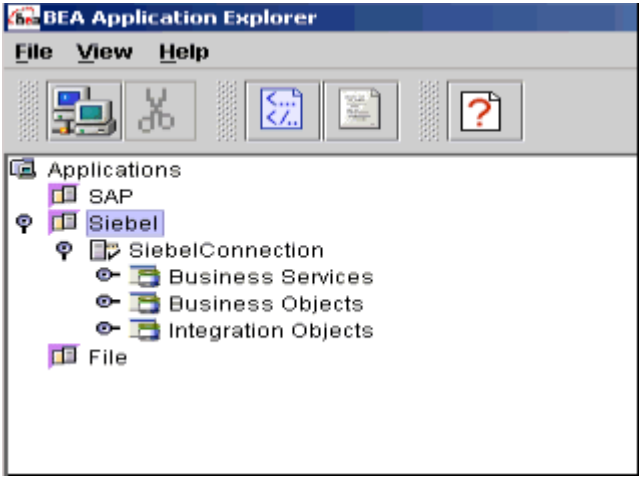
Figure 3-10 BEA Application Explorer - Loading Application Information



The Application Explorer connects to the Siebel system to extract metadata for the business services, business objects, and integration objects.

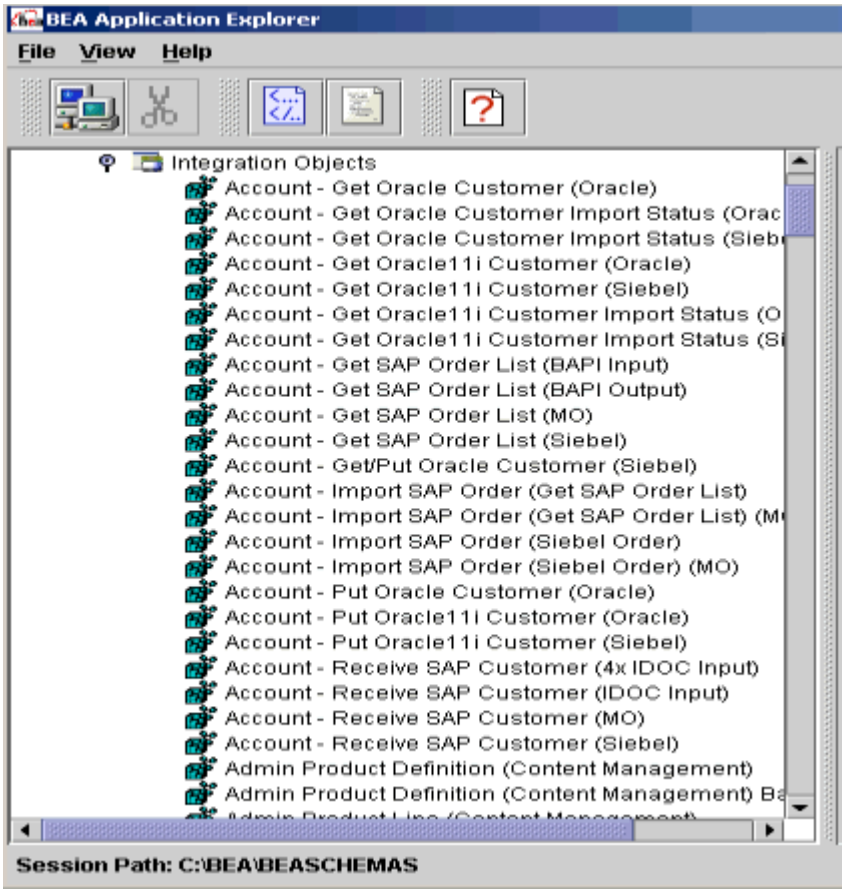
When the connection is complete, you can view business objects and services as well as integration objects.

Figure 3-11 BEA Application Explorer - Business Services, Business Objects, and Integration Objects



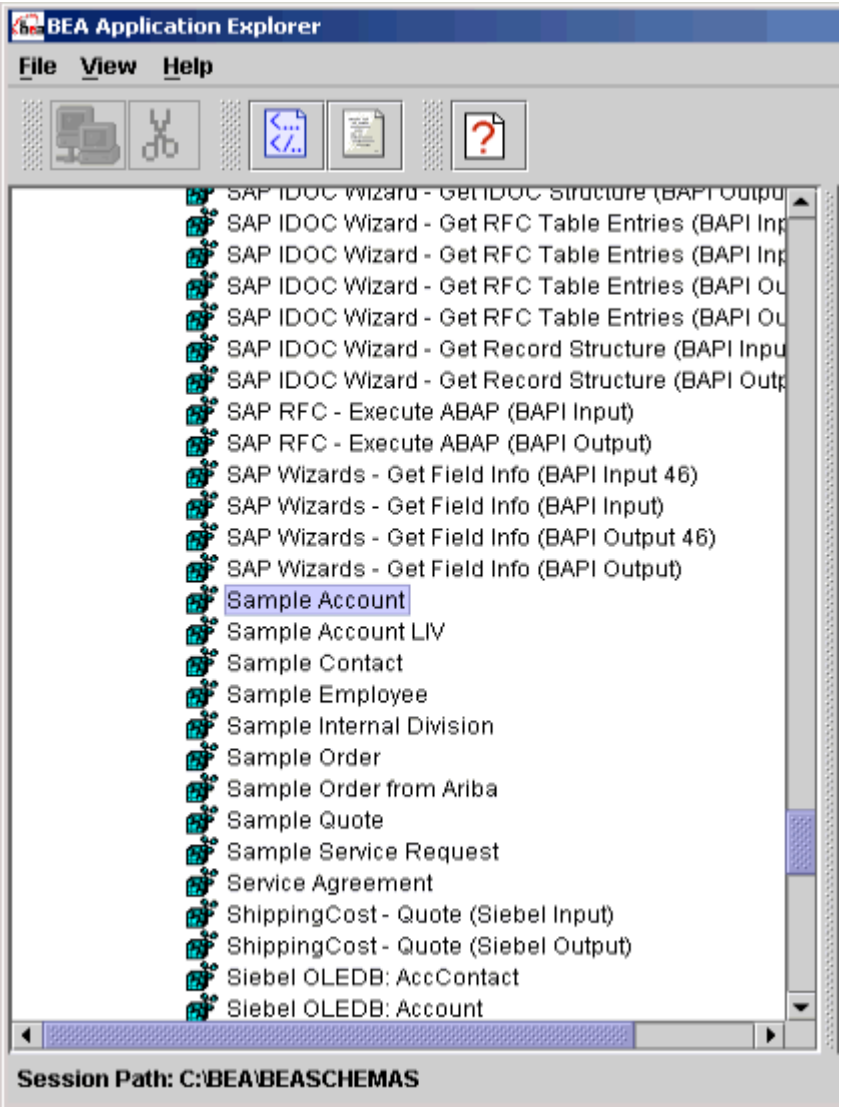
You can browse all available integration objects in the Siebel system.

Figure 3-12 BEA Application Explorer Window - Integration Objects



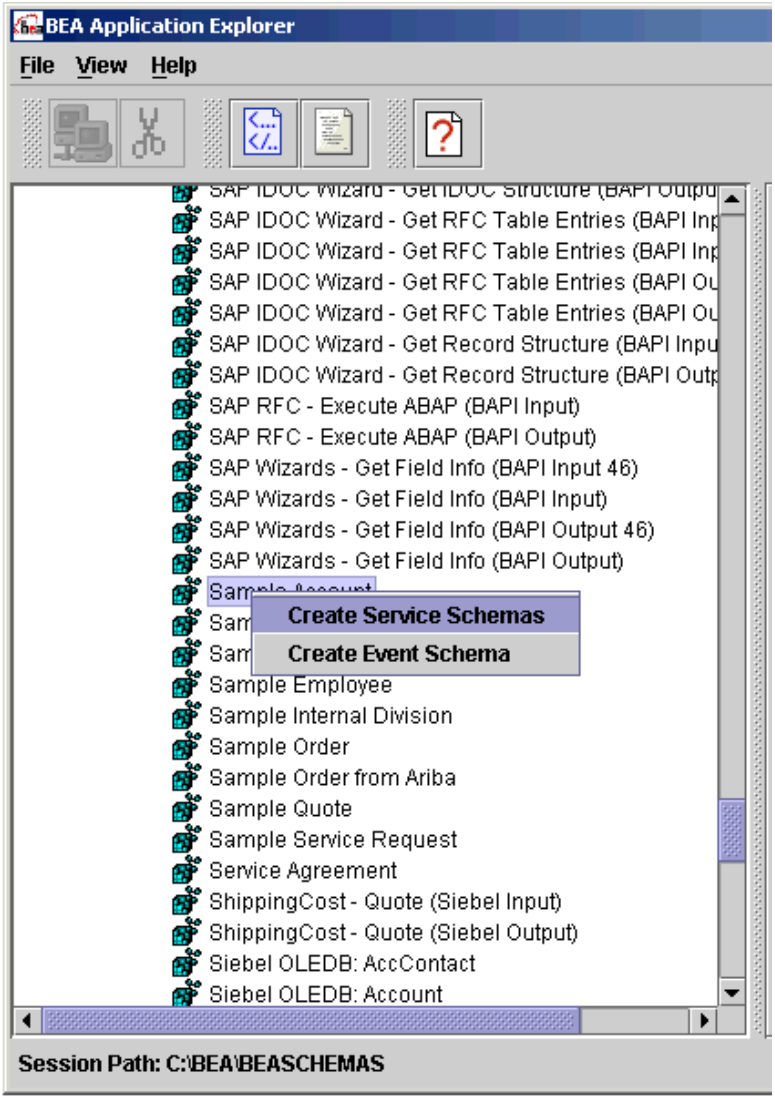
6. Expand Integration Objects.

Figure 3-13 BEA Application Explorer - Sample Account



7. Using the BEA Application Explorer, navigate to the integration object called Sample Account.

Figure 3-14 BEA Application Explorer - Schema Creation



8. After you select a Siebel integration object, such as Sample Account, right-click to generate service request and response schemas.

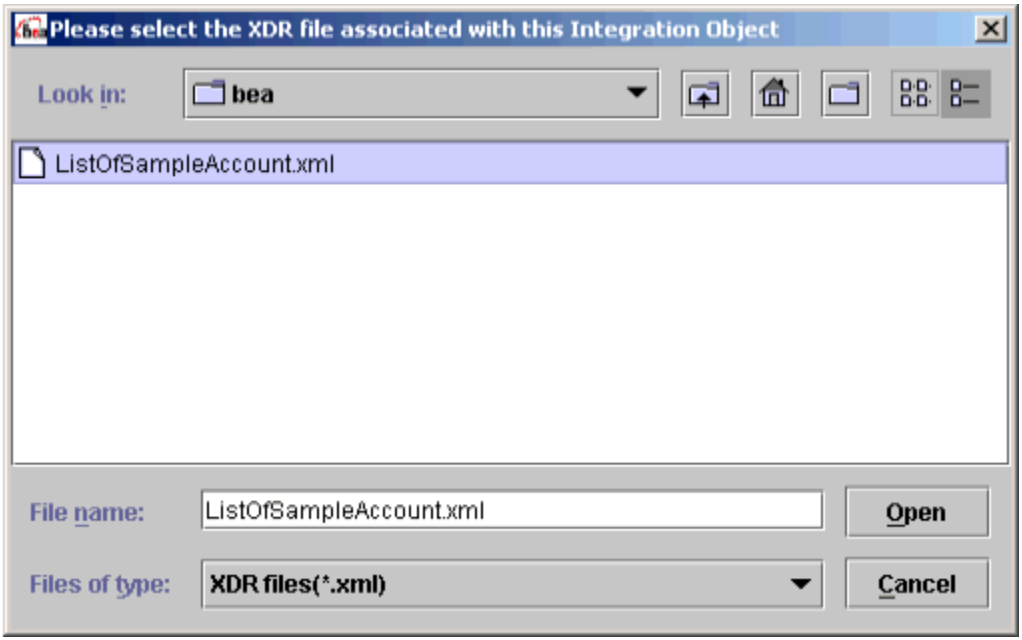
The BEA Application Explorer can generate the following WebLogic Integration schemas:

- Service XML request schema
- Service XML response schema
- Event XML schema

Creating Service Schemas

When you select Create Service Schemas, as described in the preceding section, "Establishing a Connection to Siebel" on page 3-7, a window opens where you enter the location of the previously generated XDR schema.

Figure 3-15 XDR Selection Window

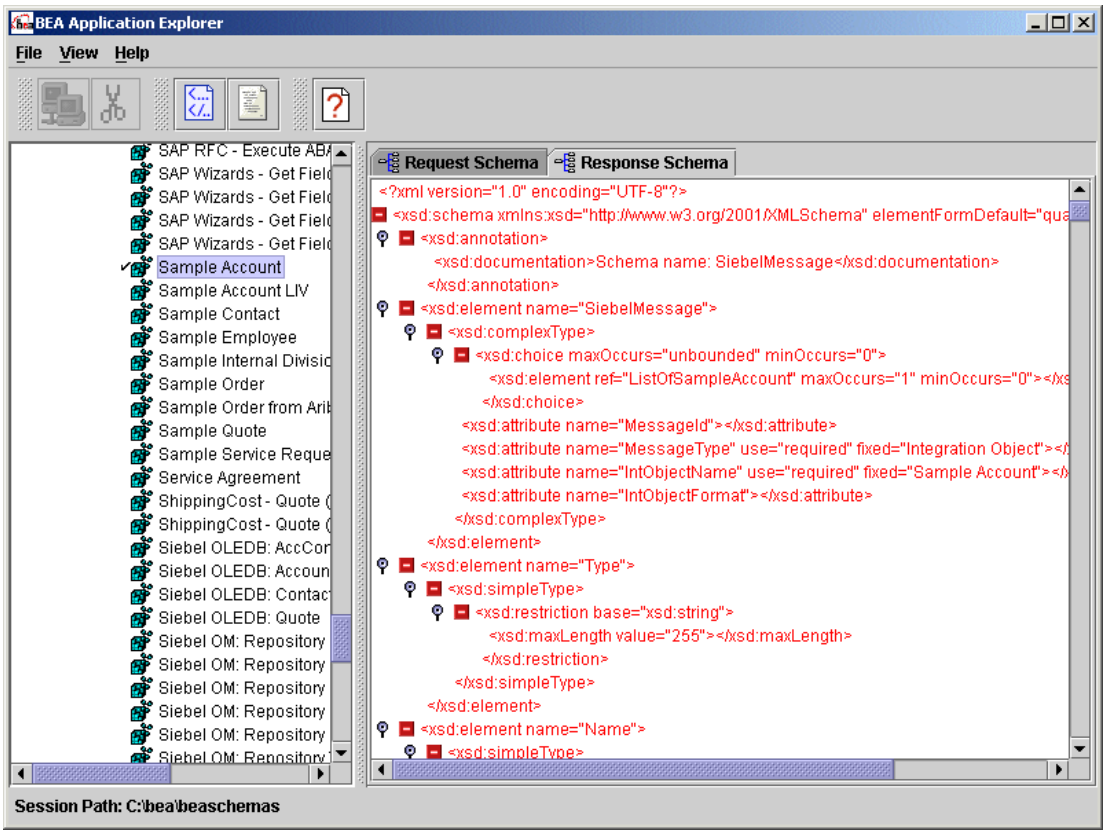


1. Select the file `ListOfSampleAccount.xml` located in `C:\BEA`.

   `ListOfSampleAccount.xml` is the name of the XDR file generated by Siebel for the integration object named Sample Account.

2. Click Open.

Figure 3-16 BEA Application Explorer - Response Schema



A directory structure is created automatically for you within the working directory. In this example, the working directory is `C:\BEA\BEASCHEMAS`.

The explorer creates a folder called `Siebel` under the working directory. It also creates subfolders for each configured Siebel connection to contain the schemas created for each connection. In this case, the schemas created for you are located in the folder called `SiebelConnection`, the connection name you established when you connected to the Siebel system using the explorer.

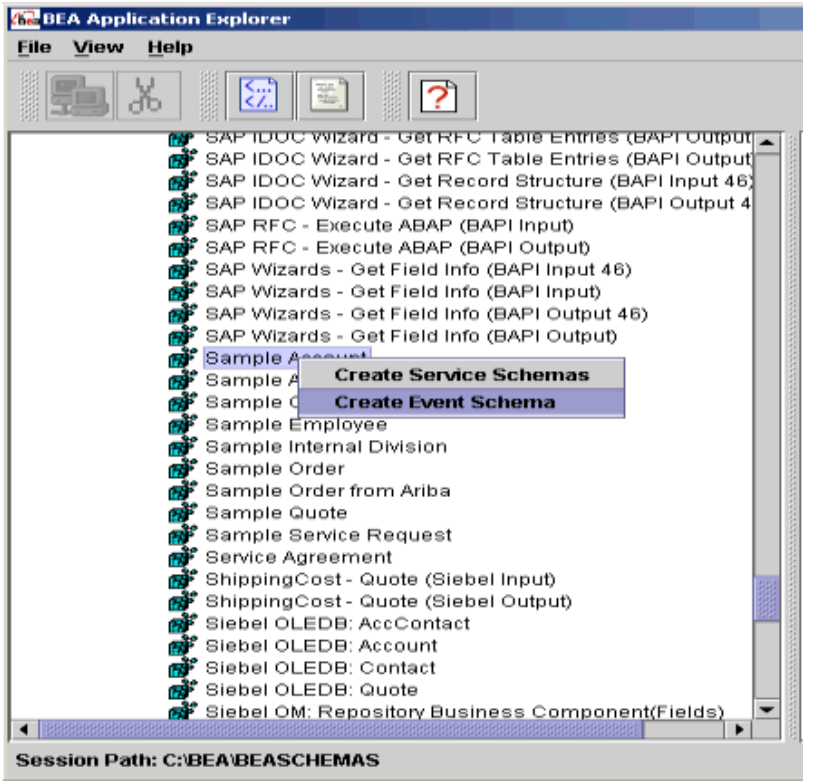
The following items have been added to the folder `C:\BEA\BEASCHEMAS\Siebel\SiebelConnection`:

- `manifest.xml`
- `service_Sample Account1-1-FA22.xsd`
- `service_Sample Account1-1-FA22_response.xsd`

Creating Event Schemas

When you select Create Event Schemas, as described in the last step of "Establishing a Connection to Siebel" on page 3-7, a window opens where you enter the location of the previously generated XDR schema.

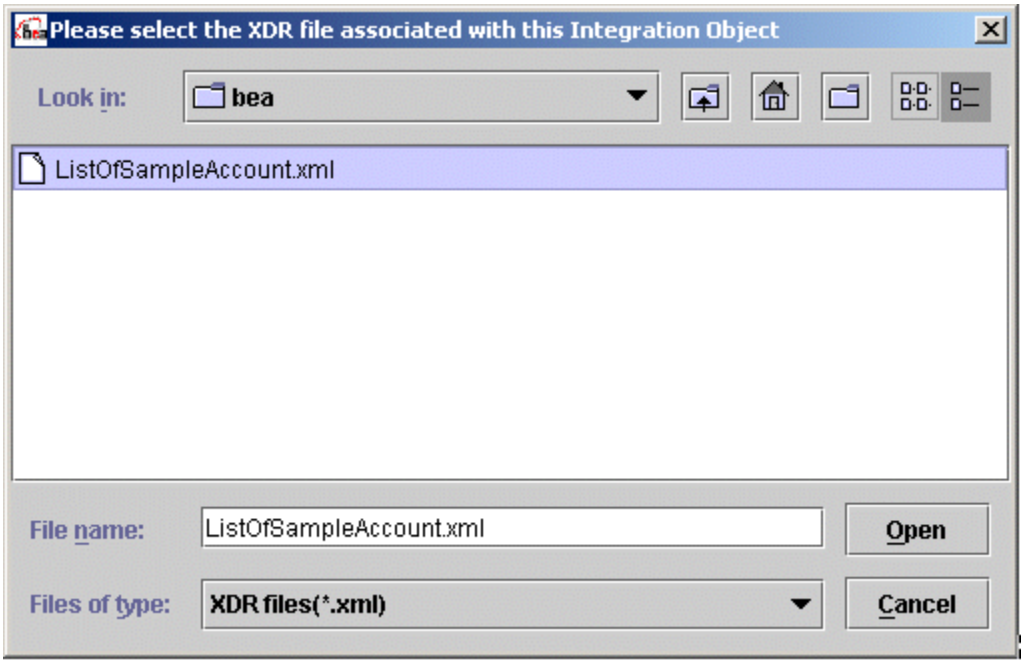
Figure 3-17 BEA Application Explorer - Event Schema Creation



1. Select the file called Sample Account, located in `C:\BEA`.

   The XDR selection window opens.

Figure 3-18 XDR Selection Window



2. Click Open.

   A directory structure is created automatically for you within the working directory, `C:\BEA\BEASCHEMAS`

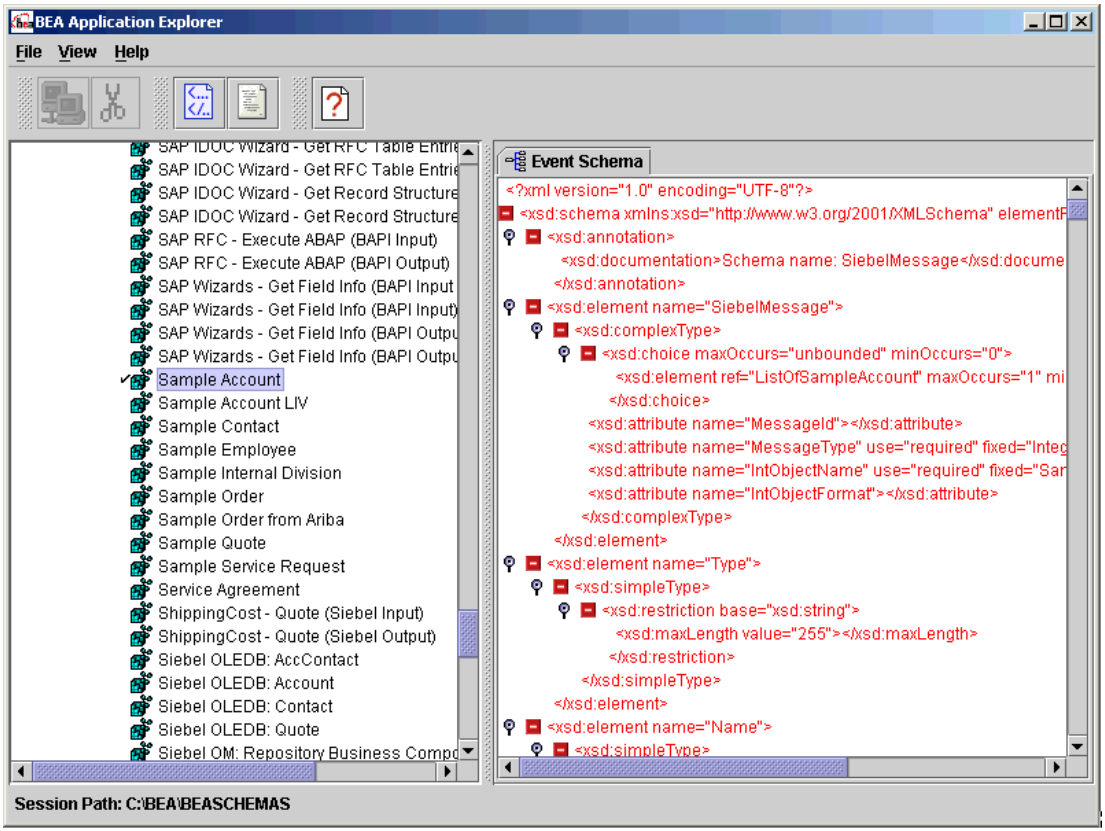
   The BEA Application Explorer creates a folder called Siebel. It also creates subfolders for each configured Siebel connection to contain the schemas created for each connection. In this case, the schemas created for you are located in the folder called SiebelConnection. SiebelConnection is the connection name you established when you connected to the Siebel system using the BEA Application Explorer.

   The following items have been added to folder:

   - `C:\BEA\BEASCHEMAS\Siebel\SiebelConnection`
   - `manifest.xml`
   - `event_Sample Account1-1-FA22.xsd`

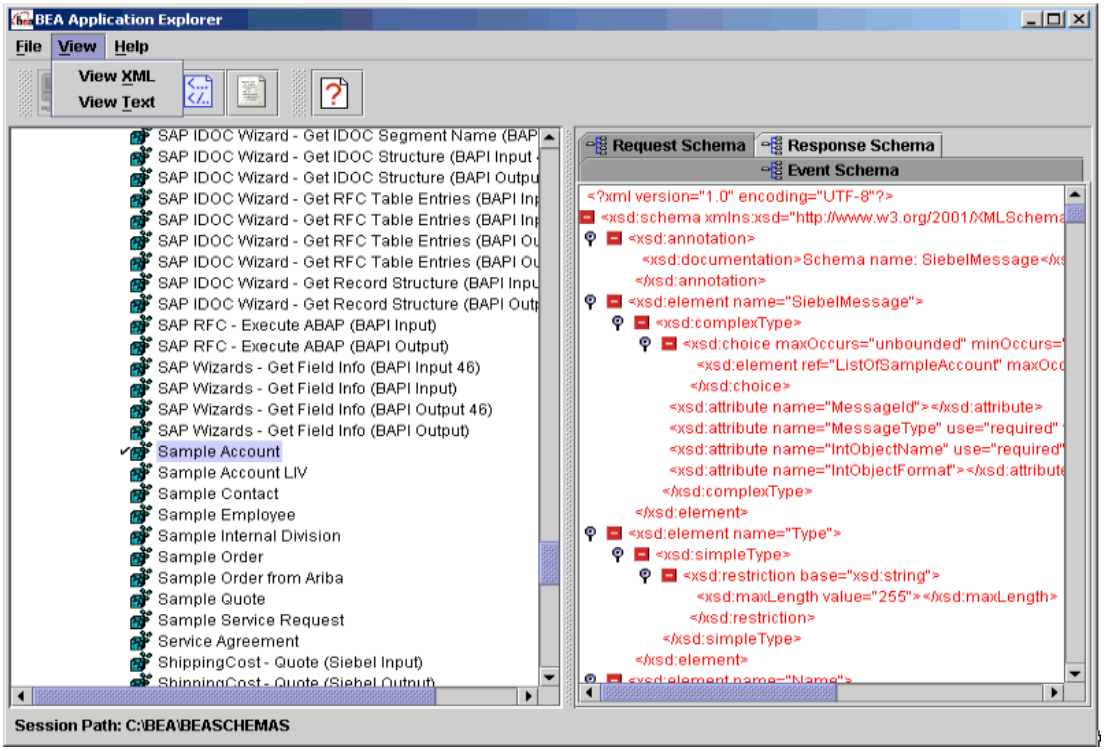
You can also view the created schemas by using the BEA Application Explorer to browse the schemas that have been published for WebLogic Integration.

Figure 3-19 BEA Application Explorer - Event Schema



3. Select Sample Account.

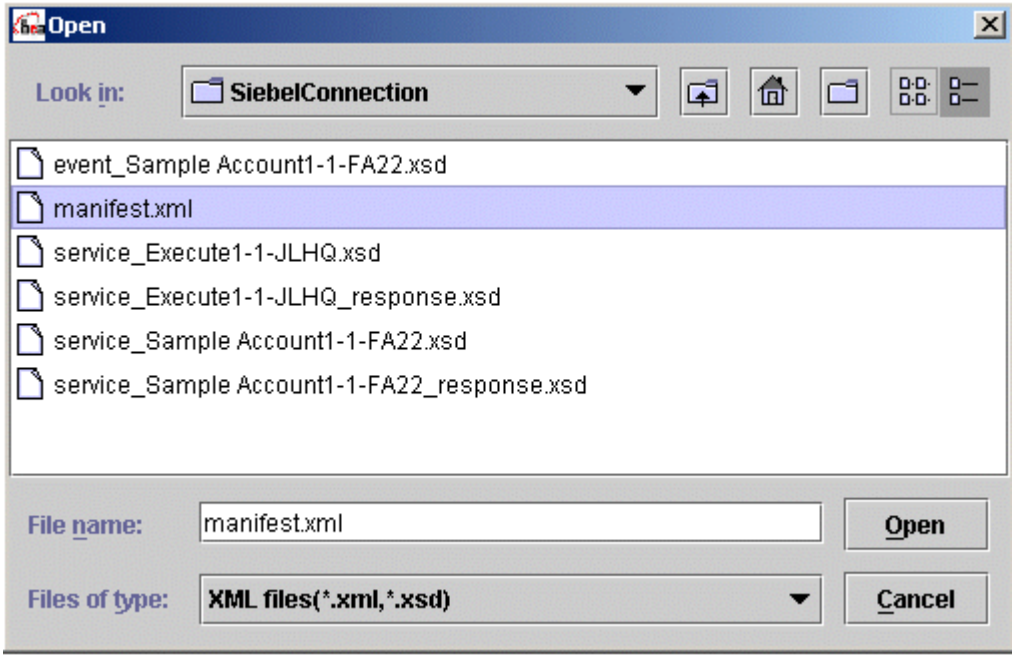
Figure 3-20 BEA Application Explorer - View XML



4. From the View menu, choose View XML.

   The Open box appears.

Figure 3-21 Open Box



5. Point to the explorer working directory to select the desired XML file to view the created schemas and `manifest.xml`.

For example, the `manifest.xml` file for the integration object, Sample Account, contains connection and configuration information. This can be used to test access to the Siebel System using WebLogic Integration console test pages.

Figure 3-22 manifest.xml File

```
C:\bea\BEASCHEMAS\siebel\SiebelConnection\manifest.xml
<manifest>
  <connection>
    <user>SADMIN</user>
    <password>ENCR(241319532013226257)</password>
    <gateway>ARIBA01</gateway>
    <server>SiebelSRV</server>
    <enterprise>ARIBA01</enterprise>
  </connection>
  <schemaref name="Sample Account1-1-FA22">
    <request root="SiebelMessage" file="service_Sample Account1-1-FA22.xsd"></request>
    <response root="SiebelMessage" file="service_Sample Account1-1-FA22_response.xsd"></respon
    <event root="SiebelMessage" file="event_Sample Account1-1-FA22.xsd"></event>
  </schemaref>
  <schemaref name="Execute1-1-JLHQ">
    <request root="BusinessService" file="service_Execute1-1-JLHQ.xsd"></request>
    <response root="BusinessService" file="service_Execute1-1-JLHQ_response.xsd"></response>
  </schemaref>
</manifest>
```

CHAPTER

4 Creating and Editing Application Views

This section explains how to create and edit an application view for any type of service or event supported by the BEA WebLogic Adapter for Siebel. This section contains the following topics:

About Application Views

The BEA WebLogic Adapter for Siebel represents the system-level interface to your Siebel applications. In order to exploit the services and events supported by the adapter, you must provide an interface to the operations you require. You can create such an interface by defining WebLogic Integration application views, or by writing custom code.

When the adapter access method is Siebel XML, the adapter interacts with Siebel integration objects. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, Siebel XML is sent to the adapter.
- Service. The adapter sends Siebel XML to Siebel in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

The following sections provide the basic procedures required to create and edit an application view. For additional information about using WebLogic Integration application integration functionality, see *Using Application Integration*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/aiuser/index.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/index.htm`

After you create the application view, and add services and events to it, a business analyst can use it to create business process workflows in WebLogic Integration Studio that use the services and events. You can create any number of application views, each with any number of services and events.

Starting the Application View Console

To start the Application View Console:

1. Open the following URL in your Web browser.

   `http://host:port/wlai`

   Here, *`host`* is the TCP/IP address or DNS name where WebLogic Server is running, and *`port`* is the socket on which the server is listening.

2. If prompted, enter a user name and password.

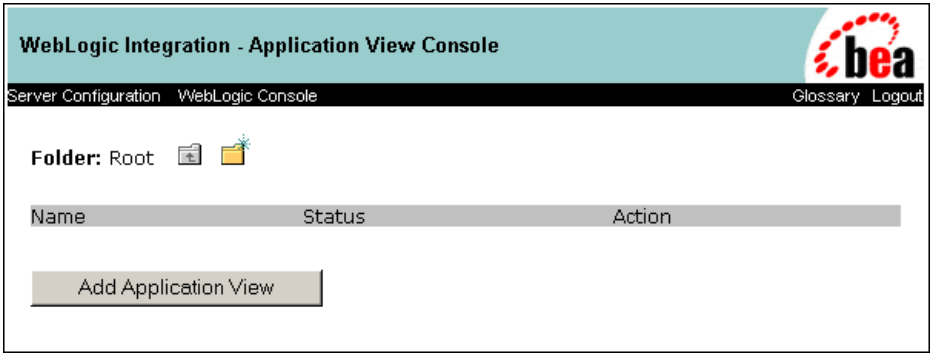
   Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the

`adapter` group, see the *BEA WebLogic Adapter for Siebel Installation and Configuration Guide*.

3. Click Login.

   The WebLogic Integration Application View Console opens.

Figure 4-1 WebLogic Integration Application View Console Window



Creating Folders

The WebLogic Integration Application View Console provides you with a root folder in which you can store all of your application views. If you wish, you can create additional folders to organize related application views into groups.

To create an application view folder:

1. Start the Application View Console as described in "Starting the Application View Console" on page 4-2.

2. Double-click the new folder icon. The Add Folder window opens.

Figure 4-2 Add Folder Window

3. Enter a name for the folder and click Save.

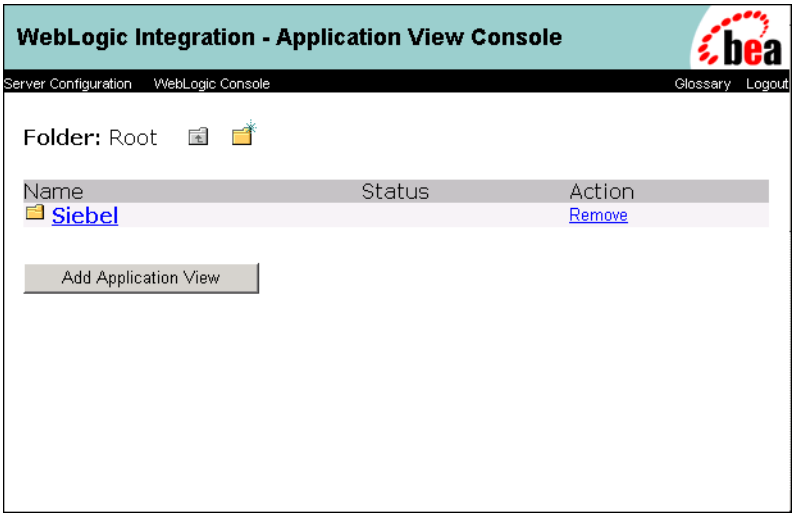
After you create a folder to contain your application views, you can create an application view as described in "Creating an Application View" on page 4-4.

Creating an Application View

To create an application view:

1. Start the Application View Console as described in "Starting the Application View Console" on page 4-2.

Figure 4-3 Application View Console



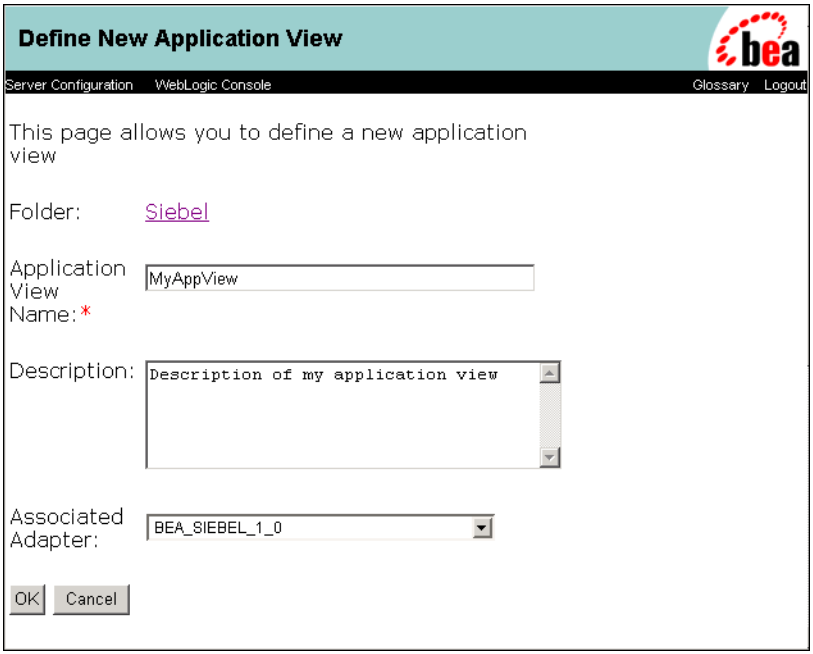
2. Select the desired Application View folder.
3. Click Add Application View.

   The Define New Application View window opens.
4. Enter a name and description for the application view.

   The name should describe the set of functions performed by this application view. Each application view name must be unique to its adapter. Valid characters include a-z, A-Z, 0-9, and _ (underscore).

   The description will be seen by users when they use this application view with business process management workflows.
5. Select `BEA_SIEBEL_1_0` from the Associated Adapter drop-down list.

Figure 4-4 Define New Application View Window



6. Click OK.

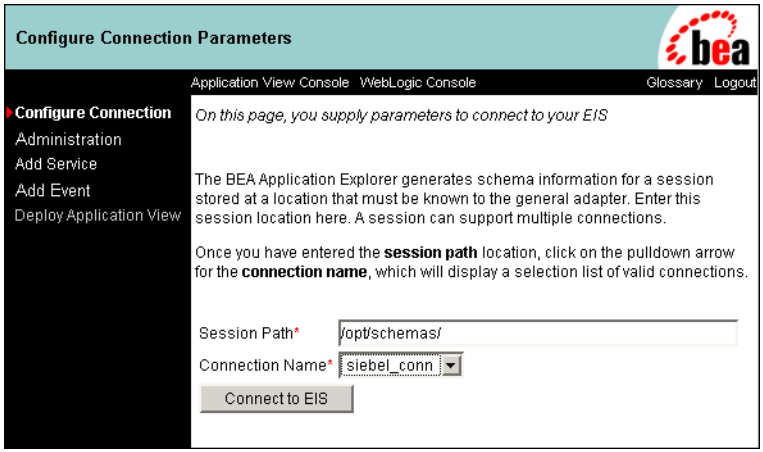
   The Configure Connection Parameters window opens.

7. Enter the BEA WebLogic Adapter for Siebel Session Path. This is the location on the file system of the working directory established for the creation of schemas. It is the same path that you specify when you establish the working directory for the BEA Application Explorer. See the following section for additional information:

   - For Siebel integration objects, see "Establishing the Working Directory" on page 3-6.
   - For Siebel business components and business services, see "Establishing the Schema Working Directory" on page 7-3.

8. Select the connection name from the Connection Name drop-down list. This is the name of the connection used when creating schemas, where the schema `manifest.xml` is located.

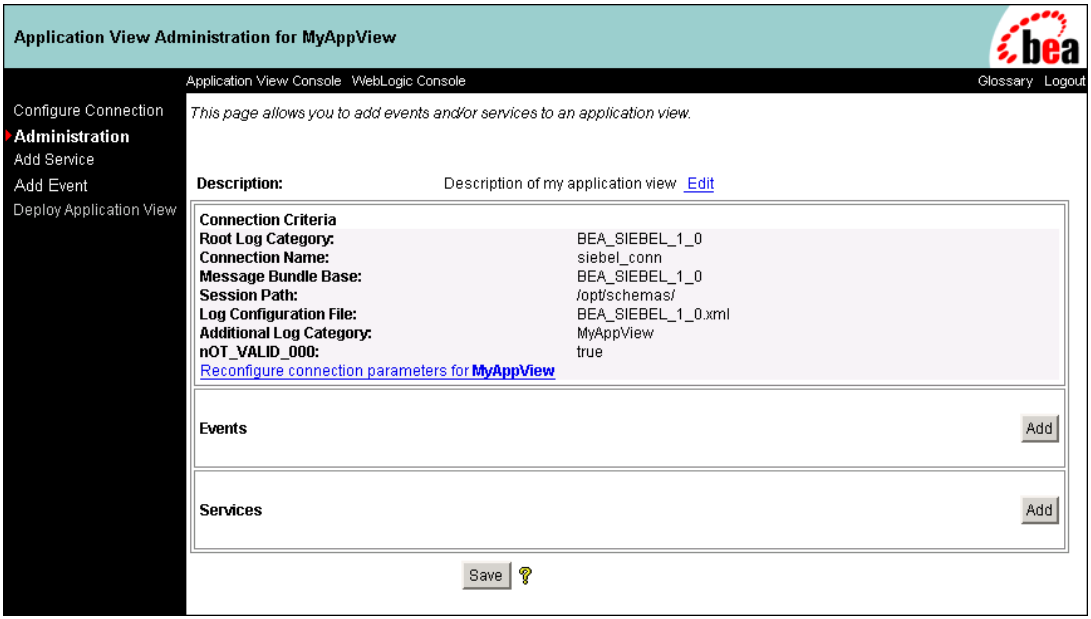
Figure 4-5 Configure Connection Parameters Window



9. Click Connect to EIS. The Application View Administration window opens.

   Note: You can access the Configure Connection Parameters window (displayed in the previous step) when the application view is not deployed by clicking the Reconfigure connection parameters link. If the application view is deployed, you must first undeploy it.

Figure 4-6 Application View Administration Window



10. Click Save.

An application view is typically configured for a single business purpose, and contains all the services and events related to that business purpose. Once you have created the application view, you can add and test the required events and services. See the following sections for instructions:

- Chapter 5, "Adding Application View Events for Siebel Integration Objects."
- Chapter 6, "Adding Application View Services for Siebel Integration Objects."
- Chapter 7, "Using Siebel Business Components and Siebel Business Services."

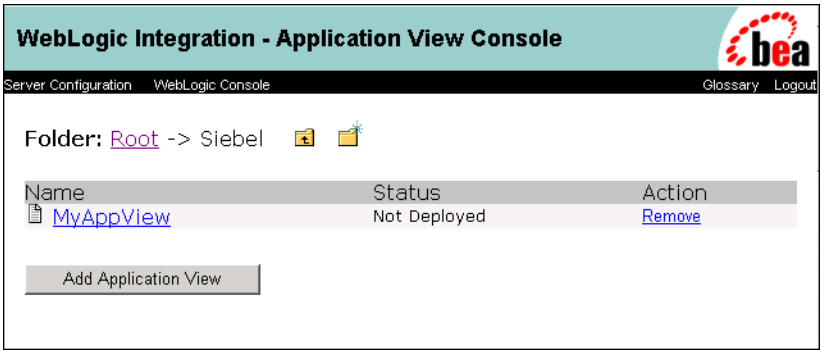
Editing an Application View

To edit an application view:

1. Start the Application View Console as described in "Starting the Application View Console" on page 4-2.

2. Select the desired Application View folder.

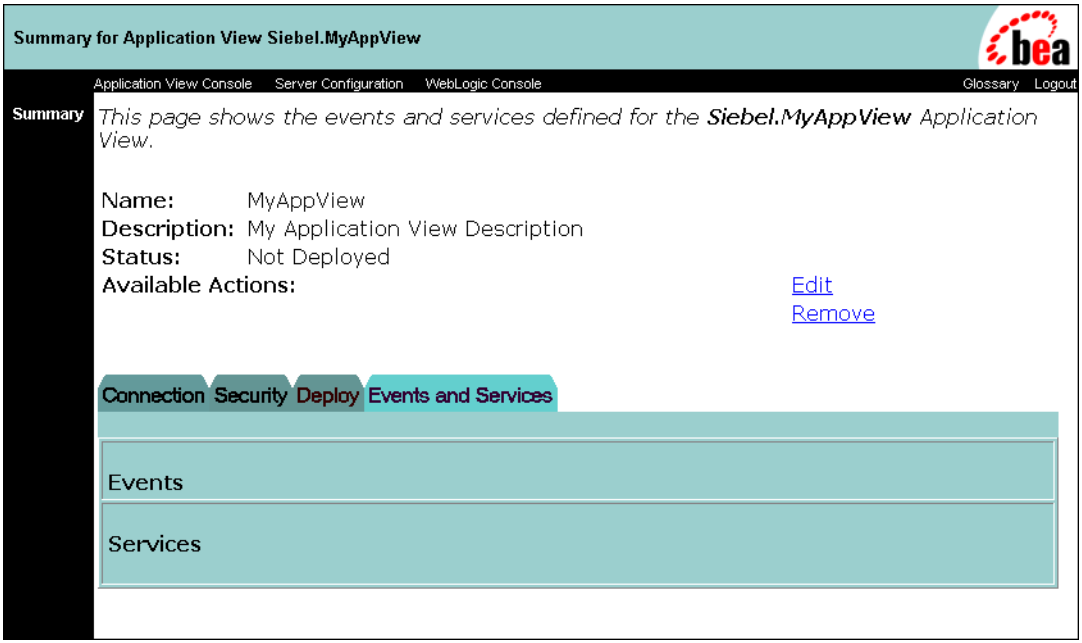
Figure 4-7 Application View Console



3. Select the Application View.

   The Summary for Application View window is displayed.

Figure 4-8 Summary for Application View Window



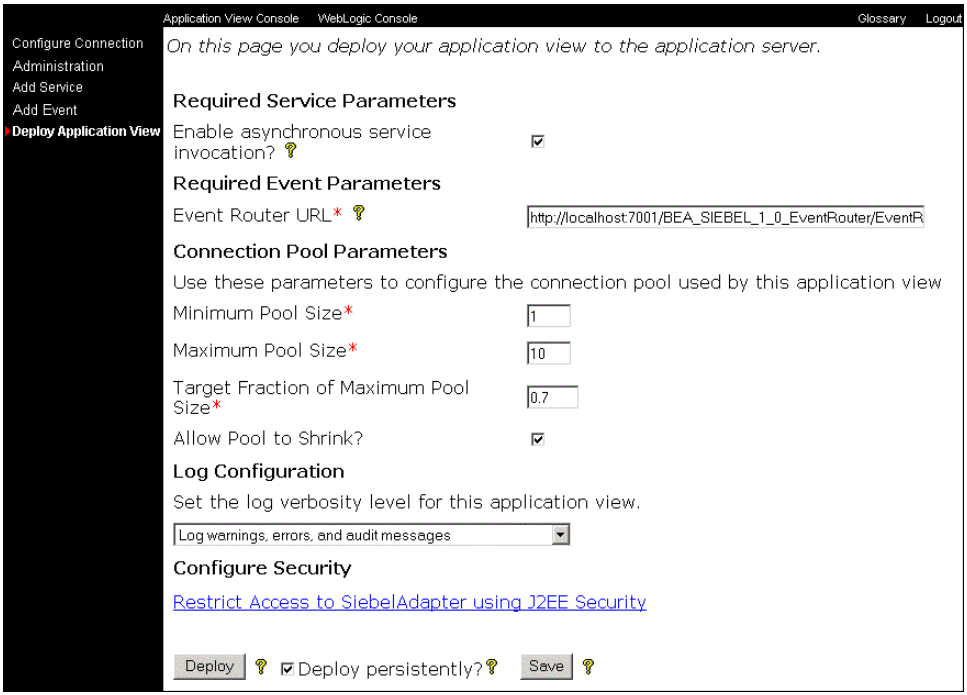
4. If the application view is deployed, select Undeploy from the Available Actions.
5. Select Edit from the Available Actions to display the Application View Administration window.

Deploying an Application View

To deploy an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. Select Continue to display the Deploy Application View window.

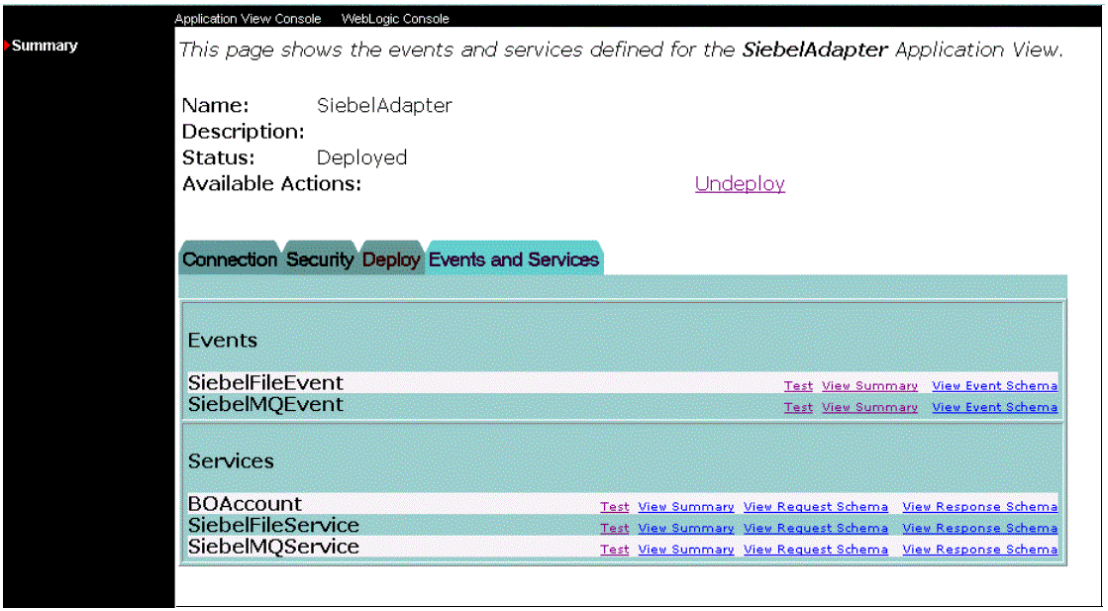
Figure 4-9 Deploy Application View Window



3. Click Deploy to deploy the application view.

   The Summary window opens.

Figure 4-10 Summary Window



You can now employ the event to create WebLogic Integration business process workflows or write custom code. For more information, see "Using Application Views in the Studio" in *Using Application Integration*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/3usruse.htm`

CHAPTER

5 Adding Application View Events for Siebel Integration Objects

After you have created the workflows and schemas required for a Siebel integration object, you can create an application view and add the required services and events. This section explains how to add events using the MQ, File, or HTTP transports. It includes the following topics:

- MQ Events
- File Events
- HTTP Events

For information about adding services using the MQ, File, or HTTP transports, see Chapter 6, "Adding Application View Services for Siebel Integration Objects."

MQ Events

An MQ event is the process by which the adapter picks up a Siebel XML file from a specific IBM MQSeries or WebSphere MQ queue and passes it to an event variable that is set in a business process management workflow. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, sends Siebel XML to the adapter.
- Service. Responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

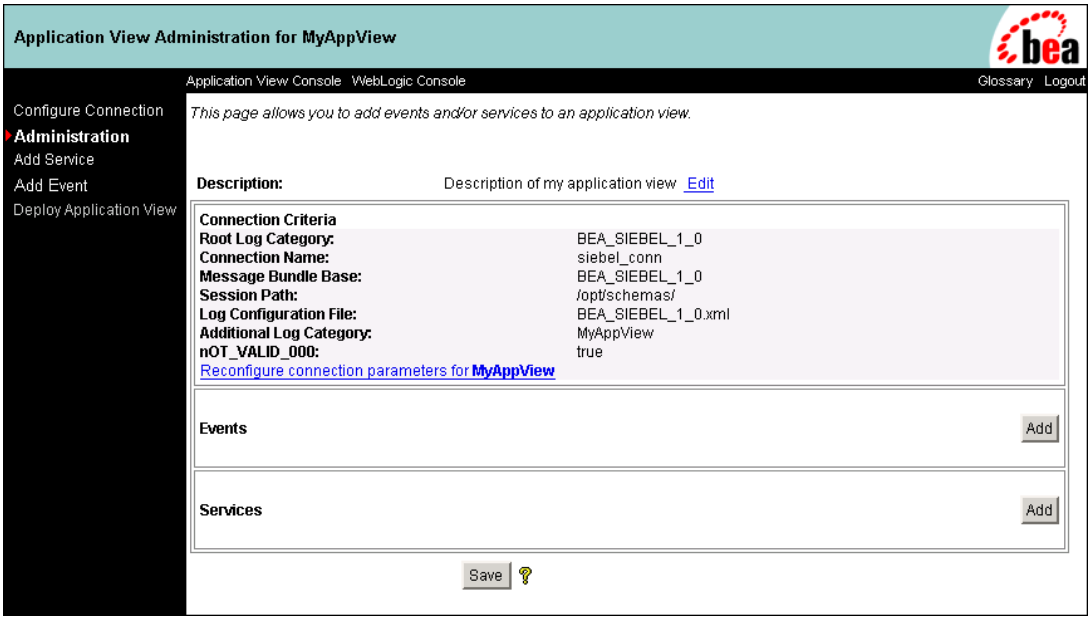
After you create the application view, a business analyst can use it to create business processes that use the application. You can add any number of events and services to an application view.

Adding an MQ Event to an Application View

To add an MQ event to an application view:

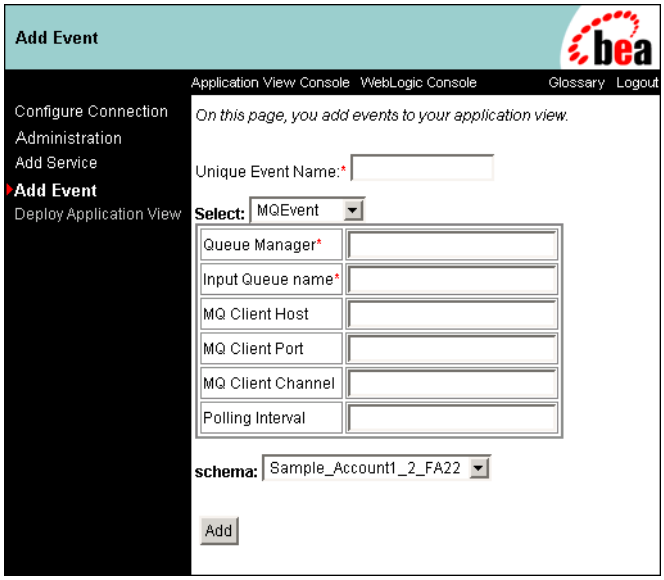
1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.

Figure 5-1 Application View Console Administration Window



2. In the Application View Console Administration Window, do one of the following to display the Add Event window:
   - In the left pane, Click Add Event.
   - In the Events section, click Add.
3. Select MQEvent from the Select drop-down list.

Figure 5-2 Add Event Window



4. In the Unique Event Name field, enter a name, for example, SiebelMQEvent. The name should describe the function performed by this event.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. Enter the required values (required fields are marked with an asterisk). Descriptions of the parameters are provided in the following table:

Table 5-1 MQEvent Parameters

Parameter	Description
Queue Manager* (*Required)	**Type:** String **Description:** Name of the MQ queue manager to be used.
Input Queue Name* (*Required)	**Type:** String **Description:** Enter the MQSeries or WebSphere MQ queue name to be polled for the Siebel XML document.

Table 5-1 MQEvent Parameters (Continued)

Parameter	Description
MQ Client Host	Type: String **Description:** For MQ client only. Host on which MQ Server is located.
MQ Client Port	**Type:** Integer **Description:** For MQ client only. Port number to connect to an MQ server.
MQ Client Channel	**Type/Value:** String **Description:** For MQ client only. Channel between an MQ client and MQ server.
Polling Interval	**Type/Sample Value:** String duration in the format `nnH:nnM:nnS` For example, `1H:2M:3S` (1 hour, 2 minutes, 3 seconds) **Description:** The maximum wait interval between checks for new documents. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread cannot respond to a stop command. If timeout is set to `0`, the listener runs once and terminates. Default is `2` seconds.

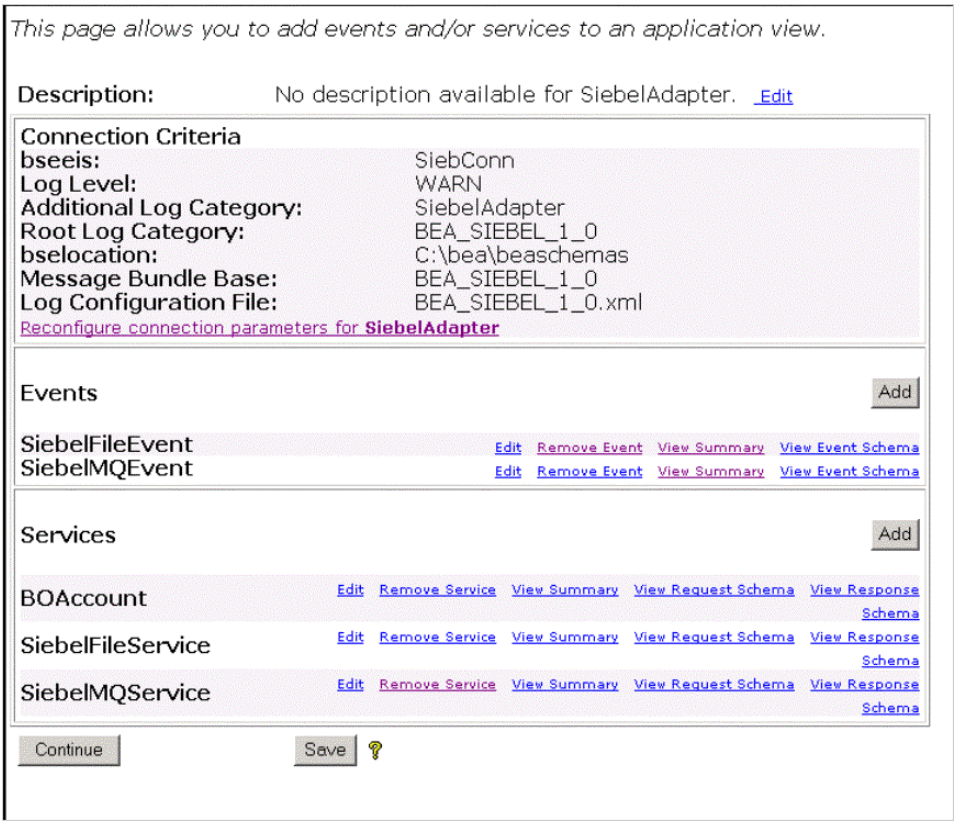
6. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All event schemas created during the session should be listed.

7. Click Add to add the event.

   The event is displayed in the Events section of the Application View Administration window.

Figure 5-3 Application View Administration Window



You can now add additional events or services, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the MQ event, you can test the event as described in the following section.

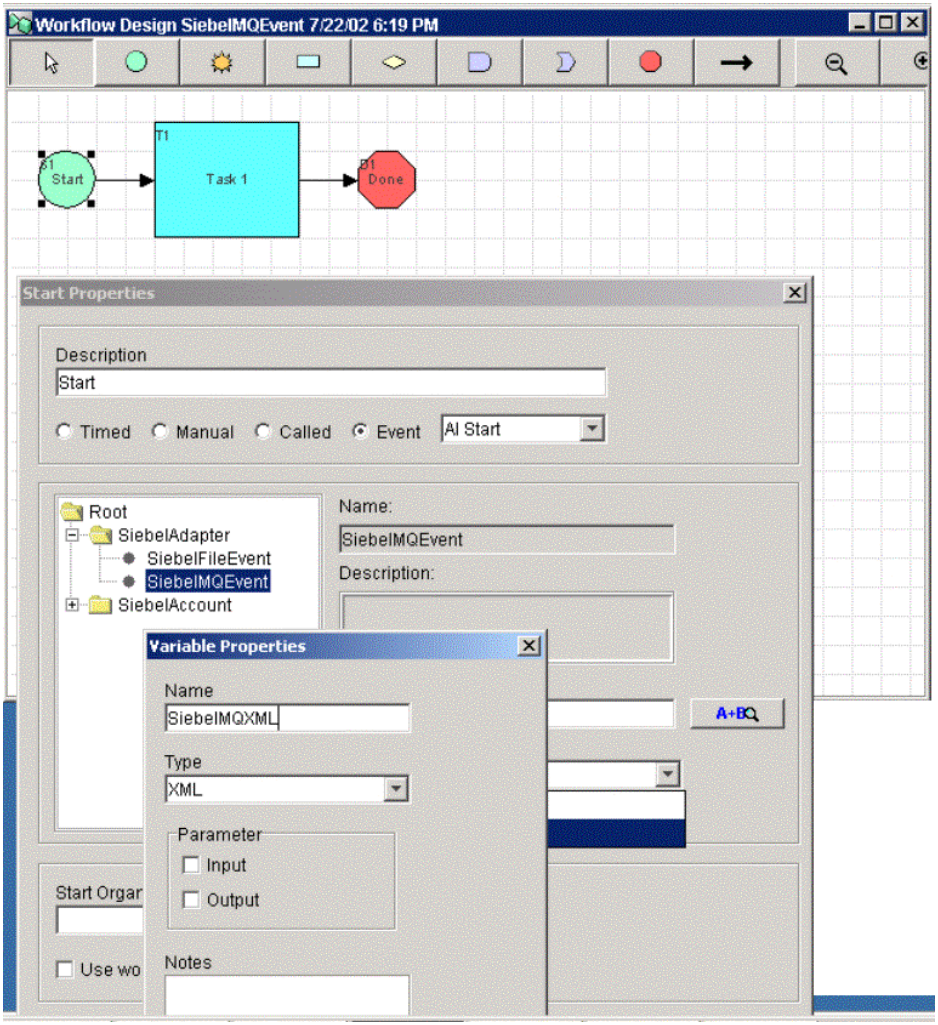
Testing an MQ Event in Studio

To test a deployed application view MQ event in the Studio, launch Studio and create a new template as described in "Using the Studio Interface" in *Using the WebLogic Integration Studio*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/studio/ch2.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/studio/ch2.htm`

For example, create a workflow for the event, SiebelMQEvent, and set up a variable called SiebelMQXML that contains the XML file for the integration object XML called Sample Account from the IBM MQSeries or WebSphere MQ queue.

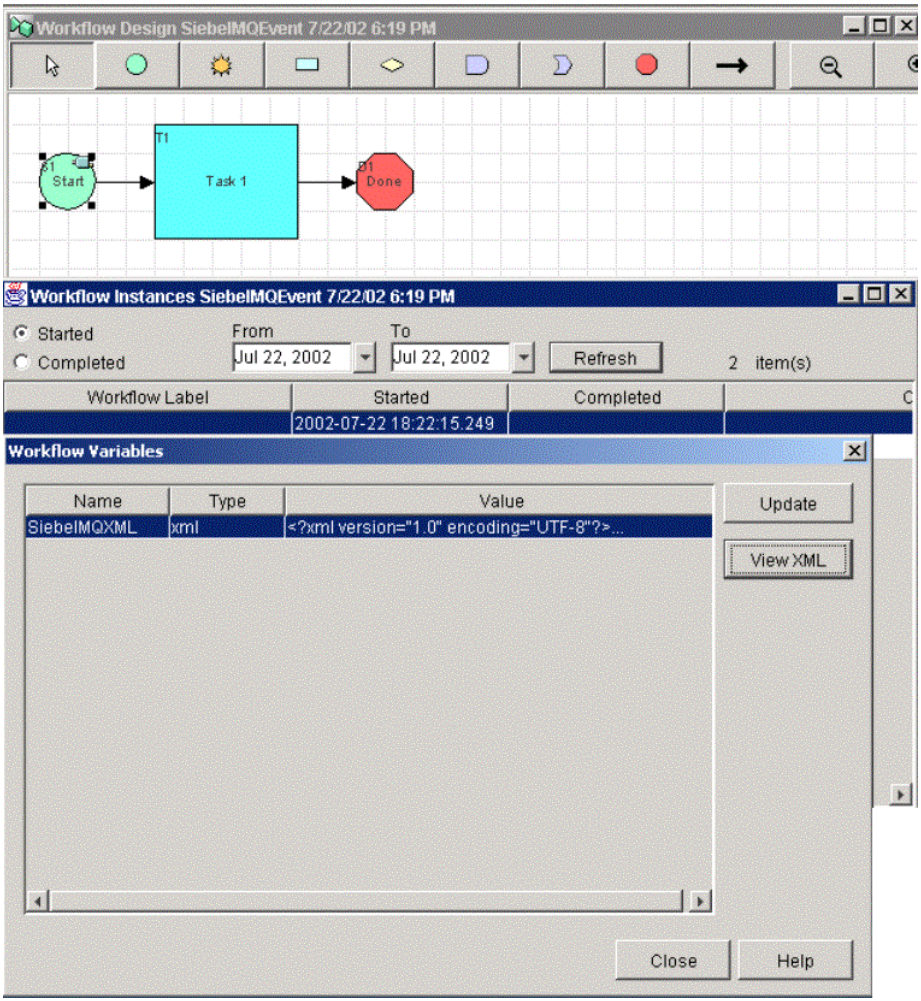
Figure 5-4 Workflow Design for SiebelMQEvent Window



If Siebel has been set up with an IBM MQSeries workflow to enable it to pass an XML integration object XML to the specific IBM MQSeries queue (in this case, `IWAYLAB2.IN`), you can test the Siebel event by "triggering" it by adding a new account. See Appendix B, "Creating Siebel Workflows."

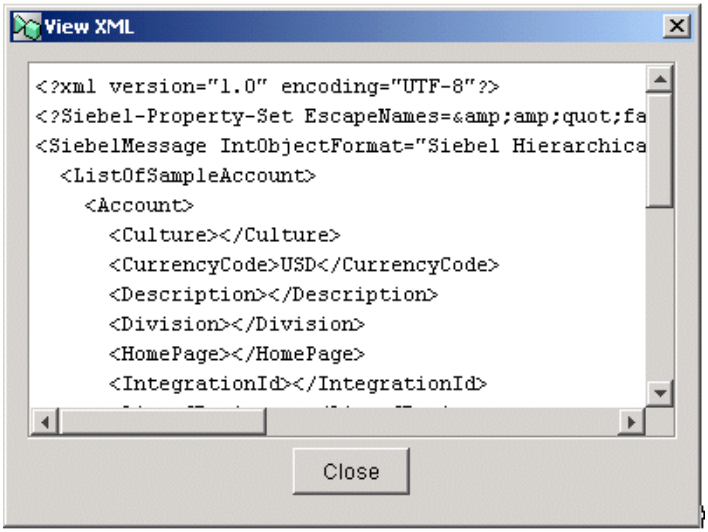
For example, if you add a new account in Siebel, the Siebel Workflow passes the Siebel XML for Sample Account to the WebLogic Application Integration event variable.

Figure 5-5 Workflow Instances SiebelMQEvent Window



A closer look at the event variable displays the Siebel XML for the Siebel integration object.

Figure 5-6 Siebel View XML Window



Note that during the process, the IBM MQSeries queue was populated with the XML document from Siebel but was immediately "consumed" by the WebLogic Application Integration business process management workflow. The end result is the empty IBM MQSeries queue and a populated business process management workflow event variable.

File Events

A file event is the process by which the adapter picks up a Siebel XML file from a specific directory on disk and passes it to an event variable that is set in a business process management workflow. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event**.** When a Siebel event occurs, sends Siebel XML to the adapter.
- Service**.** Responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

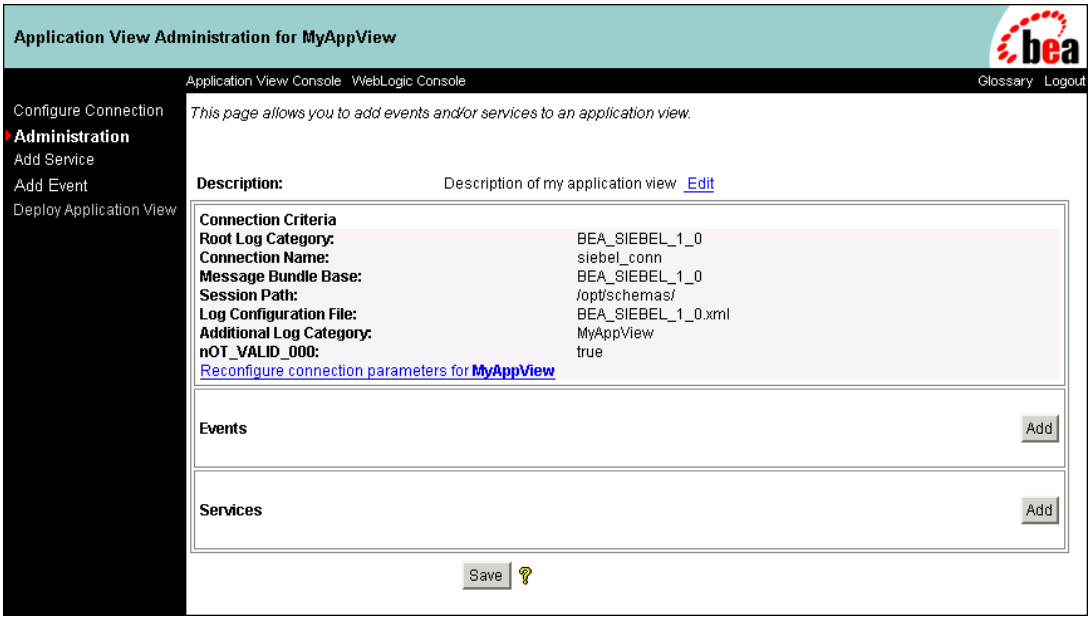
After you create the application view, a business analyst can use it to create business processes that use the application. You can add any number of events and services to an application view.

Adding a File Event to an Application View

To add a file event to an application view:

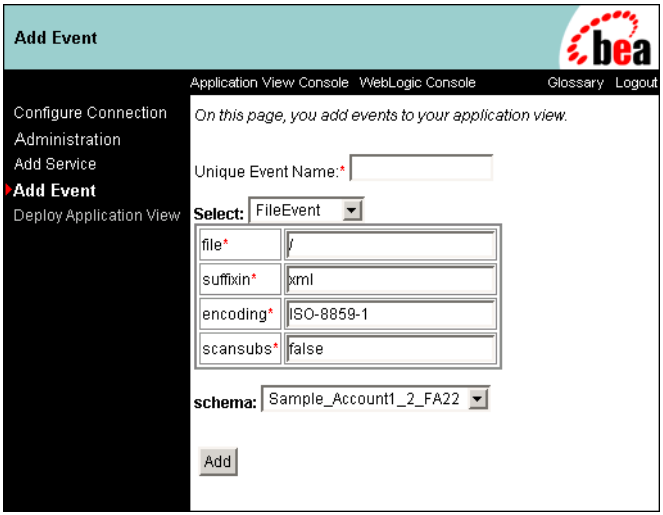
1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.

Figure 5-7 Application View Console Administration Window



2. In the Application View Console Administration Window, do one of the following to display the Add Event window:
   - In the left pane, Click Add Event.
   - In the Events section, click Add.
3. Select FileEvent from the Select drop-down list.

Figure 5-8 Add Event Window



4. In the Unique Event Name field, enter a name, for example, SiebelFileEvent. The name should describe the function performed by this event.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).
5. Enter the required values (required fields are marked with an asterisk). Descriptions of the parameters are provided in the following table:

Table 5-2 FileEvent Parameters

Parameter	Description
file* (*Required)	**Type/Value:** Directory Path **Description:** The file system location to be polled for the file event. For example, enter the location to be polled for the Siebel XML file for the Sample Account.
Suffixin* (*Required)	**Type/Value:** String **Description:** File extension for the file event. For example, enter the `xml` for the Siebel XML file for the Sample Account.
encoding* (*Required)	**Type/Value:** String **Description:** Sets the character set encoding to be used (default value ISO-8859-1-US and Western Europe).
scansubs (*Required)	**Type/Value:** Boolean (true or false) **Description:** Set to true to scan all subdirectories for documents to be processed.

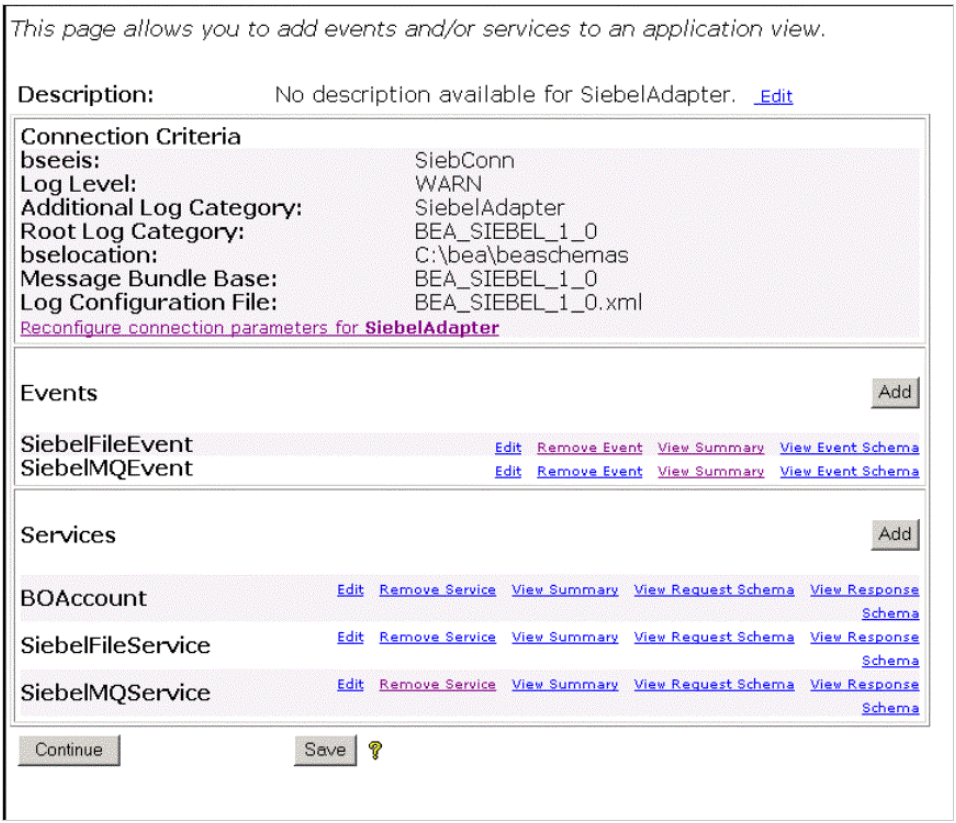
6. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All event schemas created during the session should be listed.

7. Click Add to add the event.

   The event is displayed in the Events section of the Application View Administration window.

Figure 5-9 Application View Administration Window



You can now add additional events or services, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the file event you can test the event as described in the following section.

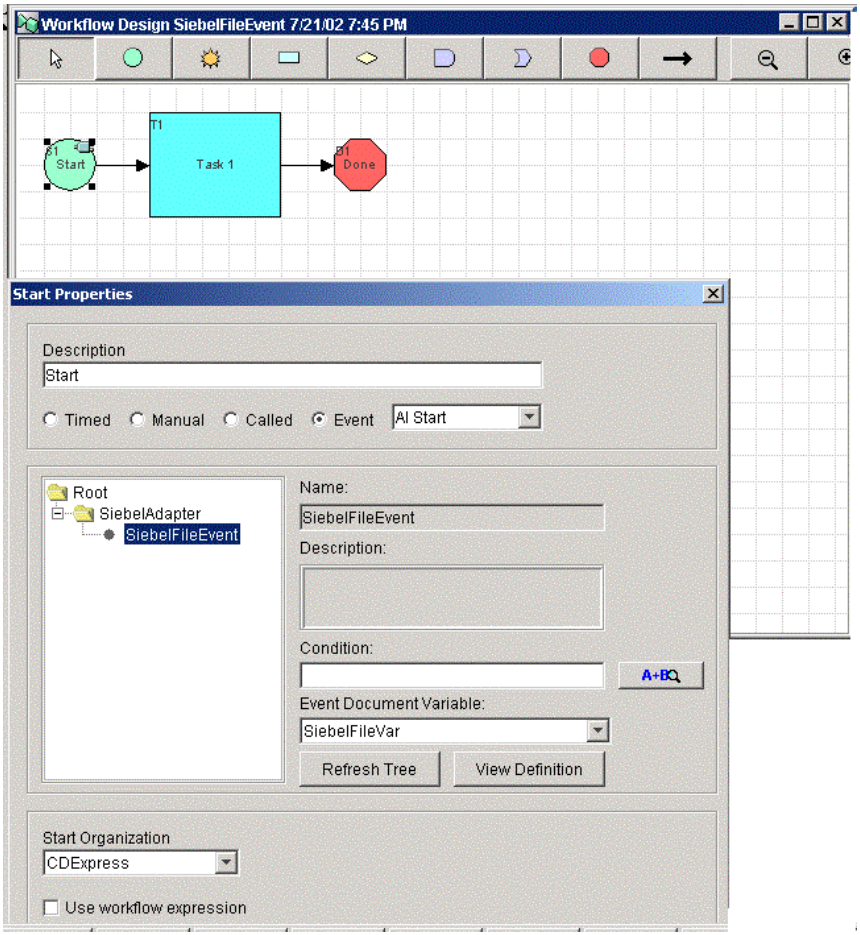
Testing a File Event in Studio

To test a deployed application view file event in the Studio, launch Studio and create a new template as described in "Using the Studio Interface" in *Using the WebLogic Integration Studio*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/studio/ch2.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/studio/ch2.htm`

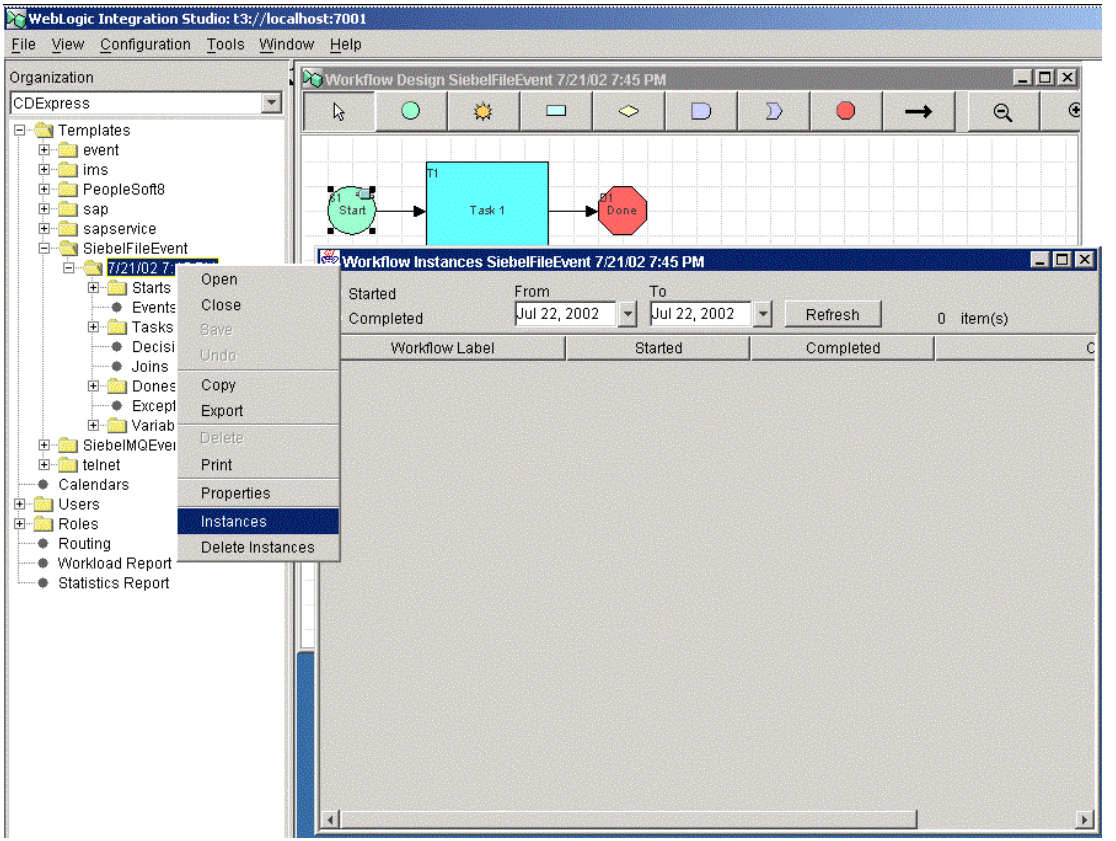
For example, create the workflow for the event called SiebelFileEvent as illustrated in the following figures.

Figure 5-10 SiebelFileEvent Workflow Window



In this example, the variable SiebelFileEvent contains the Siebel XML from the file directory.

Figure 5-11 SiebelFileEvent XML File Window



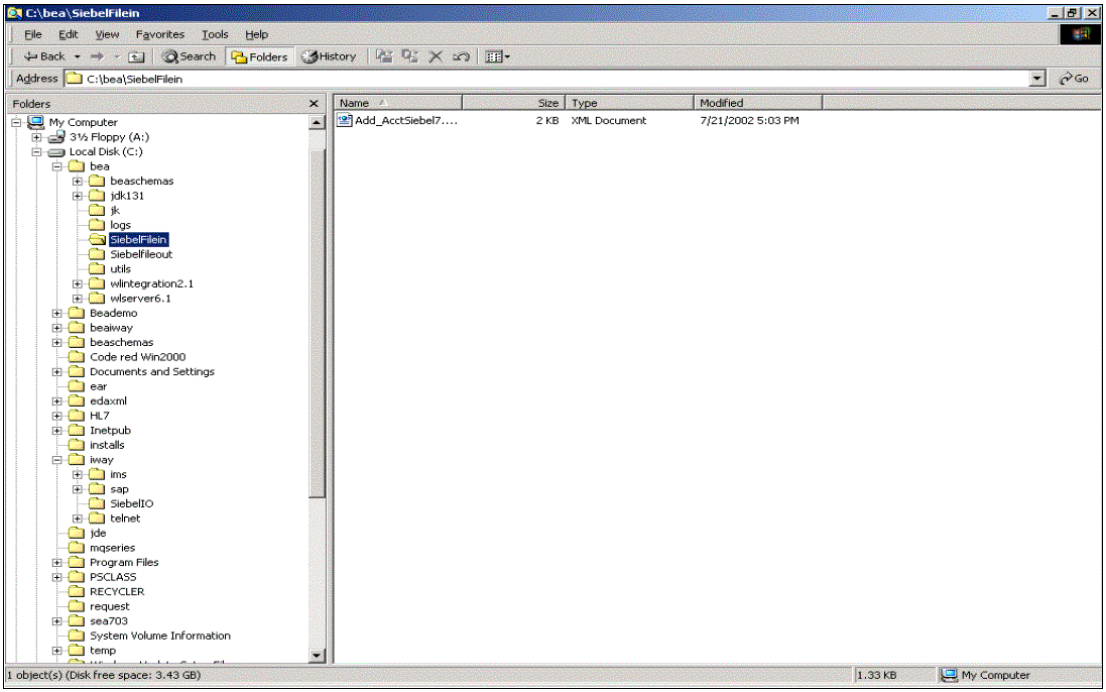
To test the event, copy the Siebel XML file for the sample account into the file directory that the event is listening on. For example, do the following:

1. Copy the file into `C:\bea\SiebelFilein`.

   When the directory is polled, the XML file is taken out of the directory and transported to the input variable called SiebelFileEvent.

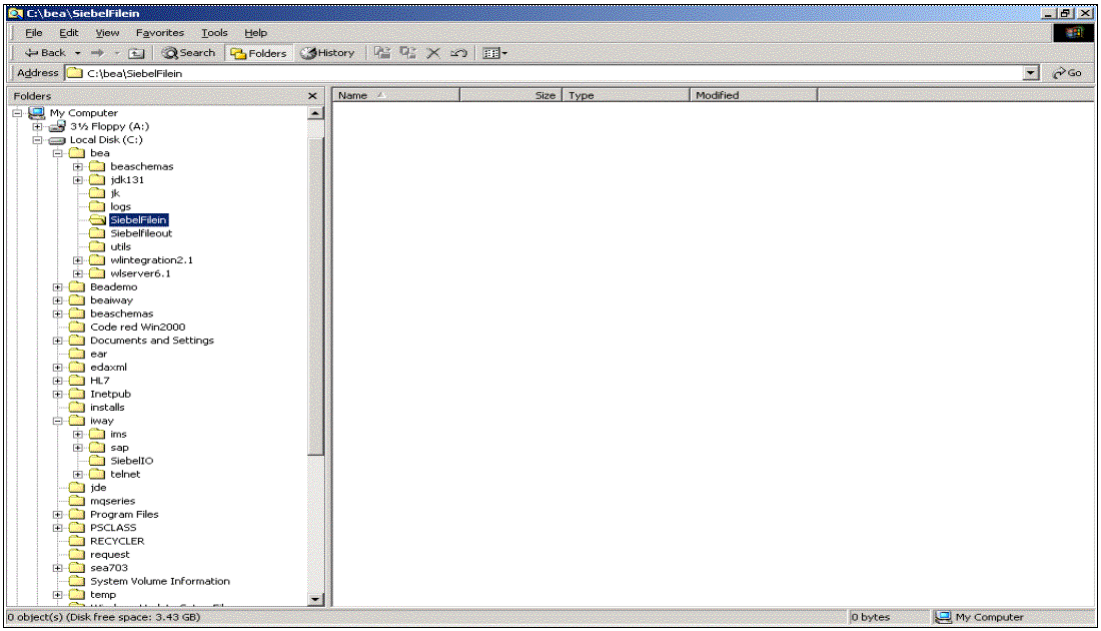
   The following window displays the file that was just copied to the directory:

Figure 5-12 Siebel Directory Window



The file is “consumed” by the Siebel event listener and placed within the variable in the workflow.

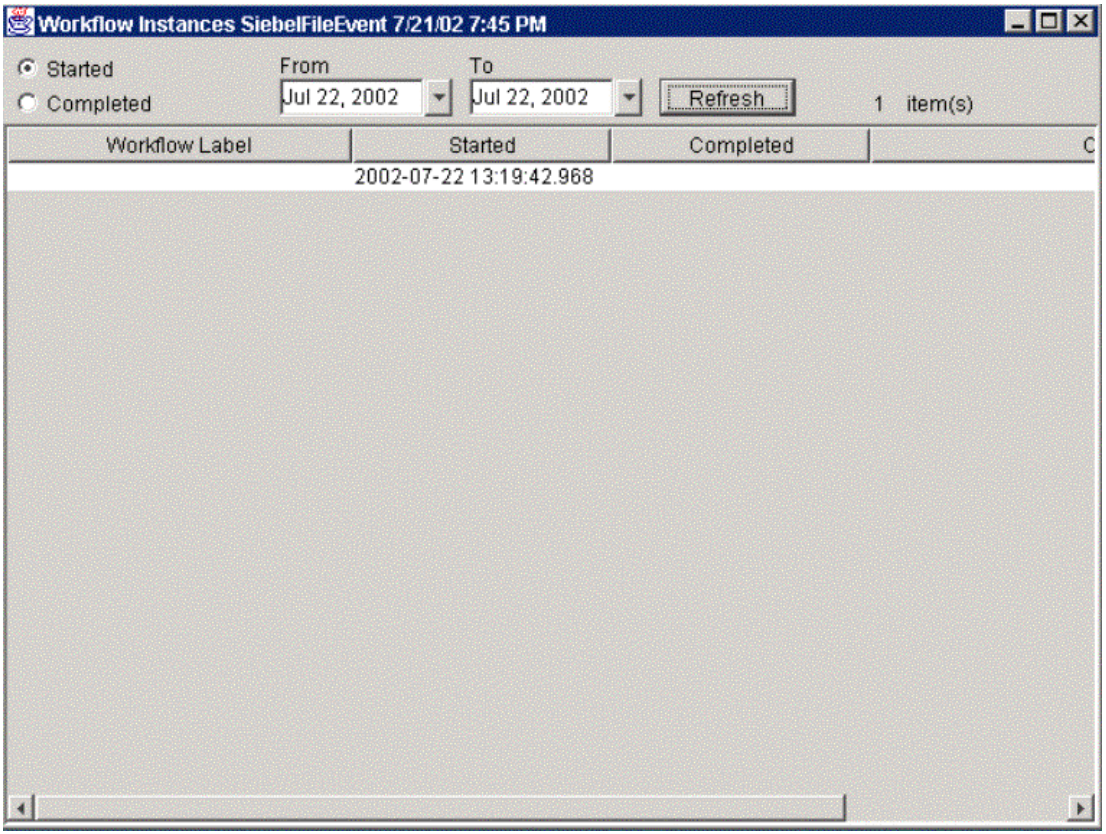
Figure 5-13 SiebelFileIn Window



Note that the file is no longer in the directory.

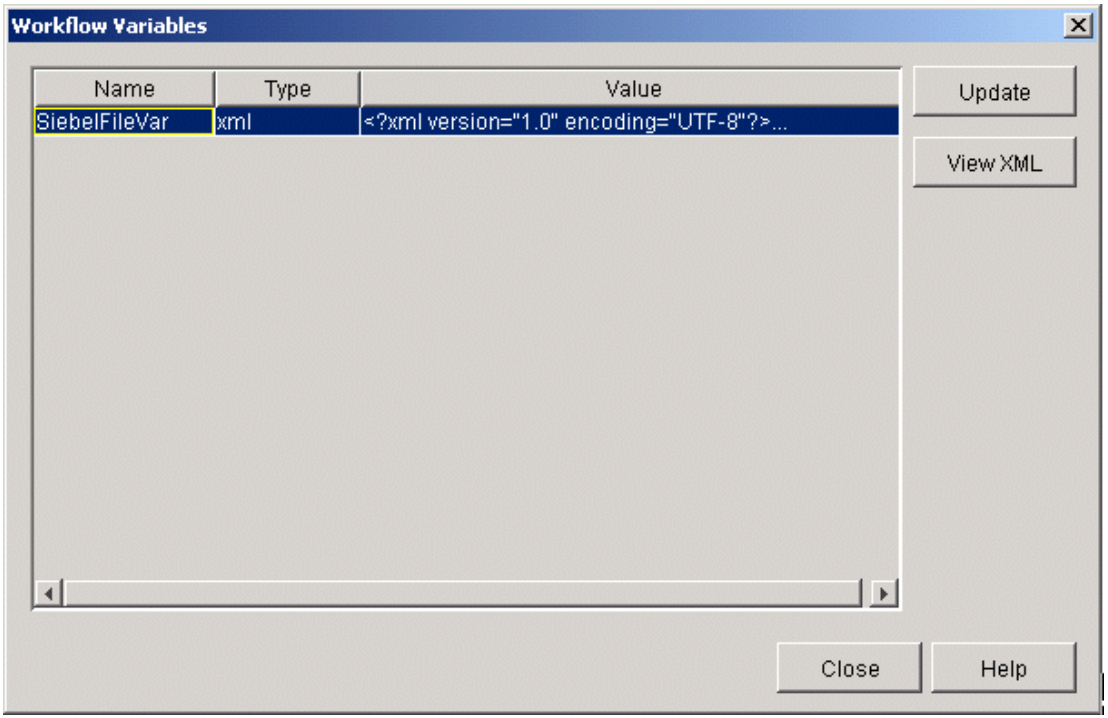
2. Open the WebLogic Workflow window.

Figure 5-14 WebLogic Application Integration Workflow Window



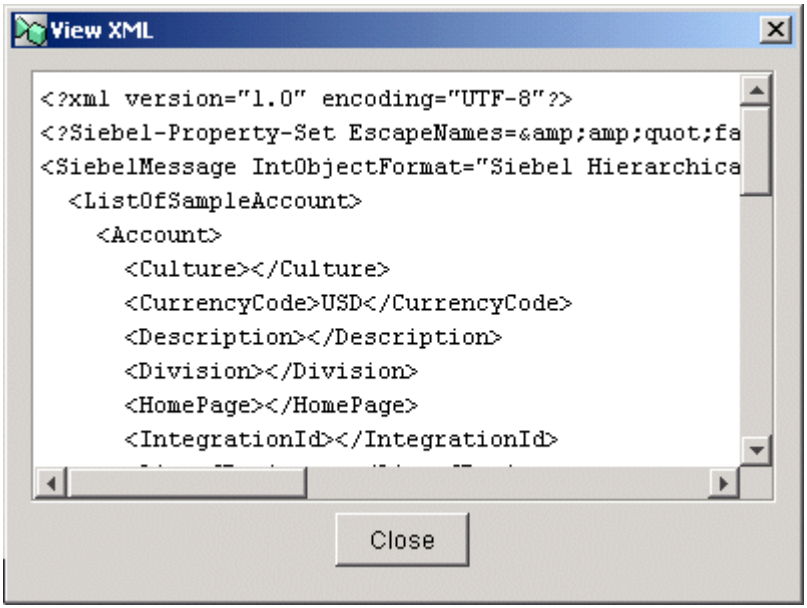
3. Click the date for a closer look at the variable instance.

Figure 5-15 Workflow Variables Window



4. Click View XML.

Figure 5-16 View XML Window



HTTP Events

An HTTP event is the process by which the adapter receives a Siebel XML file using Siebel HTTP and passes it to an event variable that is set in a business process management workflow. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, the workflow sends Siebel XML to the adapter.
- Service**.** The workflow responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

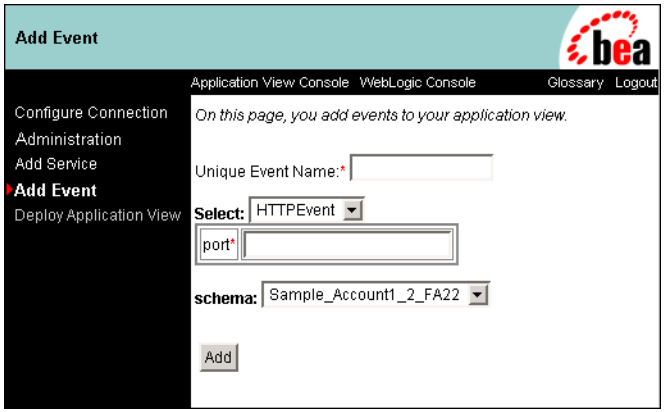
After you create the application view, a business analyst can use it to create business processes that employ the application. You can add any number of events and services to an application view.

Adding an HTTP Event in an Application View

To add an HTTP event to an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. In the Application View Console Administration Window, do one of the following to display the Add Event window:
   - In the left pane, Click Add Event.
   - In the Events section, click Add.
3. Select HTTPEvent from the Select drop-down list.

Figure 5-17 Add Event Window



4. In the Unique Event Name field, enter a name, for example, Sieb7HTTPEvent. The name should describe the function performed by this event.

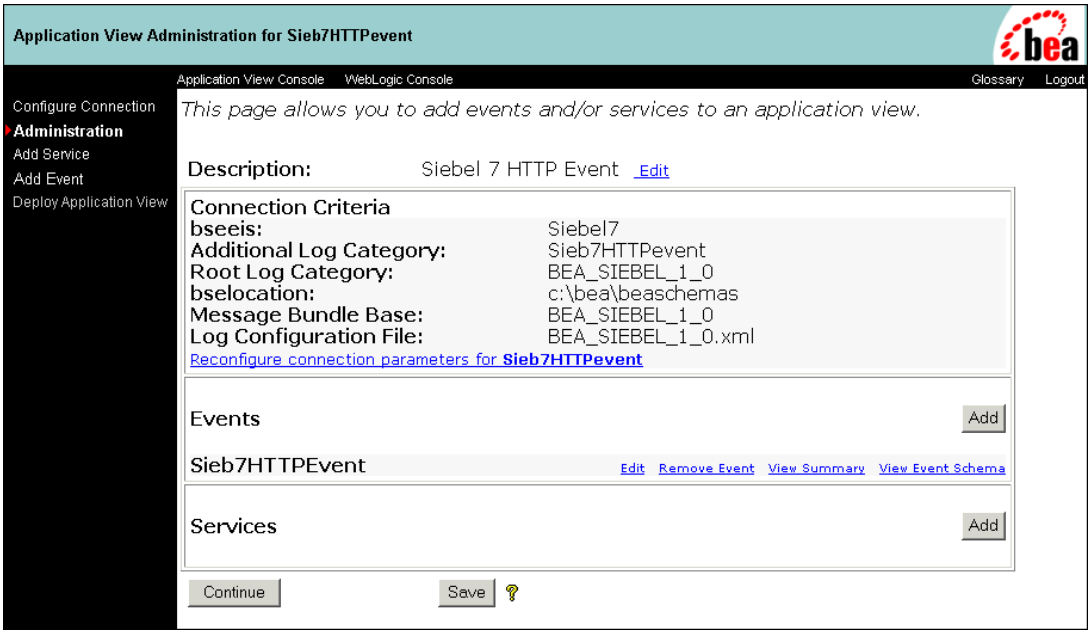
   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. Enter the port number. For example, 4444.
6. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All event schemas created during the session should be listed.
7. Click Add to add the event.

   The event is displayed in the Events section of the Application View Administration window.

Figure 5-18 Application View Administration Window



You can now add additional events or services, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the file event you can test the event as described in the following section.

Testing an HTTP Event in Studio

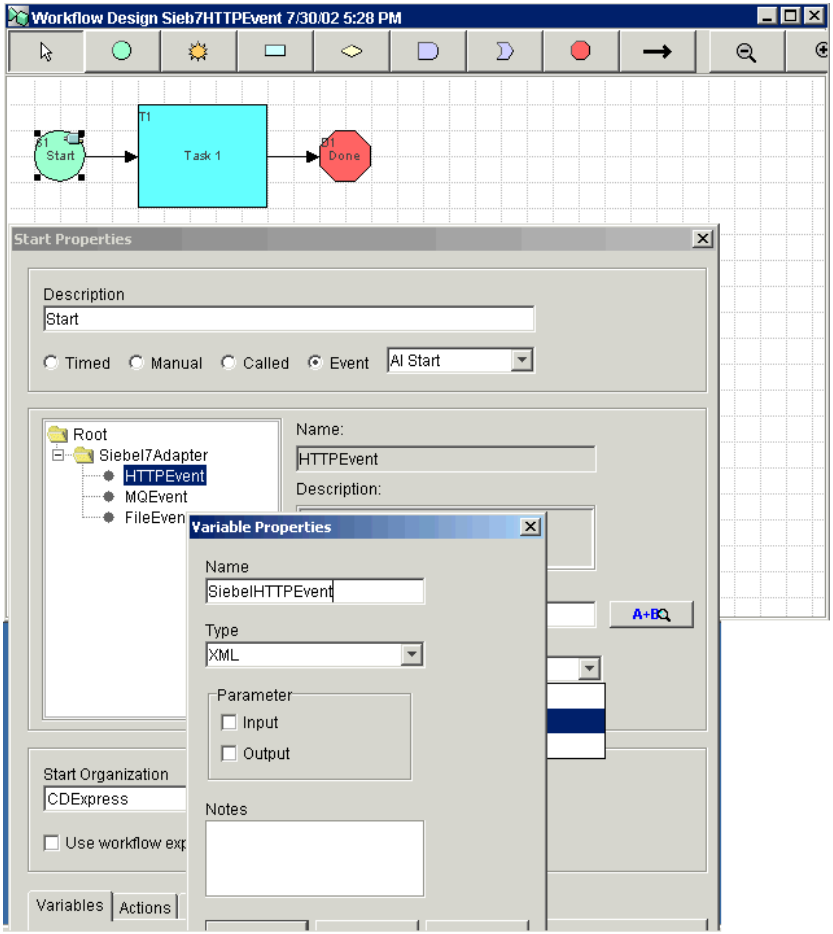
To test a deployed application view file event in the Studio, launch Studio and create a new template as described in "Using the Studio Interface" in *Using the WebLogic Integration Studio*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/studio/ch2.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/studio/ch2.htm`

For example, do the following:

1. Within Studio, create a workflow for the event called Sieb7HTTP.
2. Set up a variable called SiebelHTTPEvent that contains the Siebel XML file for the integration object called Sample Account.

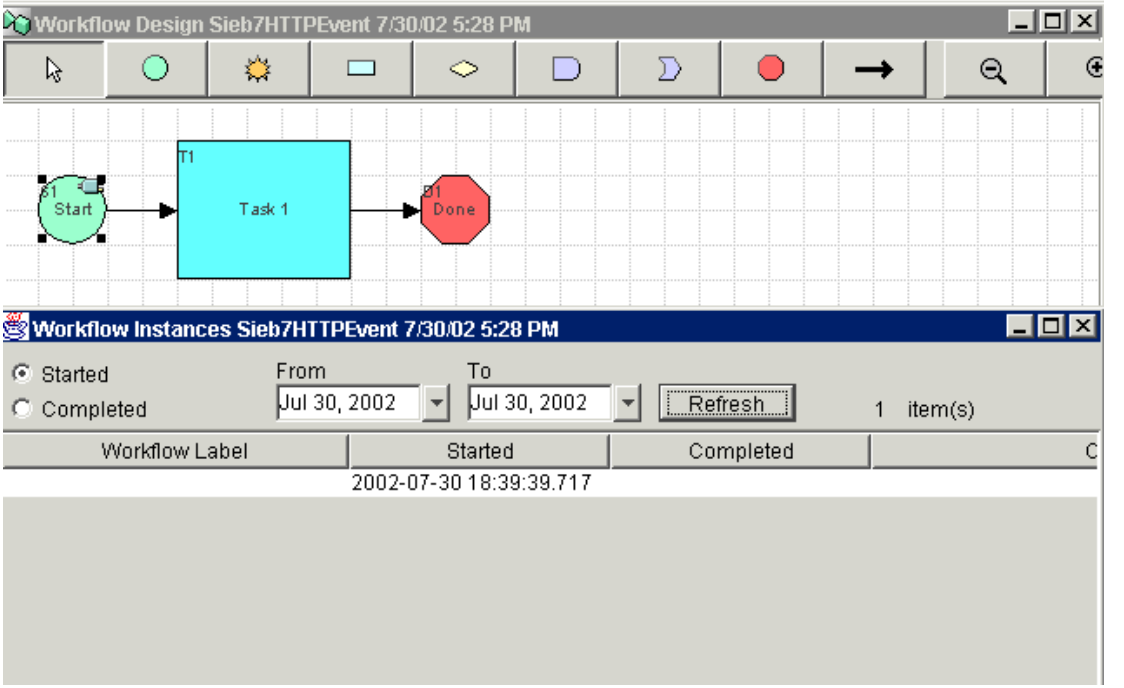
Figure 5-19 WebLogic Integration Studio Window - SiebelHTTPEvent Workflow



3. If a Siebel Workflow has been set to send Siebel XML for Sample Account using the Siebel EAI HTTP transport, you can test the Siebel event by triggering the workflow. See Appendix B, "Creating Siebel Workflows."

   For example, if you add a new account in Siebel, the Siebel Workflow passes the Siebel XML for Sample Account to the event adapter.

Figure 5-20 WebLogic Integration Studio - Instance for SiebelHTTPEvent



a. Select Started.

b. Click Refresh.

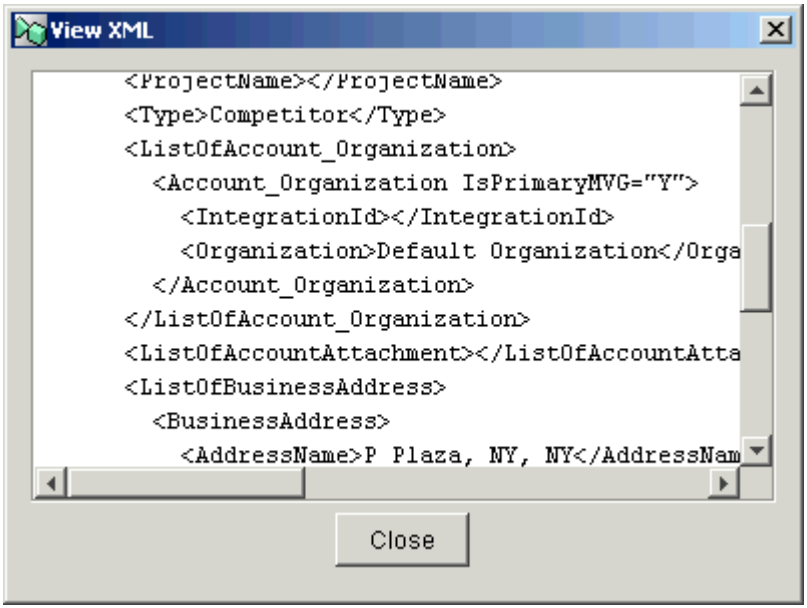
A list of workflows appears in the lower pane.

c. Right-click any instance of the workflow and select Variables.

d. When the Workflow Variables window opens, click View XML to see the entire contents of the workflow message.

A closer look at the event variable displays the Siebel XML for the Siebel integration object Sample Account:

Figure 5-21 View XML Window



The file is “consumed” by the Siebel event listener and placed within the variable in the WebLogic Application Integration business process management workflow.

CHAPTER

6 Adding Application View Services for Siebel Integration Objects

After you have created the workflows and schemas required for a Siebel integration object, you can create an application view and add the required services and events. This section explains how to add servicies using the MQ, File, or HTTP transports. It includes the following topics:

- MQ Services
- File Service
- HTTP Service

For information about adding events using the MQ, File, or HTTP transports, see Chapter 5, "Adding Application View Events for Siebel Integration Objects."

MQ Services

An MQ service is the process by which the adapter sends a Siebel XML file to an IBM MQSeries or WebSphere MQ queue. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, the workflow sends Siebel XML to the adapter.
- Service. The workflow responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

After you create the application view, a business analyst can use it to create business processes that use the application. You can add any number of services and events to an application view.

Adding an MQ Service to an Application View

To add an MQ service to an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. In the Application View Console Administration Window, do one of the following to display the Add Service window:
   - In the left pane, Click Add Service.
   - In the Services section, click Add.
3. Select MQEmitter from the Select drop-down list.

Figure 6-1 Add Service Window

4. In the Unique Service Name field, enter a name, for example, `SiebelMQService`. The name should describe the function performed by this service.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. Enter the required values (required fields are marked with an asterisk). Descriptions of the parameters are provided in the following table:

Table 6-1 MQEmitter Parameters

Parameter	Description
manager* (*Required)	**Type/Value:** String **Description:** Name of the MQ Queue Manager to be used.

Table 6-1 MQEmitter Parameters (Continued)

Parameter	Description
queue* (*Required)	**Type/Value:** String **Description:** The name of the MQSeries or WebSphere MQ queue that is to be polled for an XML document by the Siebel MQSeries Receiver process.
correlid	**Type/Value:** String **Description:** The correlation ID to set in the MQ message header.
MQ Client Host	**Type/Value:** String **Description:** For MQ Client only. Host on which MQ Server is located.
MQ Client Port	**Type/Value:** Integer **Description:** For MQ Client only. Port number to connect to an MQ Server.
MQ Client Channel	**Type/Value:** String **Description:** For MQ Client only. Channel between an MQ Client and MQ Server.

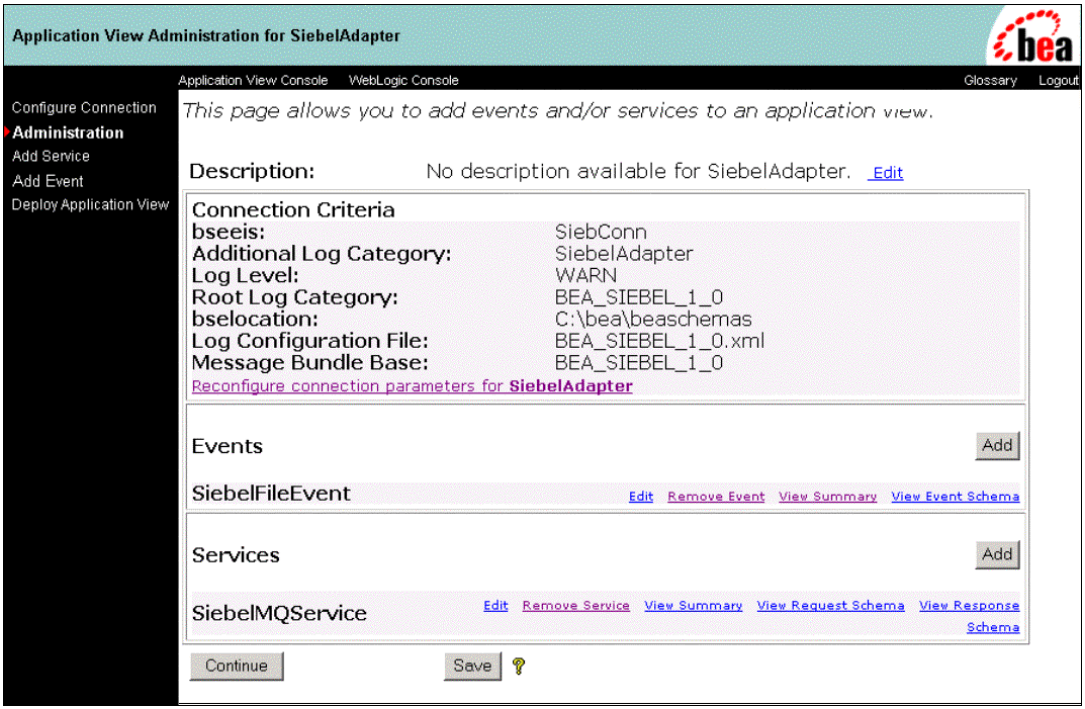
6. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All service schemas created during the session should be listed.

7. Click Add to add the service.

   The service is displayed in the Services section of the Application View Administration window.

Figure 6-2 Application View Administration Window



You can now add additional services or events, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the MQ service, you can test the service as described in the following section.

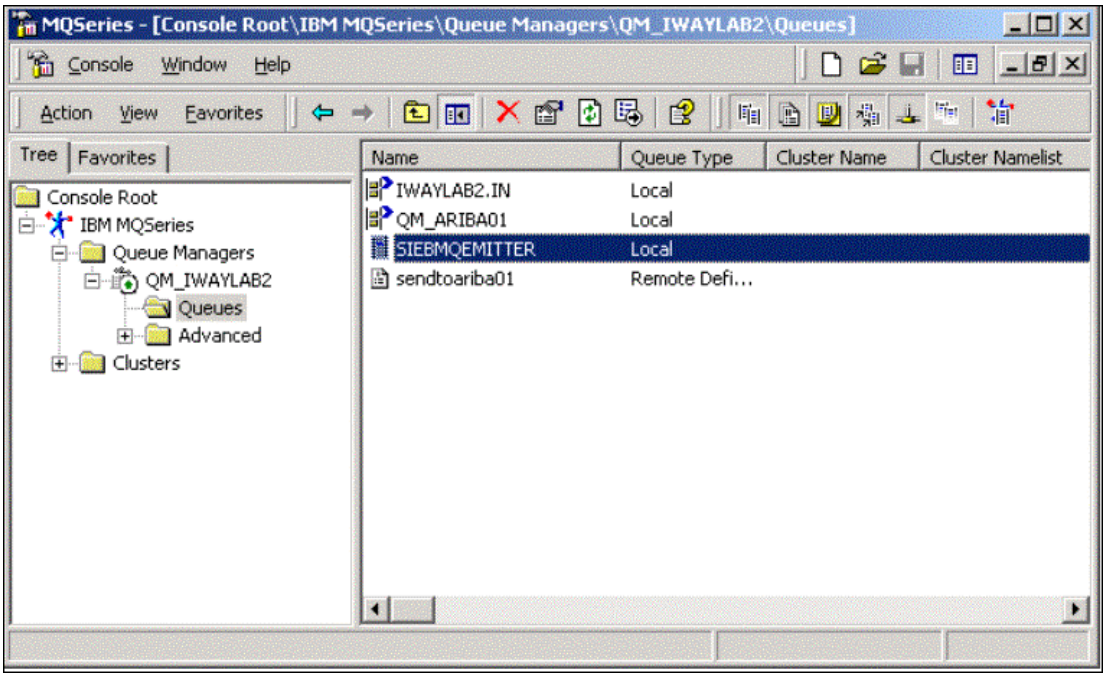
Testing an MQ Service

Before testing the MQ service (MQEmitter), verify that the queue contains no messages. For example, do the following::

1. Open the MQSeries Explorer and choose Queue Managers→*MyQueueManager*→Queues.

   Here, *MyQueueManager* is the queue manager name specified when you added the MQEmitter service to the application view.

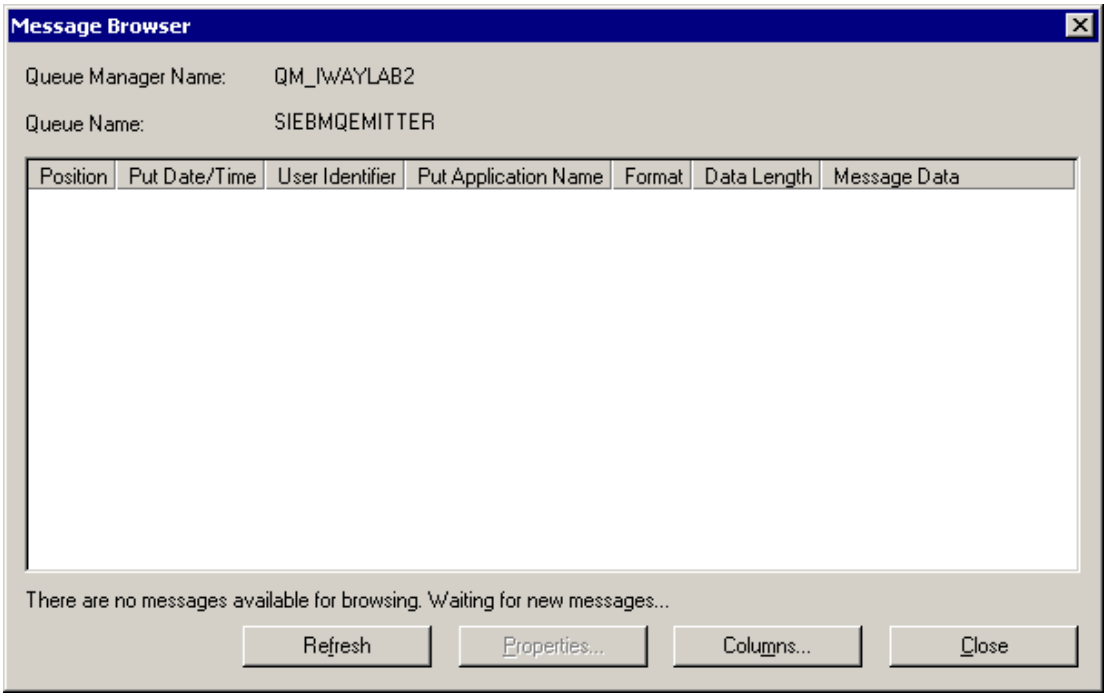
Figure 6-3 MQSeries Explorer Window



2. Click the queue you specified when you added the MQEmitter service to the application view.

3. Verify that there is no message.

Figure 6-4 Message Browser Window



To test the MQSeries service:

1. In the Summary for Application View window, click the Test link for the service. For example, click the Test link for `SiebelMQService` as shown in the following figure.

Figure 6-5 Summary for Application View Window

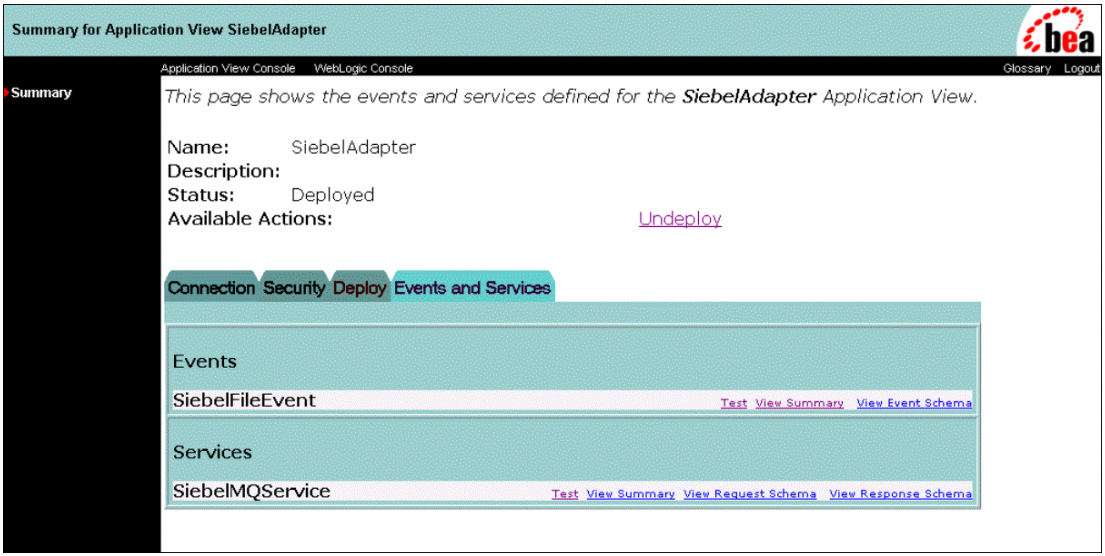
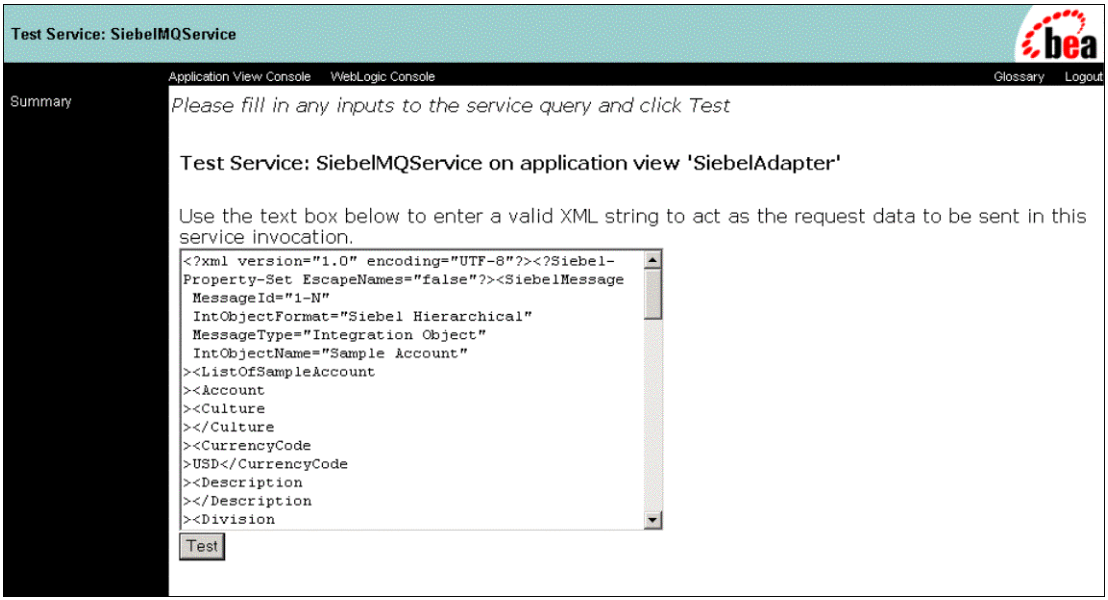


Figure 6-6 Test Service Window

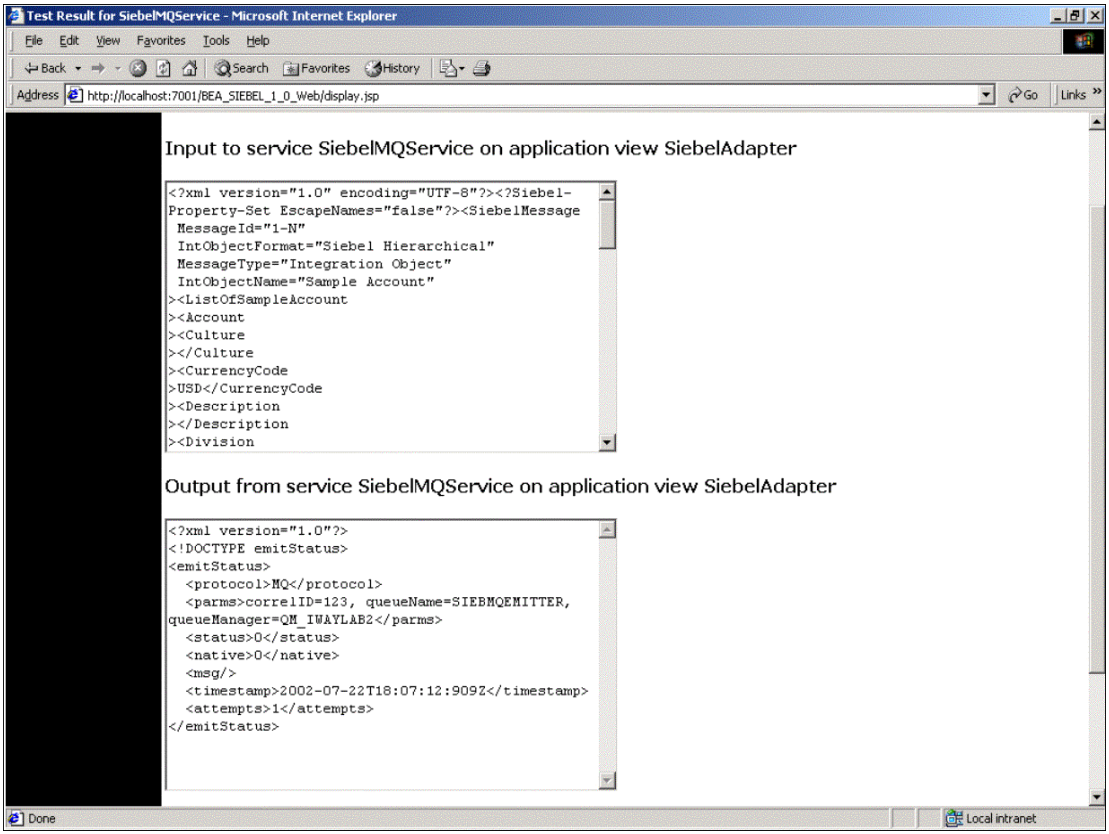


2. Insert a Siebel XML file instance for the Siebel integration object Sample Account.

3. Click Test.

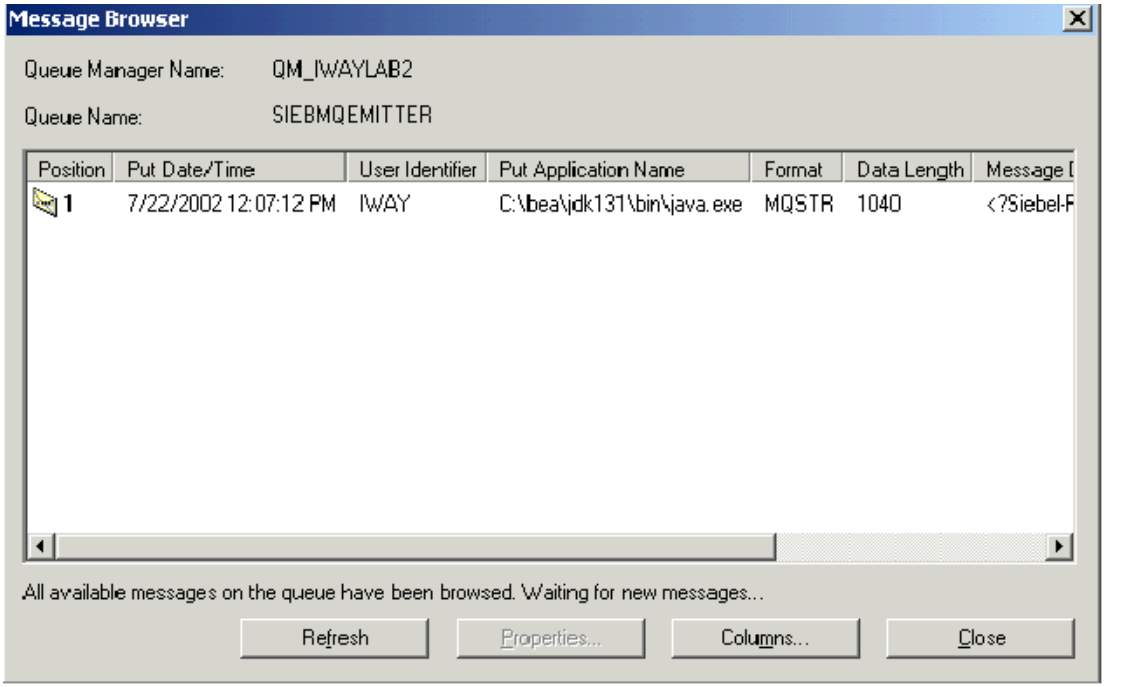
   The result is displayed with a status of 0, indicating the successful placement of the Siebel XML on the MQ queue.

Figure 6-7 Input and Output Service Window



4. In the MQSeries Explorer, select Queue Managers→*MyQueueManager*→Queues. The queue now contains the XML message. This indicates that the service ran successfully.

Figure 6-8 Message Browser Window



A Siebel Workflow configured to listen using the Siebel EAI MQSeries Transport can now retrieve and further process the XML document.

File Service

A file service is the process by which the adapter sends a Siebel XML file to a specific directory on disk. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, the workflow sends Siebel XML to the adapter.

- Service. The workflow responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

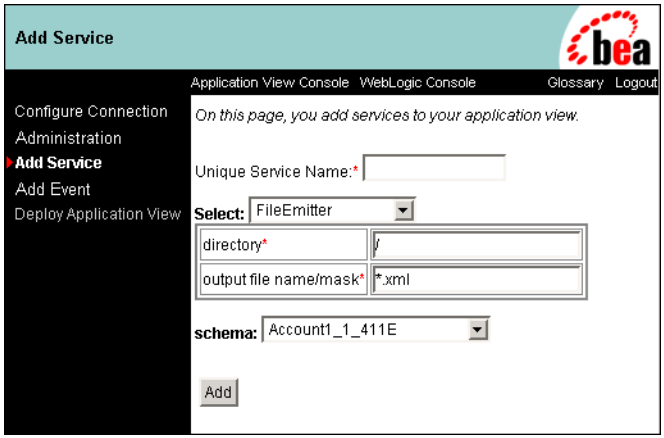
After you create the application view, a business analyst can use it to create business processes that use the application. You can add any number of services to an application view.

Adding a File Service to an Application View

To add a file service to an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. In the Application View Console Administration Window, do one of the following to display the Add Service window:
   - In the left pane, Click Add Service.
   - In the Services section, click Add.
3. Select FileEmitter from the Select drop-down list.

Figure 6-9 Add Service Window



4. In the Unique Service Name field, enter a name, for example, `SiebelFileService`. The name should describe the function performed by this service.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. Enter the target file system location to for the Siebel XML file. For example, `c:\bea\Siebelfileout`.

6. Enter the output file name/mask. For example, `siebelxmlout.xml`.

7. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All service schemas created during the session should be listed.

8. Click Add to add the service.

   The service is displayed in the Services section of the Application View Administration window.

You can now add additional services or events, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the File service, you can test the service as described in the following section.

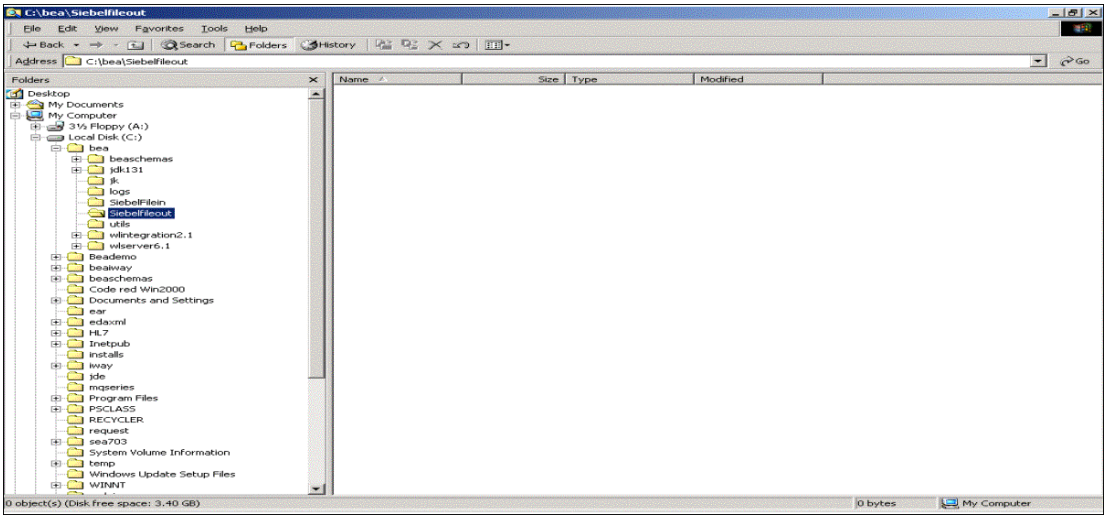
Testing a File Service

Before testing the file service (FileEmitter), verify that the target file system location is empty. For example, do the following::

1. Open Windows Explorer.

2. Select the directory where the Siebel XML file for the integration object XML for Sample Account is to be placed by the service. For example, go to `C:\bea\Siebelfileout` and verify that it is empty.

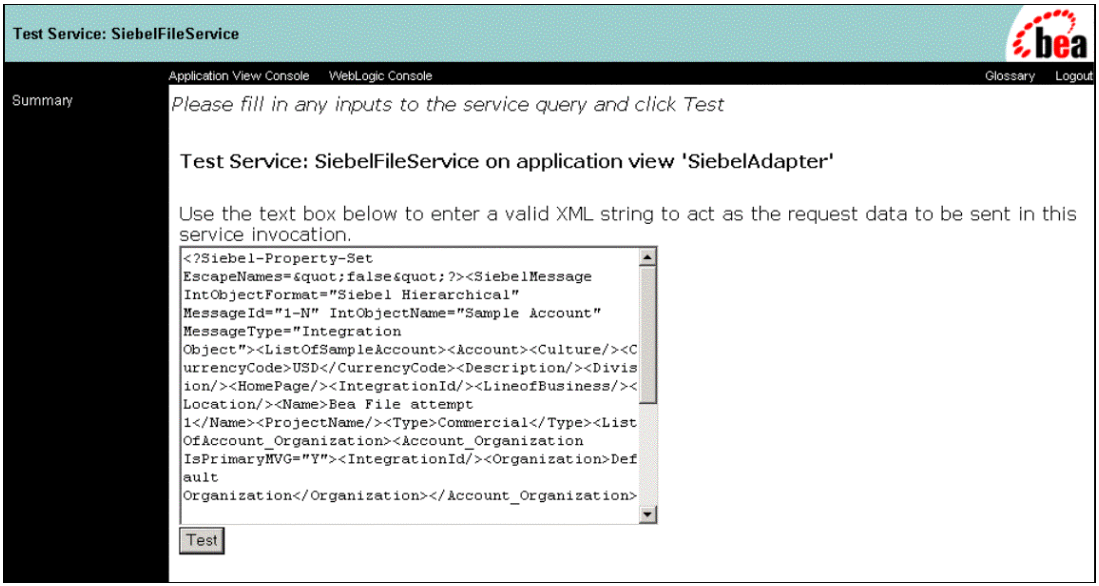
Figure 6-10 Window for Directory Verification



To text the file service,

1. In the Summary for Application View window, click the Test link for the service.
2. In the Test Service window, copy the Siebel XML file for Sample Account.

Figure 6-11 Test Service Window



3. Click Test.

   The results appear in the Test Results window.

Figure 6-12 Test Results Window

This page shows the results from testing a service.

Input to service SiebelFileService on application view SiebelAdapter

```
<?Siebel-Property-Set EscapeNames="false"?
><SiebelMessage IntObjectFormat="Siebel
Hierarchical" MessageId="1-N"
IntObjectName="Sample Account"
MessageType="Integration
Object"><ListOfSampleAccount><Account><Culture/><C
urrencyCode>USD</CurrencyCode><Description/><Divis
ion/><HomePage/><IntegrationId/><LineofBusiness/><
Location/><Name>Bea File attempt
1</Name><ProjectName/><Type>Commercial</Type><List
OfAccount_Organization><Account_Organization
IsPrimaryMVG="Y"><IntegrationId/><Organization>Def
ault
Organization</Organization></Account_Organization>
```

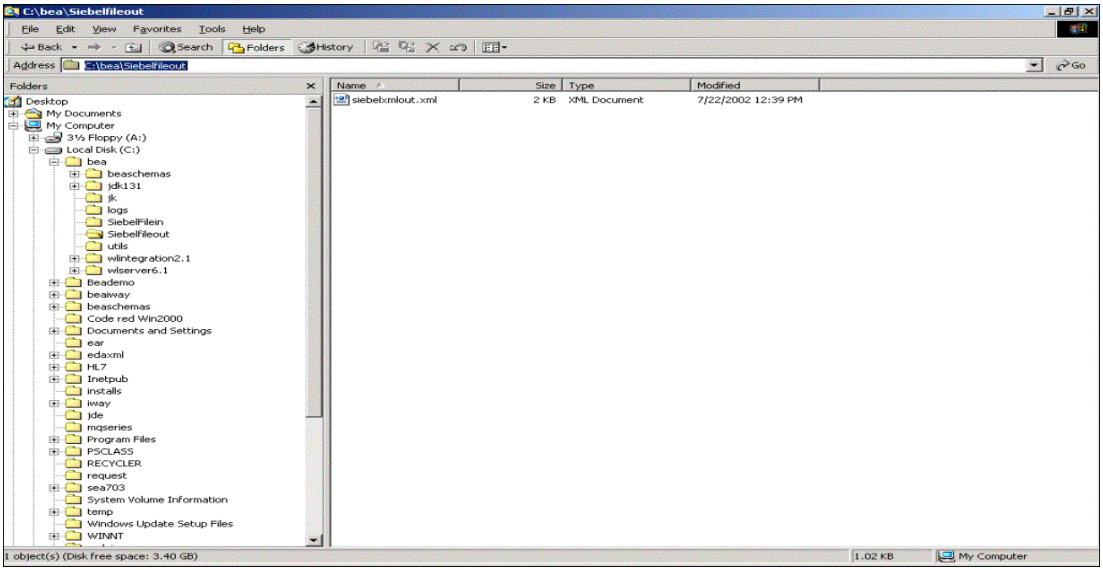
Output from service SiebelFileService on application view SiebelAdapter

```
<?xml version="1.0"?>
<!DOCTYPE emitStatus>
<emitStatus>
  <protocol>FILE</protocol>
  <parms>directory=C:\bea\Siebelfileout,
pattern=siebelxmlout.xml</parms>
  <status>0</status>
  <msg/>
  <timestamp>2002-07-22T18:39:27:952Z</timestamp>
  <attempts>1</attempts>

<name>C:\bea\Siebelfileout/siebelxmlout.xml</name>
</emitStatus>
```

4. Verify that the `c:\bea\Siebelfileout` directory now contains the Siebel XML file `siebelxmlout.xml`.

Figure 6-13 Window for Directory Display



A Siebel Workflow, configured to retrieve the file using the Siebel EAI File Transport, can now further process the XML document.

HTTP Service

An HTTP service is the process by which the adapter sends a Siebel XML file through HTTP to Siebel. This method of integration requires a Siebel Workflow within the Siebel system. The workflow interacts with the adapter as follows:

- Event. When a Siebel event occurs, the workflow sends Siebel XML to the adapter.
- Service. The workflow responds to Siebel XML received from the adapter in order to cause a Siebel business event.

For information on how to create Siebel Workflows, see Appendix B, "Creating Siebel Workflows."

After you create the application view, a business analyst can use it to create business processes that use the application. You can add any number of events to an application view.

Adding an HTTP Service to an Application View

To add an HTTP service to an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. In the Application View Console Administration Window, do one of the following to display the Add Service window:
   - In the left pane, Click Add Service.
   - In the Services section, click Add.
3. Select HTTPEmitter from the Select drop-down list.

Figure 6-14 Add Service Window

4. In the Unique Service Name field, enter a name, for example, `Sieb7HTTPQService`. The name should describe the function performed by this service.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. In the URL field, provide the URL where Siebel is listening, for example, *`http://ariba01/eai/start.swe`*.

6. In the sweextsource field, provide the name of the Siebel Workflow process to invoke, for example, `wf`.

7. In the sweextcmd field, provide the Siebel command value, for example, `Execute`.

8. Enter the login name and password in the username and password fields.

9. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All service schemas created during the session should be listed.

10. Click Add to add the service.

    The service is displayed in the Services section of the Application View Administration window.

You can now add additional services or events, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the HTTP service, you can test the service as described in the following section.

Testing an HTTP Service

To test an HTTP service:

1. In the Summary for Application View window, click Test.

   The Test Service window opens.

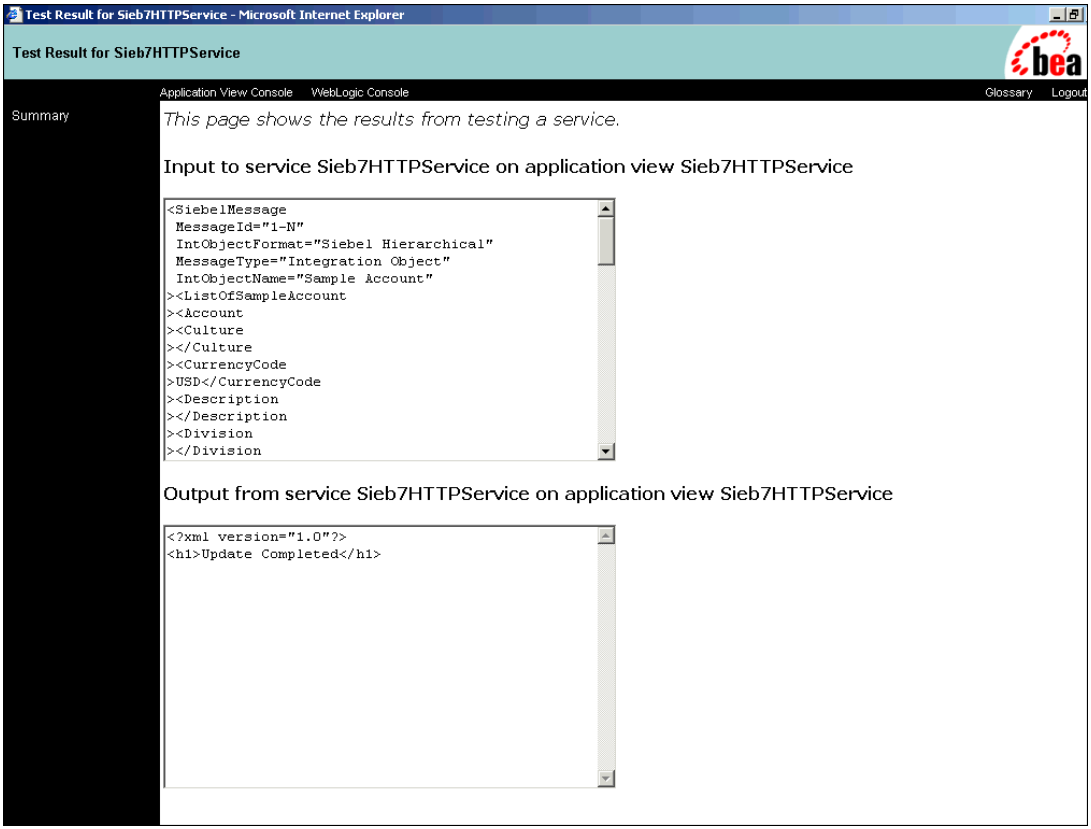
Figure 6-15 Test Service Window

2. In the Test Service window, copy the Siebel XML file for Sample Account, for example, to add an Account called HTTP Service Test.

3. Click Test.

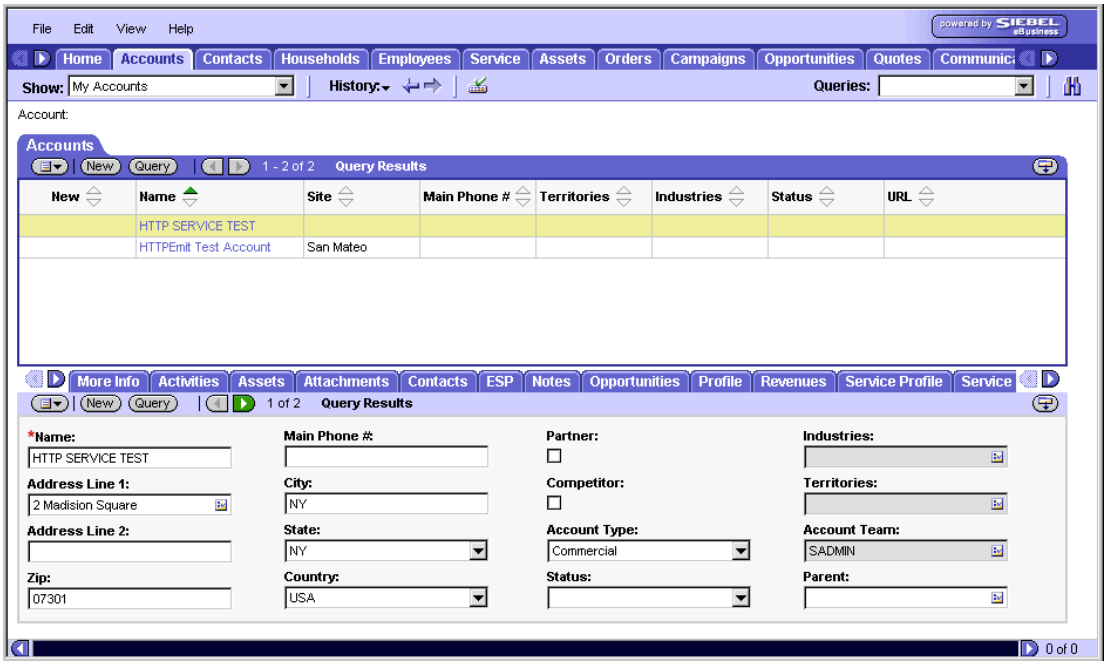
   The results appear. Note that the response indicates that the update was completed.

Figure 6-16 Test Results Window



4. Using the Siebel Client, verify that the Account has been added.

Figure 6-17 Siebel Call Center Account Window



Note that the Siebel Account HTTP Service Test has been added.

CHAPTER

7 Using Siebel Business Components and Siebel Business Services

This section explains how the BEA WebLogic Adapter for Siebel enables the processing of Siebel Business Services and Business Components using the Siebel Java Data Bean and Siebel Data COM Interface. It includes the following topics:

Overview

The BEA WebLogic Adapter for Siebel enables the processing of Siebel Business Services and Business Components. You can access and integrate with Siebel Business Services and Business Components directly; there is no need to create Siebel Workflows. There is also no need to consider a transport layer such as IBM MQSeries, File, or HTTP, since the service is accomplished through a TCP connection. The service request begins with the sending of a service request document. In most cases, the result is an XML response document signifying the execution of the business service or business component.

Creating Schemas for Siebel Business Services and Business Components

When running a service, you must create request and response schemas for the service. Use the BEA Application Explorer to generate these schemas directly against a Siebel Business Service or Siebel Business Component.

This section illustrates the steps required to create BEA schemas for Siebel Business Services and Business Components. For more information on using the BEA Application Explorer in general, see the *BEA Application Explorer Installation & Configuration Guide*.

Open BEA Application Explorer.

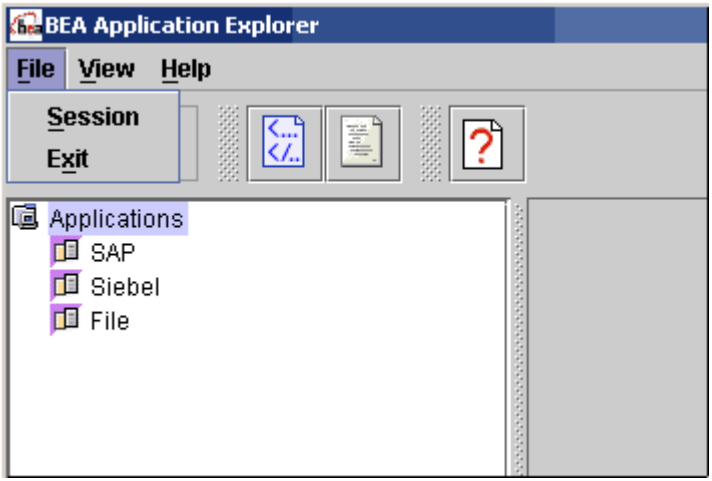
Figure 7-1 BEA Application Explorer

Establishing the Schema Working Directory

Establish the directory associated with your WebLogic Integration server to import event and service XML schemas into the application view repository.

After you invoke the BEA Application Explorer, the following screen opens.

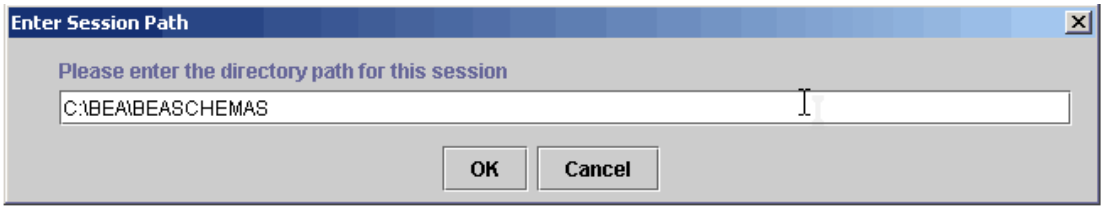
Figure 7-2 BEA Application Explorer Window



1. Choose File→Session.

   The Enter Session Path input box appears.

Figure 7-3 Enter Session Path Input Box



2. Enter the session path.

   This is where schemas are placed when they are generated by the explorer. In this example, `C:\BEA\BEASCHEMAS` is the explorer's working directory.

3. Click OK. The session path appears at the bottom of the explorer window.

Establishing a Connection to Siebel

In order to browse the Siebel Business Services, Business Components, and Integration Objects, you must create a connection to Siebel. After it is created, this connection is automatically saved. You must establish a connection to Siebel every time you start the BEA Application Explorer (BAE) or after a disconnect.

Figure 7-4 BEA Application Explorer - Establishing a New Connection

To establish a connection to Siebel:

1. Right-click Siebel and choose New Connection.

   The input box for entering a new connection name appears.

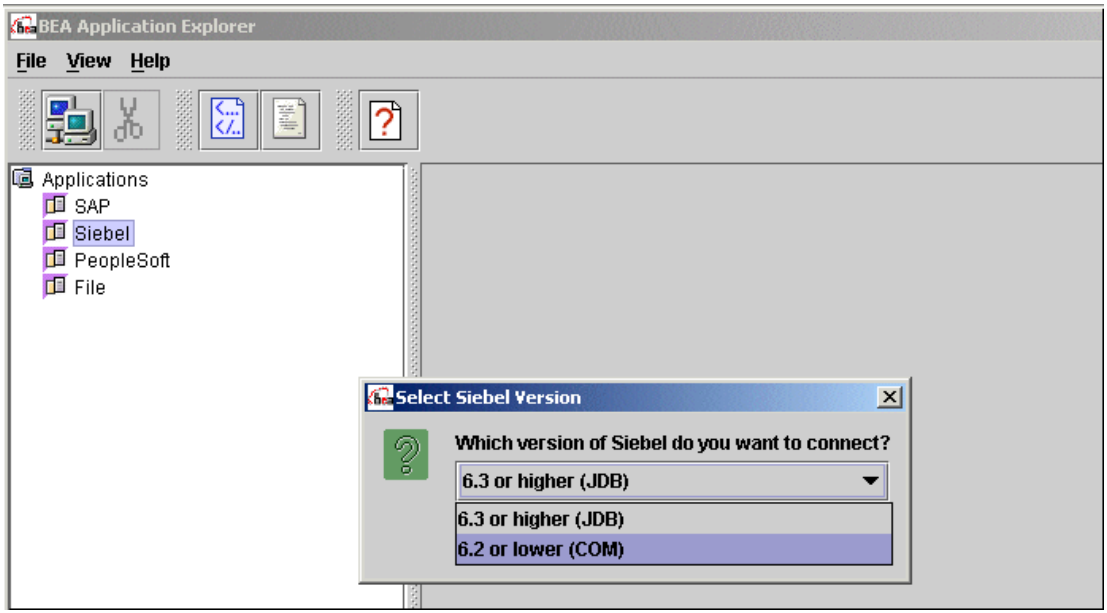
Figure 7-5 New Connection Name Input Box

   a. Enter a name for the new connection, for example, SiebelConnection.

   b. Click OK.

      The Select Siebel Version input box appears.

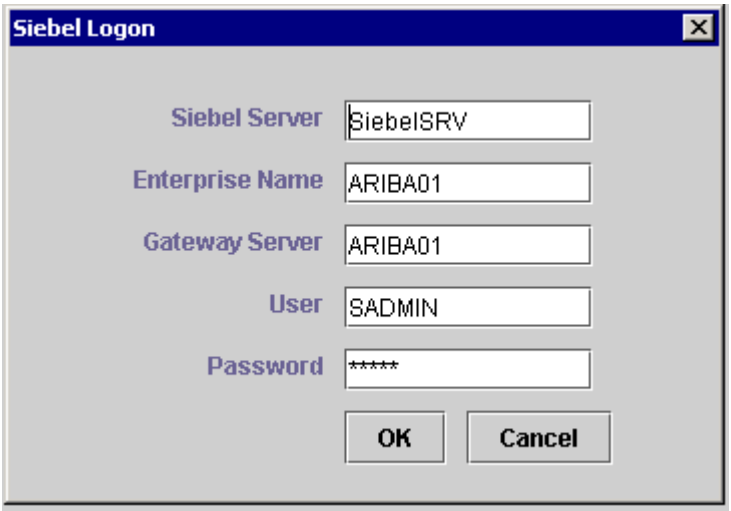
Figure 7-6 BEA Application Explorer - Select Siebel Version



2. Select the Siebel version for the connection from the drop-down box.

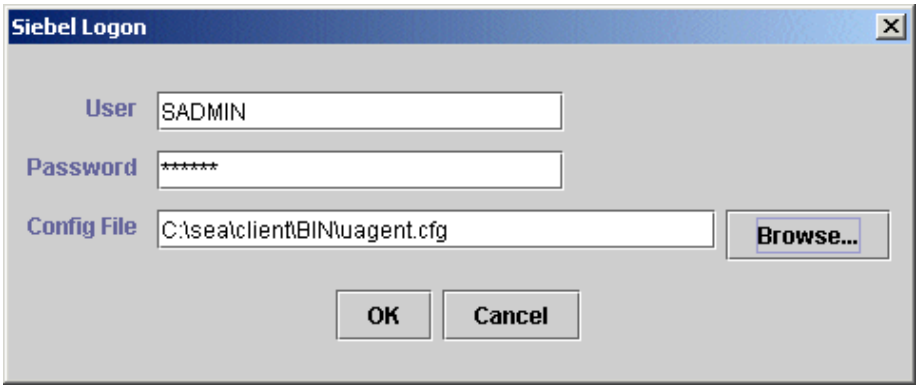
   When you select 6.3 or higher (JDB), the following logon box appears.

Figure 7-7 Logon Box for Siebel 6.3 or higher (JDB)



When you select 6.2 or lower (COM), the following logon box opens.

Figure 7-8 Logon Box for Siebel 6.2 or lower (COM)



a. Enter the parameters required to establish the connection to the Siebel system.

For Siebel 6.3 or higher (JDB):

- Siebel Server
- Enterprise Name
- Gateway Server
- User
- Password

For Siebel 6.2 or lower (COM):

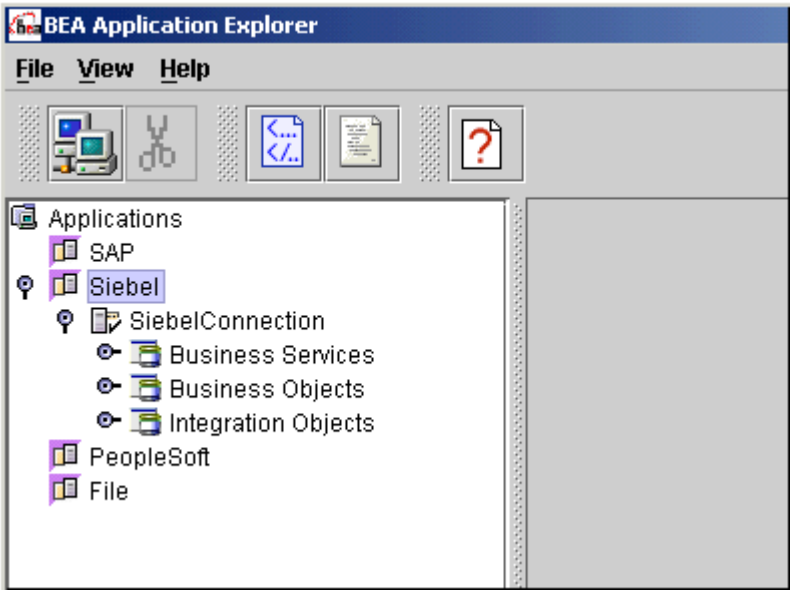
- User
- Password
- Config File

The configuration parameters supplied are those used by Siebel client applications to connect to the Siebel system. For more information about these parameters, see your Siebel documentation or ask your Siebel system administrator.

b. Click OK.

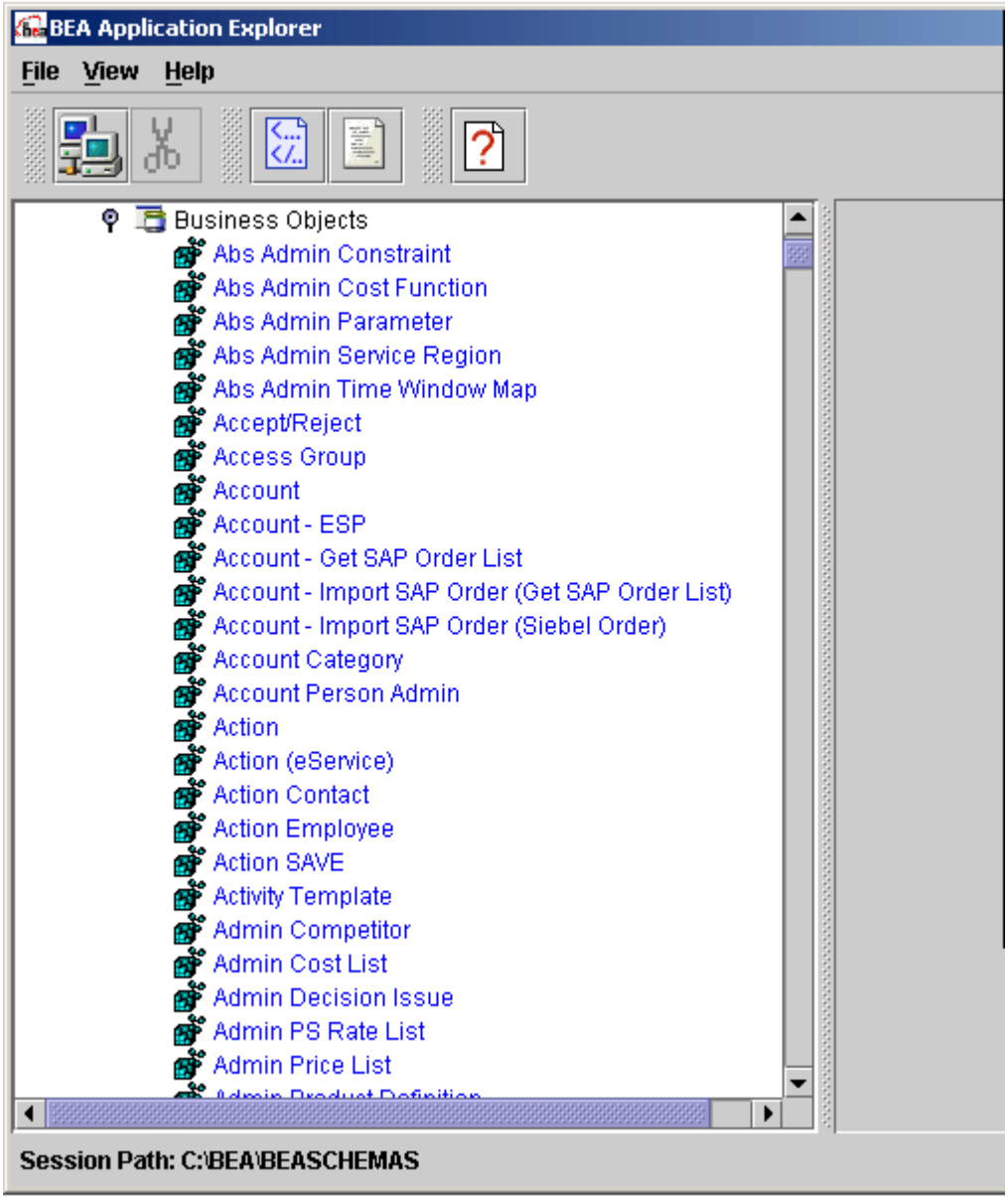
The Application Explorer connects to the Siebel system to extract business services, business objects, and integration objects.

Figure 7-9 BEA Application Explorer - Siebel Connection



You can browse all available business objects and business services in the Siebel system.

Figure 7-10 BEA Application Explorer - Business Objects



3. After a Siebel Business Service or Business Object is selected, right-click to generate service requests and response schemas.

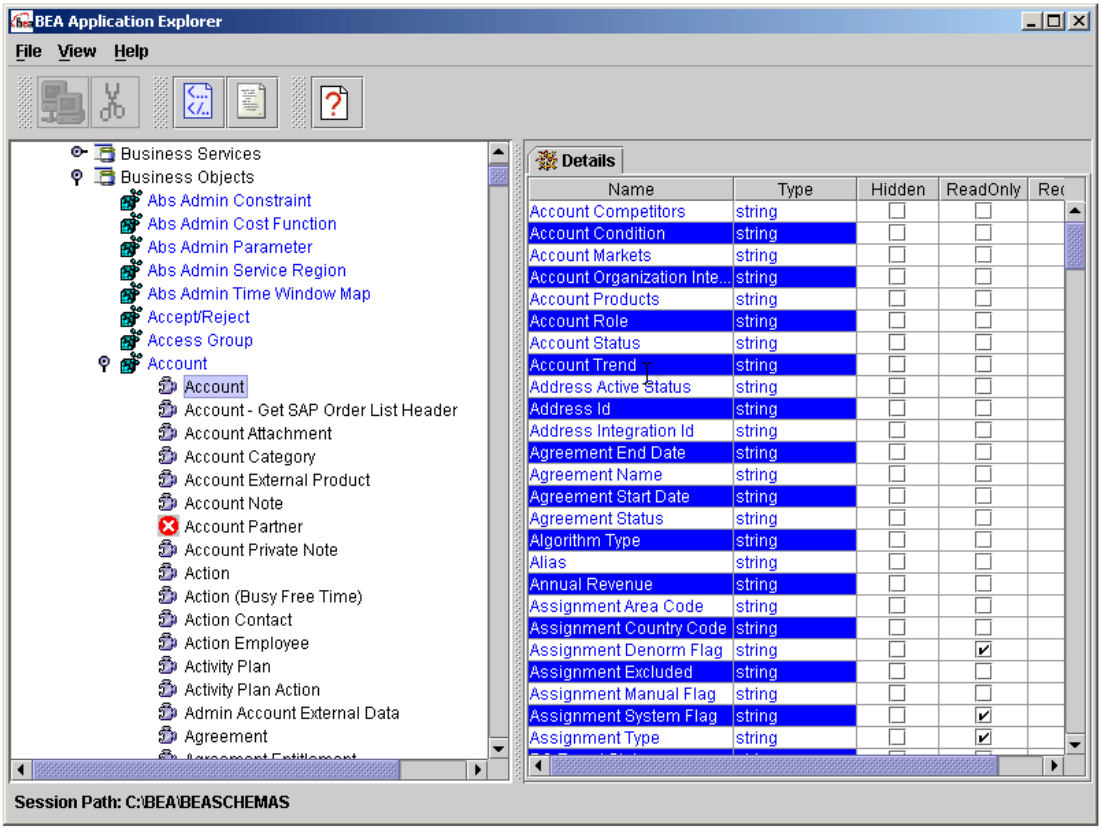
   After it is selected, BEA Application Explorer generates the following WebLogic Integration schemas:

   - Service XML request schemas
   - Service XML response schemas

Business Components

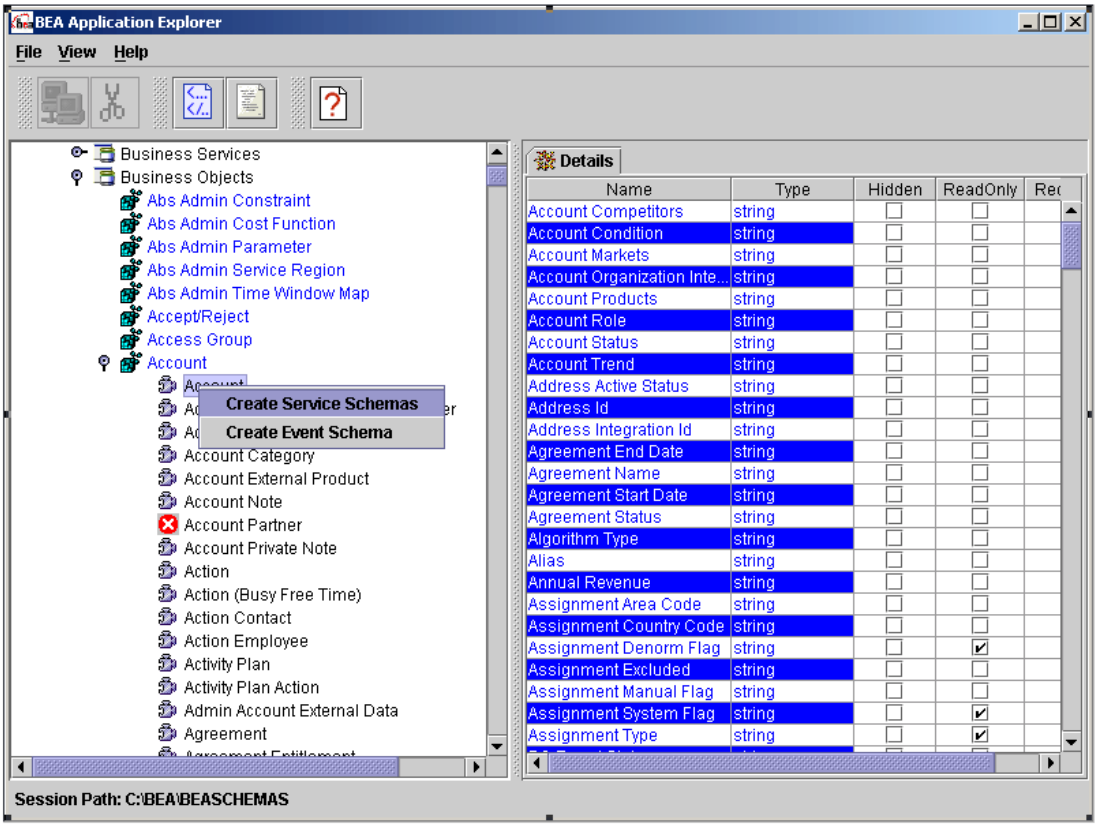
Siebel Business Objects contain one or more Siebel Business Components. Clicking a business object displays its components. In the following example, the Account business object is expanded is to display all of its business components. Note that the business component called Account is selected and displays all of its parameters, data types, and other attributes on the Details tab in the right pane.

Figure 7-11 BEA Application Explorer Window - Business Components



You can now generate service request and response schemas for the business component.

Figure 7-12 BEA Application Explorer - Schema Generation for Business Component

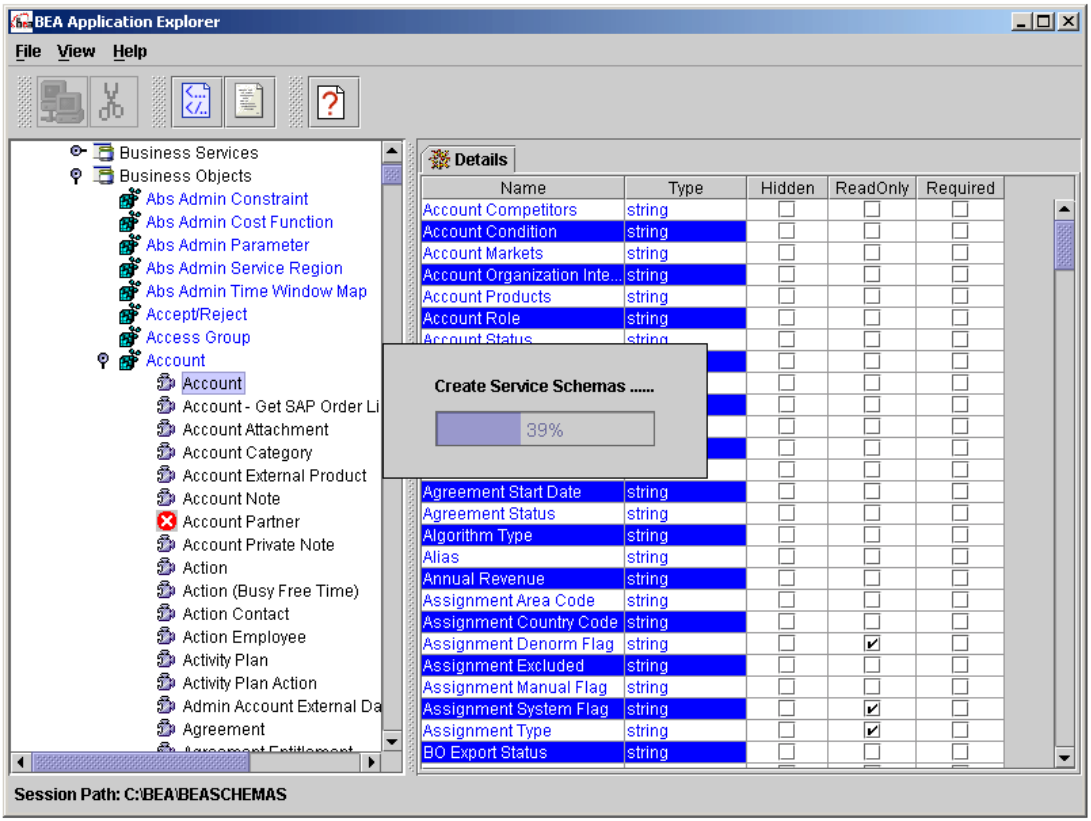


To generate a service schema for a business component:

1. Right-click the business component, Account, and select Create Service Schemas from the shortcut menu.

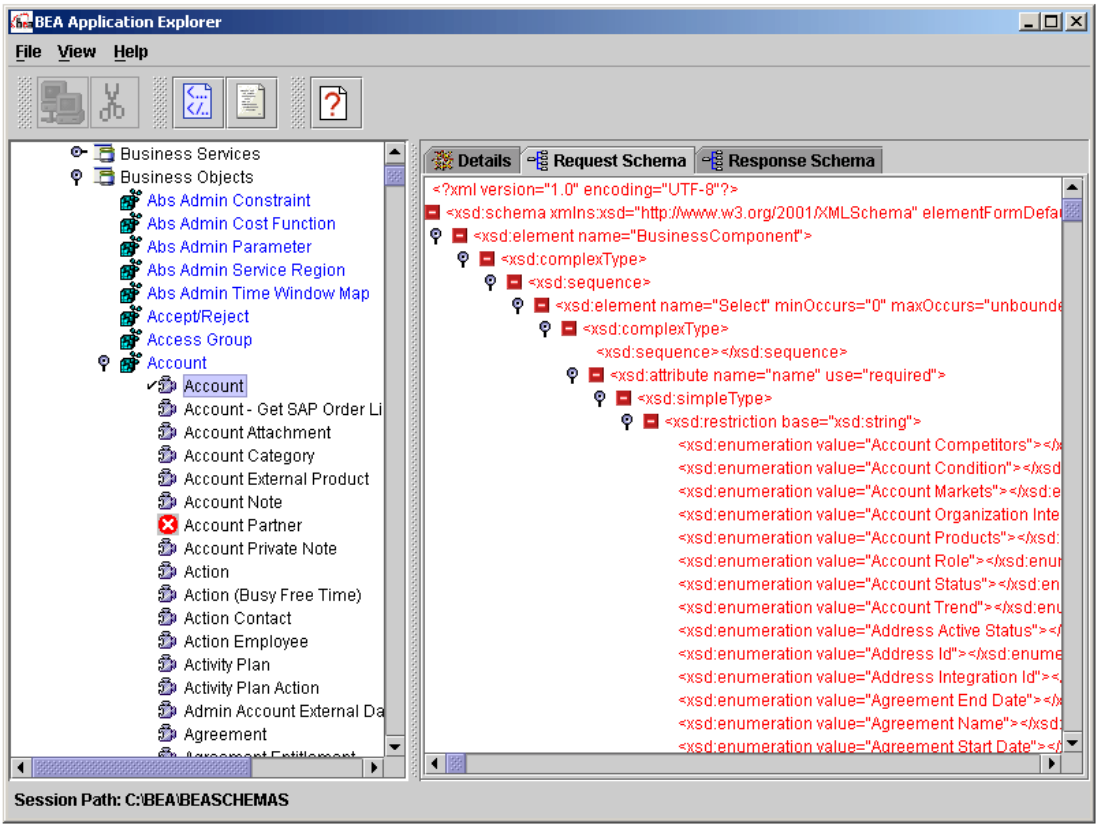
   The Application View Explorer accesses the Siebel repository and builds XSD schemas that are then published to the `C:\BEA\BEASCHEMAS` directory.

Figure 7-13 BEA Application Explorer - Service Schema Processing for Business Component



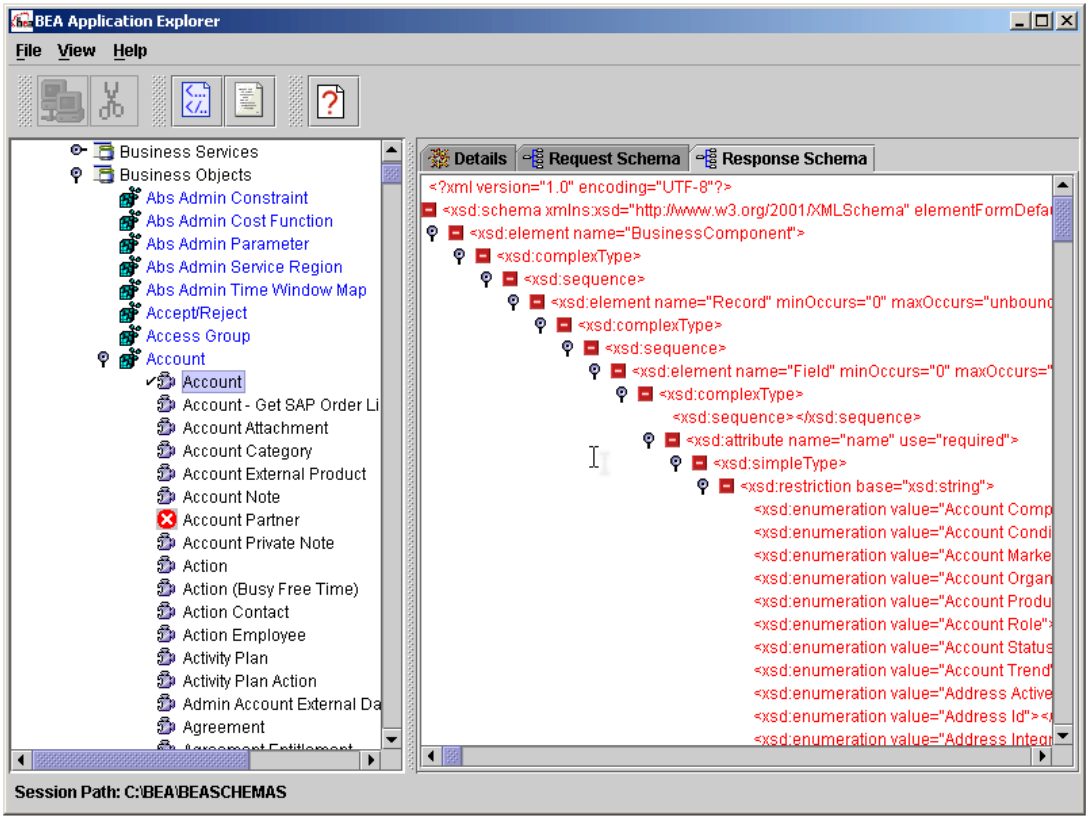
The following window displays request schema information in the right pane.

Figure 7-14 BEA Application Explorer - Request Schema for Business Component



The following window displays response schema information in the right pane.

Figure 7-15 BEA Application Explorer - Response Schema for Business Component



A directory structure is created automatically within the working directory, `C:\BEA\BEASCHEMAS`.

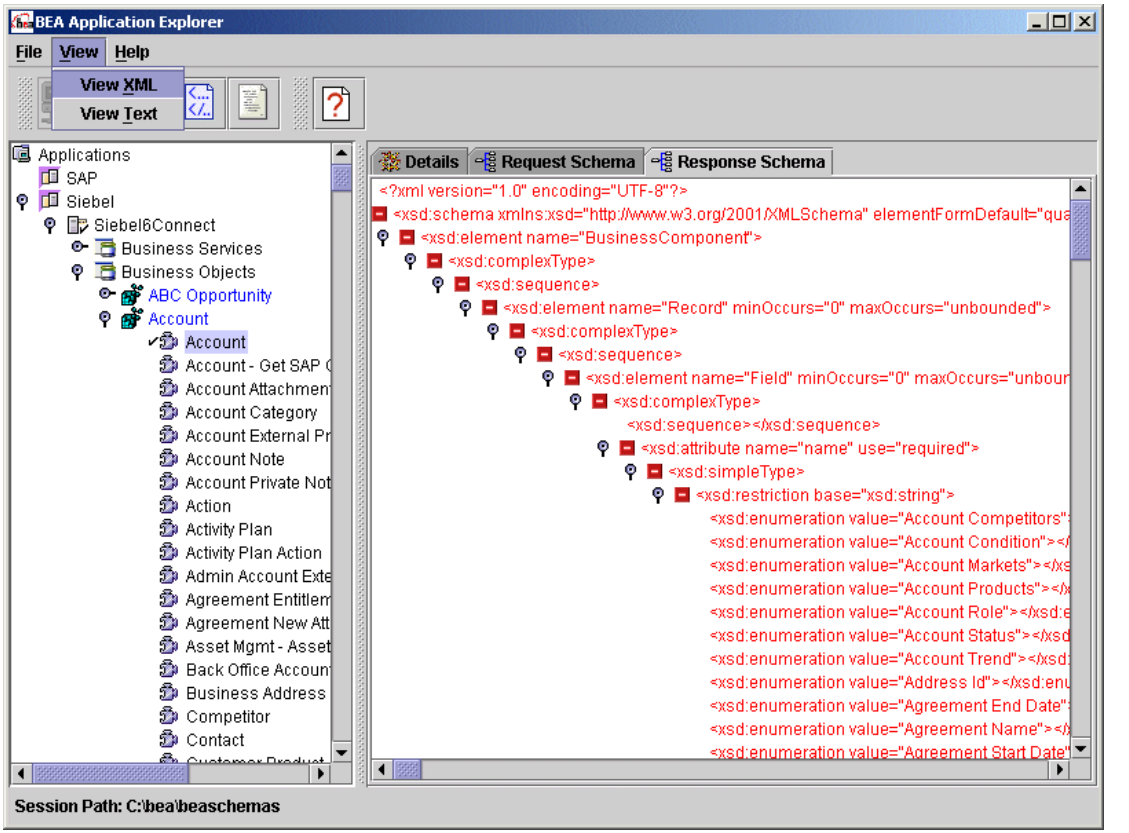
The explorer creates a folder called Siebel. It also creates subfolders for each configured Siebel connection to contain the schemas created for each connection. In this case, the schemas created are located in the folder called SiebelConnection (the connection name you established when you connected to the Siebel system using the explorer).

The following members have been added to the folder, `C:\BEA\BEASCHEMAS\Siebel\SiebelConnection`:

- `manifest.xml`
- `service_Account1-1-411E.xsd`
- `service_Account1-1-411E_response.xsd`

Use the explorer to browse the schemas published for WebLogic Integration.

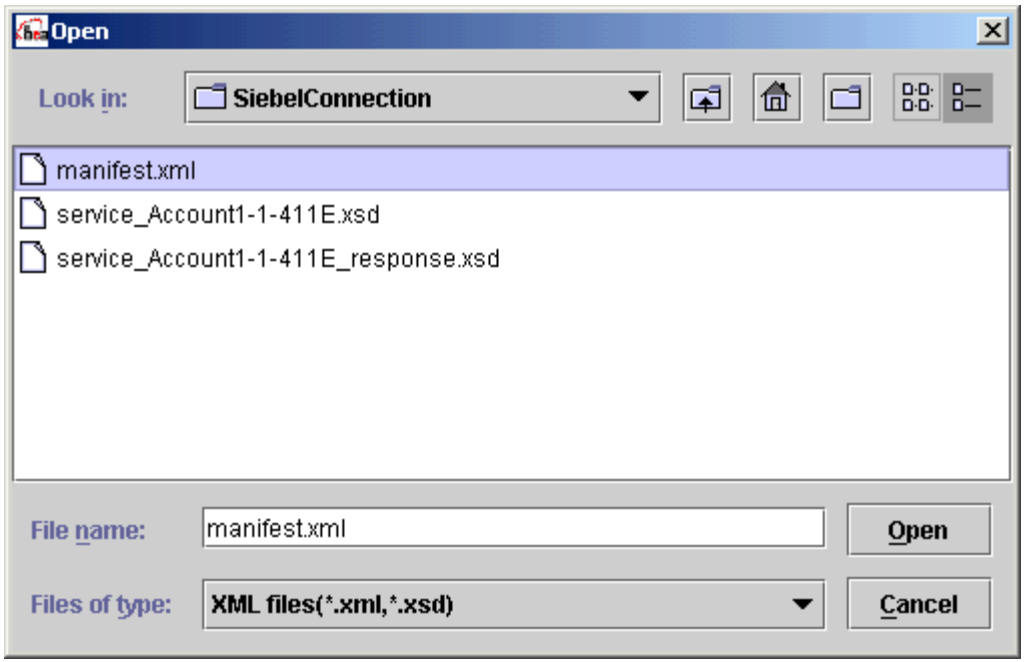
Figure 7-16 BEA Application Explorer - View XML for Business Component



2. From the View menu, choose View XML.

3. When the Open window appears, browse to the explorer working directory to select the desired XML file and to view any of the created schemas and `manifest.xml`.

Figure 7-17 Open Connection Window



a. Select the `manifest.xml` file.

b. Click Open.

For example, the `manifest.xml` file for the Account business component contains connection and configuration information established for the Siebel connection called SiebelConnection. This information is used to test access to the Siebel system when using the WebLogic Integration JSP console test pages.

Figure 7-18 manifest.xml File for a Business Component

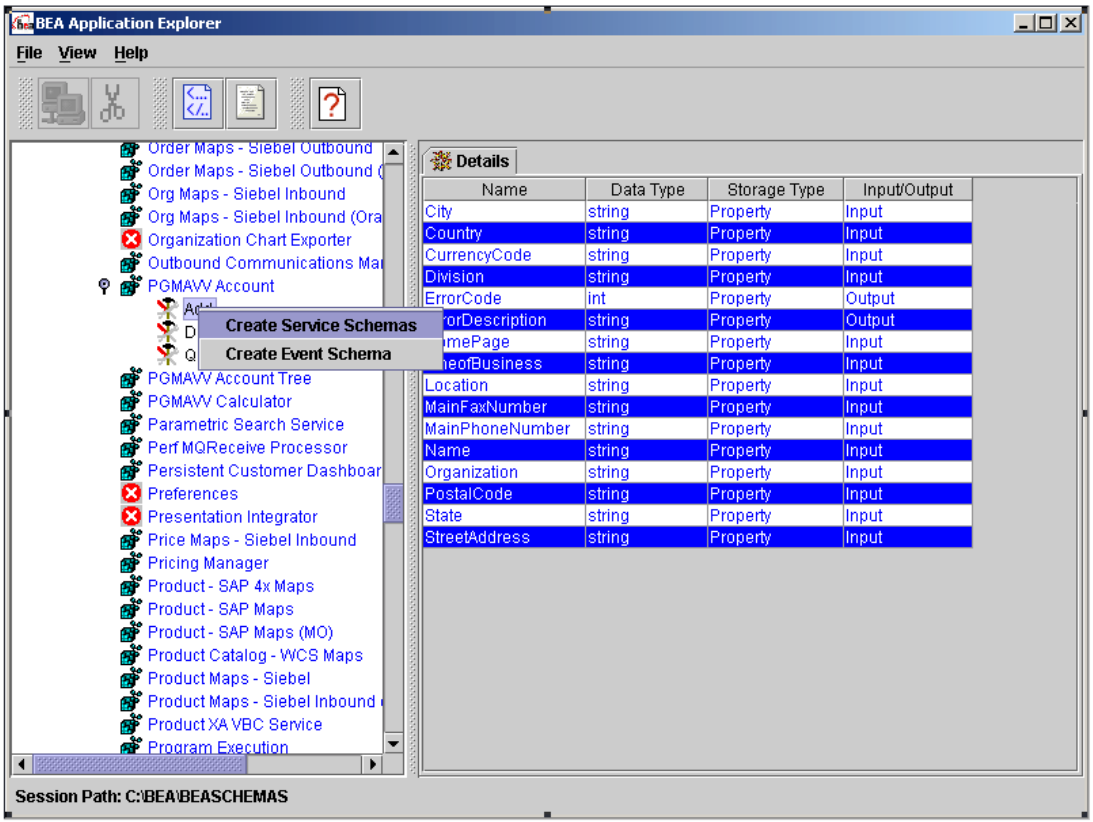
```
C:\bea\BEASCHEMAS\siebel\SiebelConnection\manifest.xml
<manifest>
  <connection>
    <user>SADMIN</user>
    <password>ENCR(241319532013226257)</password>
    <gateway>ARIBA01</gateway>
    <server>SiebelSRV</server>
    <enterprise>ARIBA01</enterprise>
  </connection>
  <schemaref name="Account1-1-411E">
    <request root="BusinessComponent" file="service_Account1-1-411E.xsd"></request>
    <response root="BusinessComponent" file="service_Account1-1-411E_response.xsd"></response
  </schemaref>
  <schemaref name="Sample Account1-1-FA22">
    <request root="SiebelMessage" file="service_Sample Account1-1-FA22.xsd"></request>
    <response root="SiebelMessage" file="service_Sample Account1-1-FA22_response.xsd"></respon
    <event root="SiebelMessage" file="event_Sample Account1-1-FA22.xsd"></event>
  </schemaref>
  <schemaref name="Execute1-1-JLHQ">
    <request root="BusinessService" file="service_Execute1-1-JLHQ.xsd"></request>
    <response root="BusinessService" file="service_Execute1-1-JLHQ_response.xsd"></response>
  </schemaref>
</manifest>
```

Business Services

The following is an example of a custom business service called PGMAVV Account that enables you to add, delete, and query against customer account data in Siebel. The BEA Application Explorer enables you to create schemas for each of these methods (for instance, the Add method, as in the following example).

After generating the schema, view the XSD schema for any given business service selected. You can generate service request and response schemas for the business service.

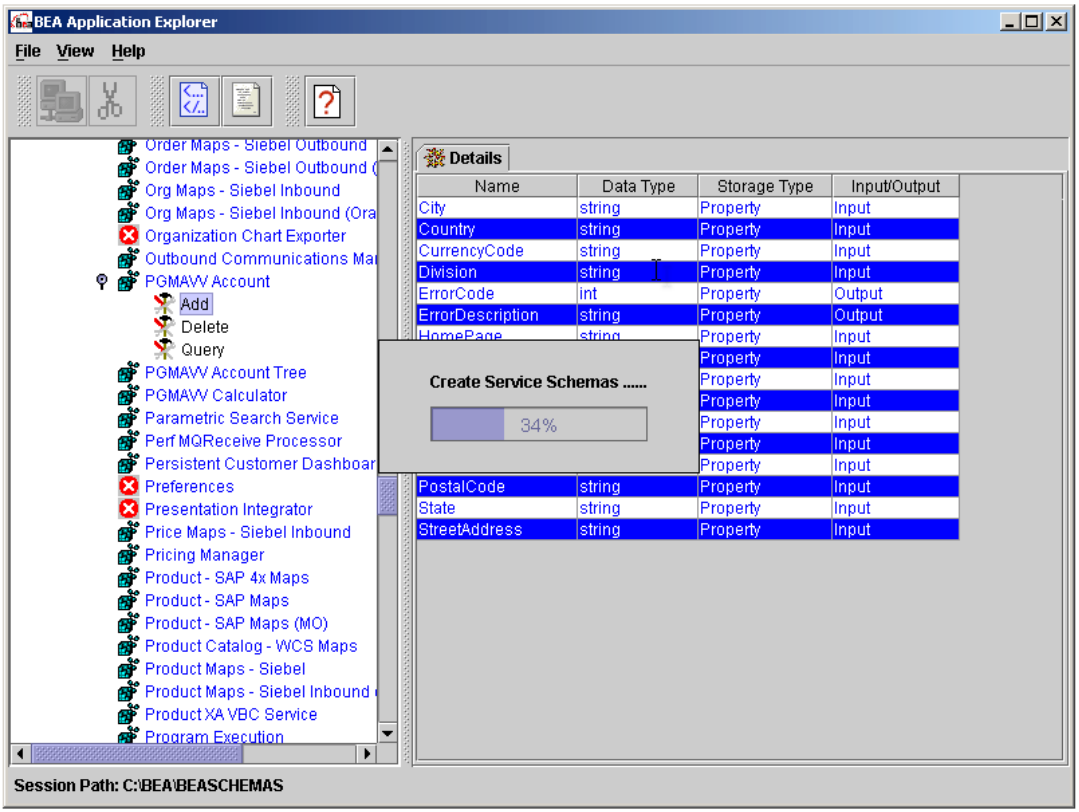
Figure 7-19 BEA Application Explorer - Schema Generation for Business Service



1. Expand PGMAVV under Business Services.
2. To generate a service schema for a business service, right-click Add and select Create Service Schemas from the shortcut menu.

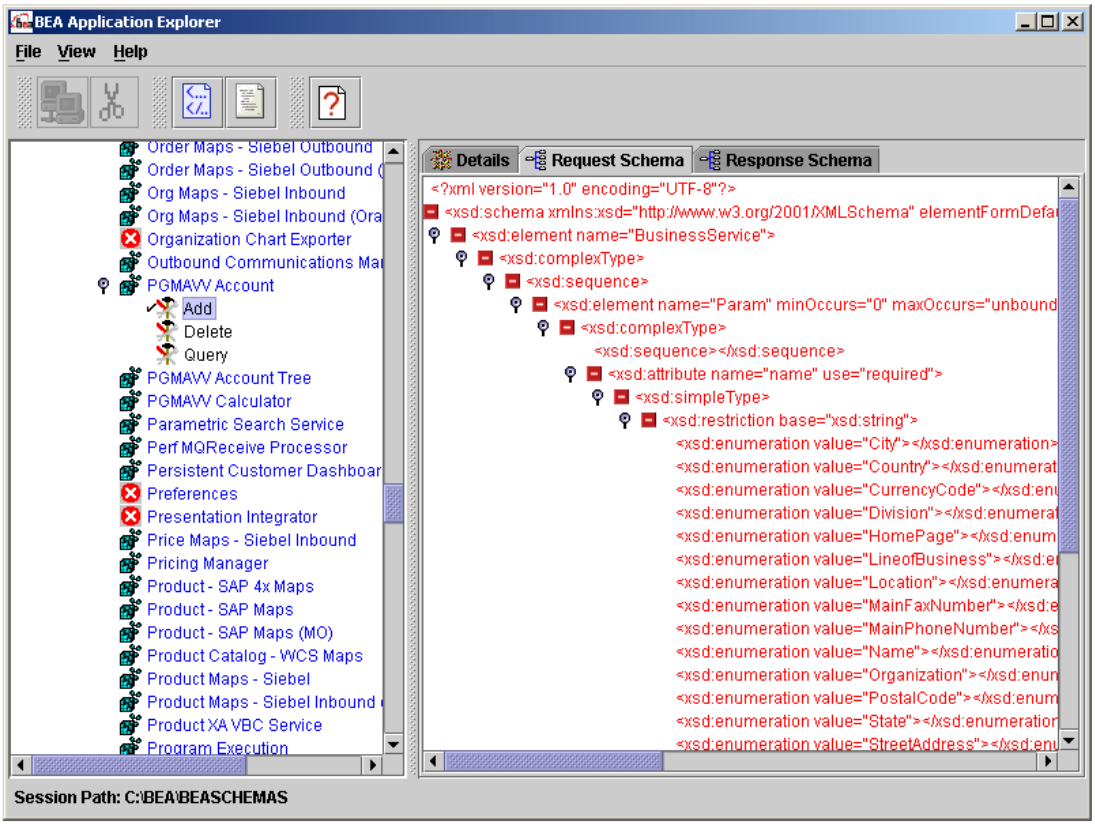
   The BEA Application Explorer accesses the Siebel repository and builds XSD schemas that are then published to the `C:\BEA\BEASCHEMAS` directory.

Figure 7-20 BEA Application Explorer - Service Schema Processing for Business Service



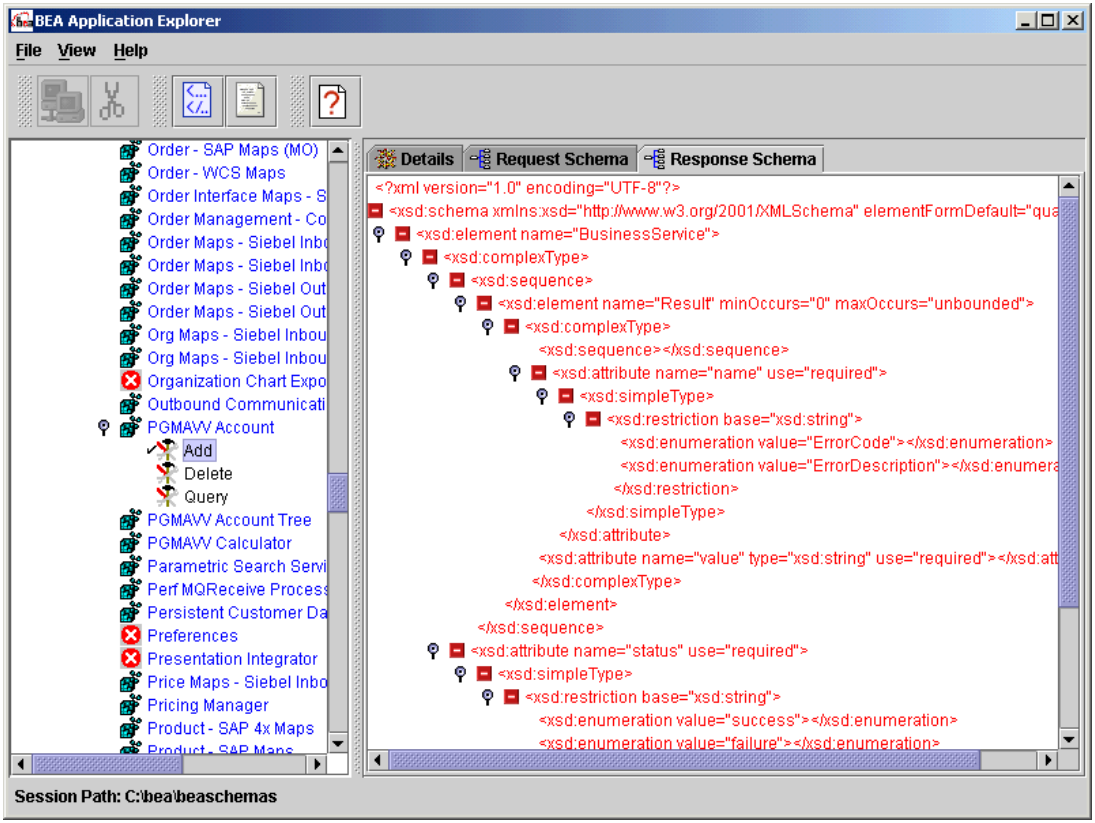
The following window displays request schema information in the right pane.

Figure 7-21 BEA Application Explorer - Request Schema for Business Service



The following window displays response schema information.

Figure 7-22 BEA Application Explorer - Response Schema for Business Service



A directory structure is created automatically within the working directory, `C:\BEA\BEASCHEMAS`.

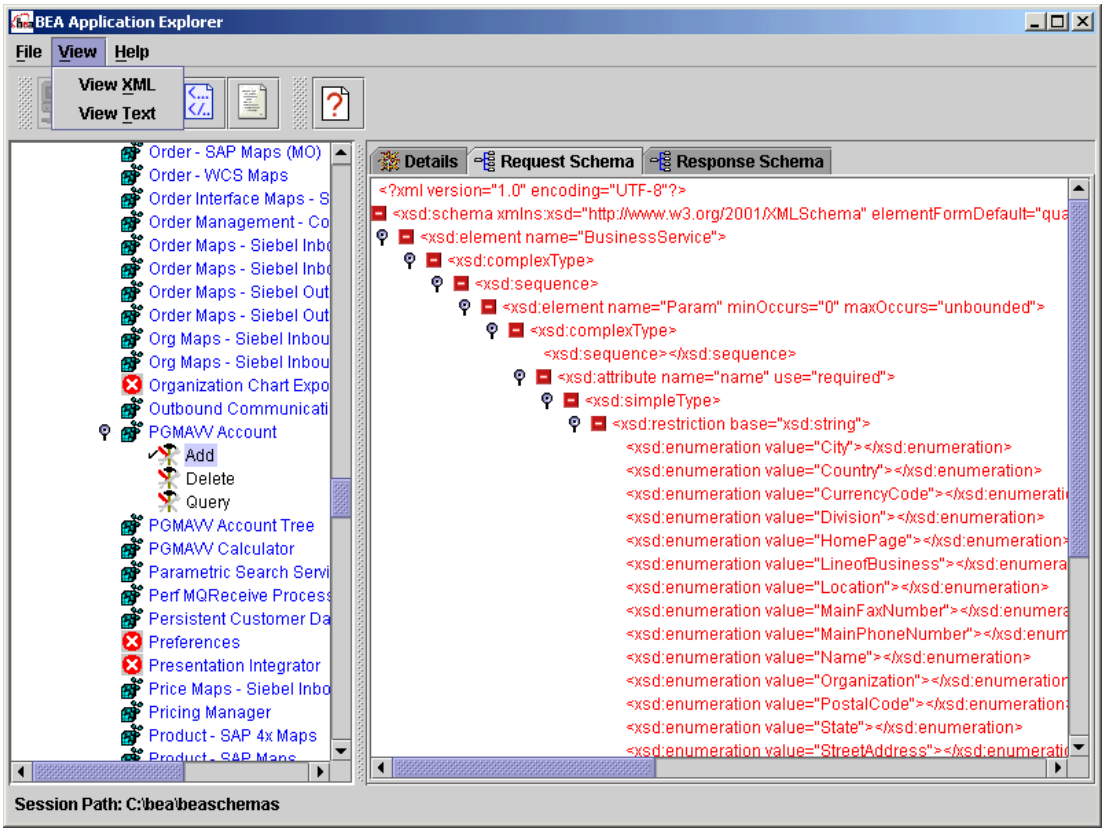
The explorer creates a folder called Siebel. It also creates subfolders for each configured Siebel connection to contain the schemas created for each connection. In this case, the schemas created are located in the folder called SiebelConnection (the connection name you established when you connected to the Siebel system using the explorer).

The following members have been added to folder, `C:\BEA\BEASCHEMAS\Siebel\SiebelConnection`:

- `manifest.xml`
- `service_Add1-6YH.xsd`
- `service_Add1-6YH_response.xsd`

Use the explorer to browse the schemas that have been published for WebLogic Integration.

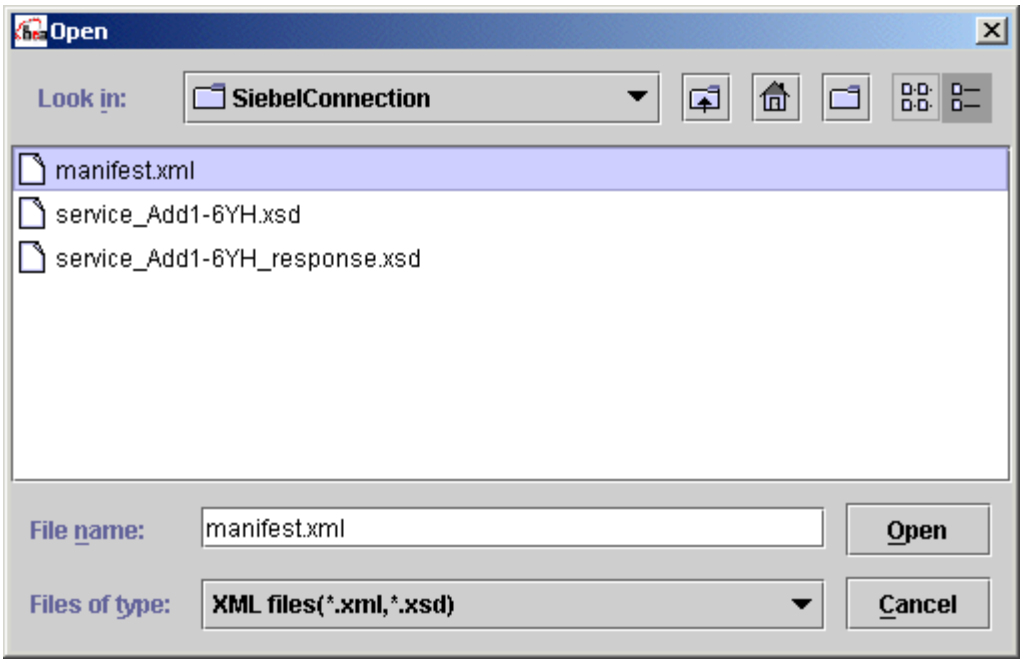
Figure 7-23 BEA Application Explorer - View XML for Business Service



3. From the View menu, choose View XML.

4. When the Open window appears, browse to the explorer working directory to select the desired XML file and to view any of the created schemas and `manifest.xml`.

Figure 7-24 Open Connection Window



5. Select the `manifest.xml` file.

For example, the `manifest.xml` file for the business service, PGMAVV Account, contains the connection and configuration information. This information is used to test access to the Siebel system when using WebLogic Integration JSP console test pages.

Once you have created the schemas required for you application view services and events, you can create an application view as described in Chapter 4, "Creating and Editing Application Views."

Java Data Bean and Siebel 6 COM Data Services

After you create the application view, as described in Chapter 4, "Creating and Editing Application Views," you can add the required Java Data Bean or Siebel 6 COM Data service to it, as described in the following section.

The type of service (Java Data Bean or COM data) you add is dependent on the version of Siebel you are using. After you have added the service to the application view, a business analyst can create business process workflows in WebLogic Integration Studio that use the service.

Adding a Java Data Bean or COM Data Service to an Application View

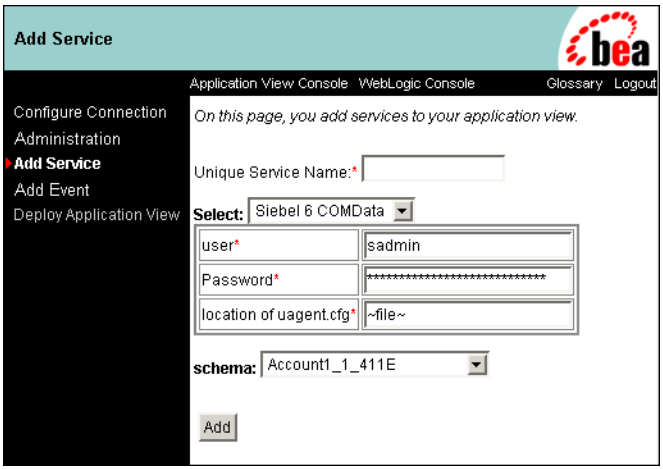
To add a Java Data Bean or COM data service to an application view:

1. If it is not already open, display the Application View Administration window as described in "Editing an Application View" on page 4-9.
2. In the Application View Console Administration Window, do one of the following to display the Add Service window:
   - In the left pane, Click Add Service.
   - In the Services section, click Add.
3. Select one of following from the Select drop-down list:
   - Siebel JavaAPI
   - Siebel 6 COMData

   You selection is dependent on the version of Siebel you are using.

Figure 7-25 Add Service Window: Siebel JavaAPI

Figure 7-26 Add Service Window: Siebel 6 COMData



4. In the Unique Service Name field, enter a name, for example, `SiebelJavaAPIService`. The name should describe the function performed by this service.

   Each service name must be unique to its application view. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

5. Enter the required values (required fields are marked with an asterisk). Descriptions of the parameters are provided in the following tables:

Table 7-1 Siebel JavaAPI Parameters

Parameter	Description
server* (*Required)	**Type/Value:** String **Description:** Name of the Siebel Application Server.
enterprise* (*Required)	**Type/Value:** String **Description:** Name of the Siebel Enterprise to which this server belongs.
gateway* (*Required)	**Type/Value:** String **Description:** Siebel gateway name server.
user*	**Type/Value:** String **Description:** Siebel administrator user name.
password*	**Type/Value:** String **Description:** Password of the administrative user.

Table 7-2 Siebel 6 COMData Parameters

Parameter	Description
user*	**Type/Value:** String **Description:** Siebel administrator user name.
password*	**Type/Value:** String **Description:** Password of the administrative user.
location of uagent.cfg	**Type/Value:** String **Description:** File system location of the `uagent.cfg` configuration file.

6. Select the appropriate schema from the drop-down list.

   The schema drop-down list corresponds to the manifest generated for you during your BEA Application Explorer session. All service schemas created during the session should be listed.

7. Click Add to add the service.

   The service is displayed in the Services section of the Application View Administration window.

You can now add additional services or events, or deploy the application view as described in "Deploying an Application View" on page 4-10.

Once you have deployed the application view containing the Java Data Bean or COM Data Interface service, you can test the service as described in the following section.

Testing a the Service

Testing evaluates whether the application view service interacts properly with the target adapter.

1. Before testing the service to add a Siebel account, verify that the account does not exist in Siebel, and then verify that the account has been added.

2. To test application view services, find the service in the Services area and click Test.

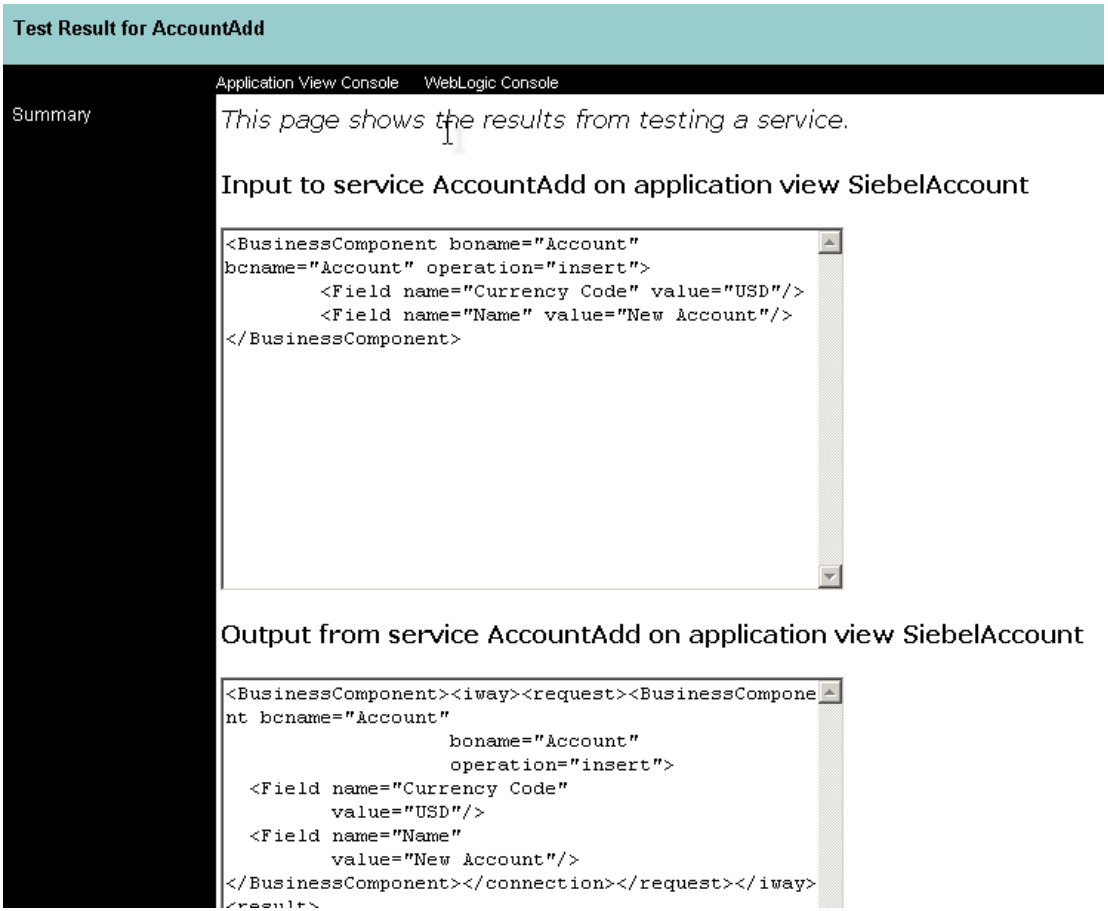
   The Test Service page opens.

Figure 7-27 Test Service Window

3. Enter the appropriate XML for the adapter. The account that you add is referenced by the name, New Account.
4. Click Test to test the service.

   The answer set returned from Siebel opens.

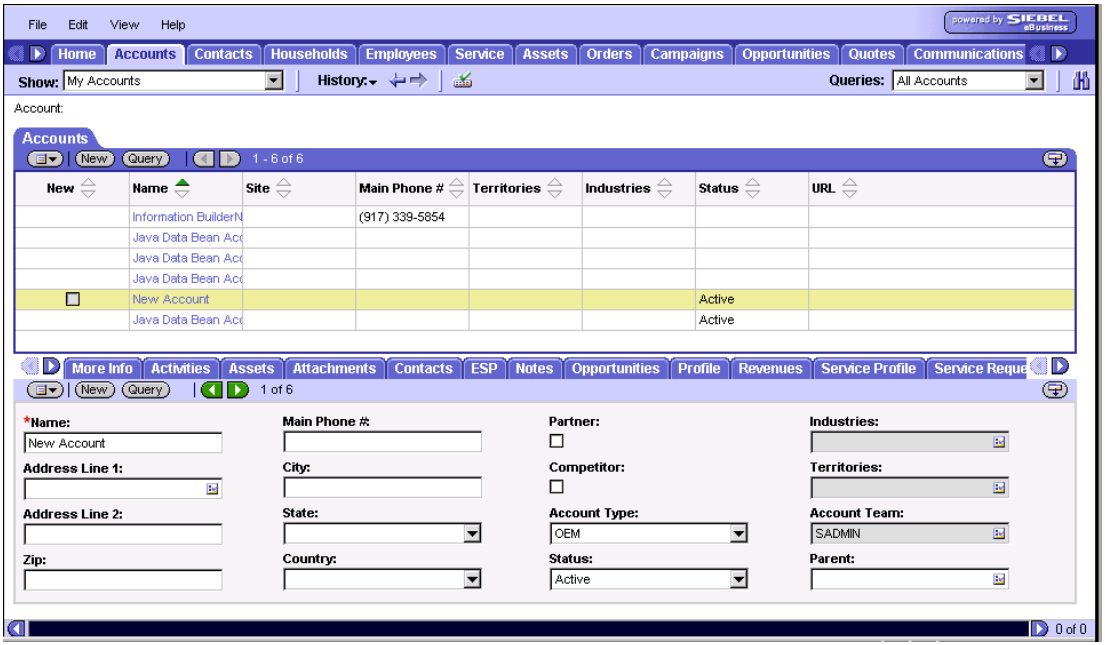
Figure 7-28 Test Result Answer Set Window



If the test fails, the Test Result page displays a timed out message.

5. Verify in Siebel that the account has been added.

Figure 7-29 Siebel Call Center Accounts Window with New Account Added



This confirms that the application view service is successfully deployed. You can now employ the service in business process workflows or write custom code. For more information, see "Using Application Views in the Studio" in *Using Application Integration*:

- For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm`
- For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/3usruse.htm`

CHAPTER

A Sample Files

This section provides sample schemas for Siebel Business Components and Siebel Business Services. It includes examples under the following topics:

- Account Business Component
- PGMAVV Account Business Service

Account Business Component

Account Request Schema

Listing A-1 Account Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="BusinessComponent">
    <xsd:complexType>
      <xsd:sequence>
<xsd:element name="Select" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Account Competitors"/>
                    <xsd:enumeration value="Account Condition"/>
                    <xsd:enumeration value="Account Markets"/>
                  <xsd:enumeration value="Account Organization Integration Id"/>
```

```
      <xsd:enumeration value="Account Products"/>
      <xsd:enumeration value="Account Role"/>
      <xsd:enumeration value="Account Status"/>
      <xsd:enumeration value="Account Trend"/>
      <xsd:enumeration value="Address Active Status"/>
      <xsd:enumeration value="Address Id"/>
      <xsd:enumeration value="Address Integration Id"/>
      <xsd:enumeration value="Agreement End Date"/>
      <xsd:enumeration value="Agreement Name"/>
      <xsd:enumeration value="Agreement Start Date"/>
      <xsd:enumeration value="Agreement Status"/>
      <xsd:enumeration value="Algorithm Type"/>
      <xsd:enumeration value="Alias"/>
      <xsd:enumeration value="Annual Revenue"/>
      <xsd:enumeration value="Assignment Area Code"/>
      <xsd:enumeration value="Assignment Country Code"/>
      <xsd:enumeration value="Assignment Excluded"/>
      <xsd:enumeration value="Assignment Manual Flag"/>
      <xsd:enumeration value="Back Office Distribution Channel"/>
      <xsd:enumeration value="Back Office Order Query End  Dt"/>
      <xsd:enumeration value="Back Office Order Query Start  Dt"/>
    <xsd:enumeration value="Back Office Sales Area Division Code"/>
      <xsd:enumeration value="Back Office Sales Organization"/>
      <xsd:enumeration value="Bill Address Flag"/>
      <xsd:enumeration value="Bill To City"/>
      <xsd:enumeration value="Bill To Country"/>
      <xsd:enumeration value="Bill To First Name"/>
      <xsd:enumeration value="Bill To Id"/>
      <xsd:enumeration value="Bill To Job Title"/>
      <xsd:enumeration value="Bill To Last Name"/>
      <xsd:enumeration value="Bill To Postal Code"/>
      <xsd:enumeration value="Bill To State"/>
      <xsd:enumeration value="Bill To Street Address"/>
      <xsd:enumeration value="Block Credit Flag"/>
      <xsd:enumeration value="Business Profile"/>
      <xsd:enumeration value="CSN"/>
      <xsd:enumeration value="City"/>
      <xsd:enumeration value="City State"/>
      <xsd:enumeration value="Competitor"/>
      <xsd:enumeration value="Country"/>
      <xsd:enumeration value="County"/>
      <xsd:enumeration value="Credit Control Area Code"/>
      <xsd:enumeration value="Credit Currency Code"/>
      <xsd:enumeration value="Credit Limit Amount"/>
      <xsd:enumeration value="Credit Profile Id"/>
      <xsd:enumeration value="Culture"/>
      <xsd:enumeration value="Currency Code"/>
      <xsd:enumeration value="Current Volume"/>
      <xsd:enumeration value="Current Volume Currency Code"/>
```

```
<xsd:enumeration value="Current Volume Exchange Date"/>
<xsd:enumeration value="Customer Account Group"/>
<xsd:enumeration value="DNBReport"/>
<xsd:enumeration value="DUNS Intcode"/>
<xsd:enumeration value="DUNS Number"/>
<xsd:enumeration value="Date Formed"/>
<xsd:enumeration value="DeDup Key Modification Date"/>
<xsd:enumeration value="DeDup Key Update"/>
<xsd:enumeration value="DeDup Keys"/>
<xsd:enumeration value="DeDup Last Match Date"/>
<xsd:enumeration value="DeDup Token"/>
<xsd:enumeration value="Deduplication Match Score"/>
<xsd:enumeration value="Deduplication Object Id"/>
<xsd:enumeration value="Description"/>
<xsd:enumeration value="Disable DataCleansing"/>
<xsd:enumeration value="Division"/>
<xsd:enumeration value="Domestic Ultimate DUNS"/>
<xsd:enumeration value="Dummy"/>
<xsd:enumeration value="EAI Sync Date"/>
<xsd:enumeration value="EAI Sync Error Text"/>
<xsd:enumeration value="EAI Sync Status Code"/>
<xsd:enumeration value="Email Address"/>
<xsd:enumeration value="Employee Here"/>
<xsd:enumeration value="Employees"/>
<xsd:enumeration value="Expertise"/>
<xsd:enumeration value="Explorer Label"/>
<xsd:enumeration value="Fax Number"/>
<xsd:enumeration value="Fiscal Year End"/>
<xsd:enumeration value="Freight Terms"/>
<xsd:enumeration value="Freight Terms Info"/>
<xsd:enumeration value="Full Address"/>
<xsd:enumeration value="GSA Flag"/>
<xsd:enumeration value="Global Ultimate DUNS"/>
<xsd:enumeration value="Goals"/>
<xsd:enumeration value="Group Type Code"/>
<xsd:enumeration value="Home Page"/>
<xsd:enumeration value="Industry Condition"/>
<xsd:enumeration value="Industry Trend"/>
<xsd:enumeration value="Integration Id"/>
<xsd:enumeration value="Internal Org Flag"/>
<xsd:enumeration value="Joined Synonym"/>
<xsd:enumeration value="Key Competitors"/>
<xsd:enumeration value="Language Code"/>
<xsd:enumeration value="Last Clnse Date"/>
<xsd:enumeration value="Last Manager Review Date"/>
<xsd:enumeration value="Last Review Manager Id"/>
<xsd:enumeration value="Line of Business"/>
<xsd:enumeration value="Location"/>
<xsd:enumeration value="Location Level"/>
```

```
<xsd:enumeration value="Main Address Flag"/>
<xsd:enumeration value="Main Fax Number"/>
<xsd:enumeration value="Main Phone Number"/>
<xsd:enumeration value="Managers Review"/>
<xsd:enumeration value="Marketing"/>
<xsd:enumeration value="Merge Sequence Number"/>
<xsd:enumeration value="Mission"/>
<xsd:enumeration value="Name"/>
<xsd:enumeration value="Name and Location"/>
<xsd:enumeration value="Not Manager Flag"/>
<xsd:enumeration value="Notes"/>
<xsd:enumeration value="Objectives"/>
<xsd:enumeration value="Organization Id"/>
<xsd:enumeration value="Organization Integration Id"/>
<xsd:enumeration value="Our Position"/>
<xsd:enumeration value="Outline Number"/>
<xsd:enumeration value="PO Approved Flag"/>
<xsd:enumeration value="PO Auto Approval Currency Code"/>
<xsd:enumeration value="PO Auto Approval Date"/>
<xsd:enumeration value="PO Auto Approval Limit"/>
<xsd:enumeration value="Parent Account Division"/>
<xsd:enumeration value="Parent Account Id"/>
<xsd:enumeration value="Parent Account Integration Id"/>
<xsd:enumeration value="Parent Account Location"/>
<xsd:enumeration value="Parent Account Location Level"/>
<xsd:enumeration value="Parent Account Name"/>
<xsd:enumeration value="Parent Account Region"/>
<xsd:enumeration value="Parent HQ DUNS"/>
<xsd:enumeration value="Partner Flag"/>
<xsd:enumeration value="Partners"/>
<xsd:enumeration value="Party Name"/>
<xsd:enumeration value="Party Type Code"/>
<xsd:enumeration value="Party UId"/>
<xsd:enumeration value="Philosophy"/>
<xsd:enumeration value="Phone Number"/>
<xsd:enumeration value="Position Integration Id"/>
<xsd:enumeration value="Postal Code"/>
<xsd:enumeration value="Price List"/>
<xsd:enumeration value="Price List End Date"/>
<xsd:enumeration value="Price List Id"/>
<xsd:enumeration value="Price List Integration Id"/>
<xsd:enumeration value="Price List Start Date"/>
<xsd:enumeration value="Primary Account City"/>
<xsd:enumeration value="Primary Account Country"/>
<xsd:enumeration value="Primary Account Postal Code"/>
<xsd:enumeration value="Primary Account State"/>
<xsd:enumeration value="Primary Account Street Address"/>
<xsd:enumeration value="Primary Address Id"/>
<xsd:enumeration value="Primary Assignment Denorm Flag"/>
```

```
        <xsd:enumeration value="Primary Assignment Manual Flag"/>
        <xsd:enumeration value="Primary Assignment System Flag"/>
        <xsd:enumeration value="Primary Assignment Type"/>
        <xsd:enumeration value="Primary Bill To Address Id"/>
        <xsd:enumeration value="Primary Bill To City"/>
        <xsd:enumeration value="Primary Bill To Country"/>
        <xsd:enumeration value="Primary Bill To First Name"/>
        <xsd:enumeration value="Primary Bill To Job Title"/>
        <xsd:enumeration value="Primary Bill To Last Name"/>
        <xsd:enumeration value="Primary Bill To Person Id"/>
        <xsd:enumeration value="Primary Bill To Postal Code"/>
        <xsd:enumeration value="Primary Bill To State"/>
        <xsd:enumeration value="Primary Bill To Street Address"/>
        <xsd:enumeration value="Primary Category Id"/>
    <xsd:enumeration value="Primary Fulfill InvLoc Integration Id"/>
        <xsd:enumeration value="Primary Fulfillment InvLoc ID"/>
    <xsd:enumeration value="Primary Fulfillment Inventory Location"/>
        <xsd:enumeration value="Primary Industry Id"/>
        <xsd:enumeration value="Primary Organization"/>
        <xsd:enumeration value="Primary Organization Id"/>
        <xsd:enumeration value="Primary Payer Account"/>
        <xsd:enumeration value="Primary Payer Account Id"/>
        <xsd:enumeration value="Primary Position Id"/>
        <xsd:enumeration value="Primary Service Agreement Id"/>
        <xsd:enumeration value="Primary Ship To Address Id"/>
        <xsd:enumeration value="Primary Ship To City"/>
        <xsd:enumeration value="Primary Ship To Country"/>
        <xsd:enumeration value="Primary Ship To First Name"/>
        <xsd:enumeration value="Primary Ship To Job Title"/>
        <xsd:enumeration value="Primary Ship To Last Name"/>
        <xsd:enumeration value="Primary Ship To Person Id"/>
        <xsd:enumeration value="Primary Ship To Postal Code"/>
        <xsd:enumeration value="Primary Ship To State"/>
        <xsd:enumeration value="Primary Ship To Street Address"/>
        <xsd:enumeration value="Primary Synonym Id"/>
        <xsd:enumeration value="Primary Territory Id"/>
        <xsd:enumeration value="Primary Type Id"/>
        <xsd:enumeration value="Profit"/>
        <xsd:enumeration value="Project Bill Type"/>
        <xsd:enumeration value="Project Comments"/>
        <xsd:enumeration value="Project Fix Fee"/>
        <xsd:enumeration value="Project Hour Limit"/>
        <xsd:enumeration value="Project Id"/>
        <xsd:enumeration value="Project Name"/>
        <xsd:enumeration value="Project Percentage of Fee"/>
        <xsd:enumeration value="Project Purchase Order"/>
        <xsd:enumeration value="Project Relationship Type"/>
        <xsd:enumeration value="Project Role"/>
        <xsd:enumeration value="Prospect Flag"/>
```

```
            <xsd:enumeration value="Province"/>
            <xsd:enumeration value="Public"/>
            <xsd:enumeration value="Reference Date"/>
            <xsd:enumeration value="Reference Flag"/>
            <xsd:enumeration value="Reference Stage"/>
            <xsd:enumeration value="Region"/>
            <xsd:enumeration value="Relationship Level"/>
            <xsd:enumeration value="Relationship Type"/>
            <xsd:enumeration value="Response Time"/>
            <xsd:enumeration value="Revenue"/>
            <xsd:enumeration value="Revenue Growth"/>
            <xsd:enumeration value="Revision Number"/>
            <xsd:enumeration value="Row Status"/>
            <xsd:enumeration value="Row Status Asterisk"/>
            <xsd:enumeration value="S-S Instance"/>
            <xsd:enumeration value="S-S Instance Id"/>
            <xsd:enumeration value="S-S Key Id"/>
            <xsd:enumeration value="Service Calendar"/>
            <xsd:enumeration value="Service Type"/>
            <xsd:enumeration value="Ship Address Flag"/>
            <xsd:enumeration value="Ship To City"/>
            <xsd:enumeration value="Ship To Country"/>
            <xsd:enumeration value="Ship To First Name"/>
            <xsd:enumeration value="Ship To Job Title"/>
            <xsd:enumeration value="Ship To Last Name"/>
            <xsd:enumeration value="Ship To Postal Code"/>
            <xsd:enumeration value="Ship To State"/>
            <xsd:enumeration value="Ship To Street Address"/>
            <xsd:enumeration value="Start Date"/>
            <xsd:enumeration value="State"/>
            <xsd:enumeration value="Strategies"/>
            <xsd:enumeration value="Strategy"/>
            <xsd:enumeration value="Street Address"/>
            <xsd:enumeration value="Street Address 2"/>
            <xsd:enumeration value="Success Factors"/>
            <xsd:enumeration value="Synonym"/>
            <xsd:enumeration value="Territory"/>
            <xsd:enumeration value="Territory Id"/>
            <xsd:enumeration value="Timestamp"/>
            <xsd:enumeration value="Today"/>
            <xsd:enumeration value="Total Potential Volume"/>
          <xsd:enumeration value="Total Potential Volume Currency Code"/>
          <xsd:enumeration value="Total Potential Volume Exchange Date"/>
            <xsd:enumeration value="Type"/>
            <xsd:enumeration value="Type MVF"/>
            <xsd:enumeration value="VAT registration number"/>
            <xsd:enumeration value="Value Proposition"/>
        </xsd:restriction>
    </xsd:simpleType>
```

```
        </xsd:attribute>
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="Field" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Account Competitors"/>
                    <xsd:enumeration value="Account Condition"/>
                    <xsd:enumeration value="Account Markets"/>
                   <xsd:enumeration value="Account Organization Integration Id"/>
                    <xsd:enumeration value="Account Products"/>
                    <xsd:enumeration value="Account Role"/>
                    <xsd:enumeration value="Account Status"/>
                    <xsd:enumeration value="Account Trend"/>
                    <xsd:enumeration value="Address Active Status"/>
                    <xsd:enumeration value="Address Id"/>
                    <xsd:enumeration value="Address Integration Id"/>
                    <xsd:enumeration value="Agreement End Date"/>
                    <xsd:enumeration value="Agreement Name"/>
                    <xsd:enumeration value="Agreement Start Date"/>
                    <xsd:enumeration value="Agreement Status"/>
                    <xsd:enumeration value="Algorithm Type"/>
                    <xsd:enumeration value="Alias"/>
                    <xsd:enumeration value="Annual Revenue"/>
                    <xsd:enumeration value="Assignment Area Code"/>
                    <xsd:enumeration value="Assignment Country Code"/>
                    <xsd:enumeration value="Assignment Excluded"/>
                    <xsd:enumeration value="Assignment Manual Flag"/>
                    <xsd:enumeration value="Back Office Distribution Channel"/>
                    <xsd:enumeration value="Back Office Order Query End  Dt"/>
                    <xsd:enumeration value="Back Office Order Query Start  Dt"/>
                  <xsd:enumeration value="Back Office Sales Area Division Code"/>
                    <xsd:enumeration value="Back Office Sales Organization"/>
                    <xsd:enumeration value="Bill Address Flag"/>
                    <xsd:enumeration value="Bill To City"/>
                    <xsd:enumeration value="Bill To Country"/>
                    <xsd:enumeration value="Bill To First Name"/>
                    <xsd:enumeration value="Bill To Id"/>
                    <xsd:enumeration value="Bill To Job Title"/>
                    <xsd:enumeration value="Bill To Last Name"/>
                    <xsd:enumeration value="Bill To Postal Code"/>
                    <xsd:enumeration value="Bill To State"/>
                    <xsd:enumeration value="Bill To Street Address"/>
                    <xsd:enumeration value="Block Credit Flag"/>
                    <xsd:enumeration value="Business Profile"/>
```

```
<xsd:enumeration value="CSN"/>
<xsd:enumeration value="City"/>
<xsd:enumeration value="City State"/>
<xsd:enumeration value="Competitor"/>
<xsd:enumeration value="Country"/>
<xsd:enumeration value="County"/>
<xsd:enumeration value="Credit Control Area Code"/>
<xsd:enumeration value="Credit Currency Code"/>
<xsd:enumeration value="Credit Limit Amount"/>
<xsd:enumeration value="Credit Profile Id"/>
<xsd:enumeration value="Culture"/>
<xsd:enumeration value="Currency Code"/>
<xsd:enumeration value="Current Volume"/>
<xsd:enumeration value="Current Volume Currency Code"/>
<xsd:enumeration value="Current Volume Exchange Date"/>
<xsd:enumeration value="Customer Account Group"/>
<xsd:enumeration value="DNBReport"/>
<xsd:enumeration value="DUNS Intcode"/>
<xsd:enumeration value="DUNS Number"/>
<xsd:enumeration value="Date Formed"/>
<xsd:enumeration value="DeDup Key Modification Date"/>
<xsd:enumeration value="DeDup Key Update"/>
<xsd:enumeration value="DeDup Keys"/>
<xsd:enumeration value="DeDup Last Match Date"/>
<xsd:enumeration value="DeDup Token"/>
<xsd:enumeration value="Deduplication Match Score"/>
<xsd:enumeration value="Deduplication Object Id"/>
<xsd:enumeration value="Description"/>
<xsd:enumeration value="Disable DataCleansing"/>
<xsd:enumeration value="Division"/>
<xsd:enumeration value="Domestic Ultimate DUNS"/>
<xsd:enumeration value="Dummy"/>
<xsd:enumeration value="EAI Sync Date"/>
<xsd:enumeration value="EAI Sync Error Text"/>
<xsd:enumeration value="EAI Sync Status Code"/>
<xsd:enumeration value="Email Address"/>
<xsd:enumeration value="Employee Here"/>
<xsd:enumeration value="Employees"/>
<xsd:enumeration value="Expertise"/>
<xsd:enumeration value="Explorer Label"/>
<xsd:enumeration value="Fax Number"/>
<xsd:enumeration value="Fiscal Year End"/>
<xsd:enumeration value="Freight Terms"/>
<xsd:enumeration value="Freight Terms Info"/>
<xsd:enumeration value="Full Address"/>
<xsd:enumeration value="GSA Flag"/>
<xsd:enumeration value="Global Ultimate DUNS"/>
<xsd:enumeration value="Goals"/>
<xsd:enumeration value="Group Type Code"/>
```

```
<xsd:enumeration value="Home Page"/>
<xsd:enumeration value="Industry Condition"/>
<xsd:enumeration value="Industry Trend"/>
<xsd:enumeration value="Integration Id"/>
<xsd:enumeration value="Internal Org Flag"/>
<xsd:enumeration value="Joined Synonym"/>
<xsd:enumeration value="Key Competitors"/>
<xsd:enumeration value="Language Code"/>
<xsd:enumeration value="Last Clnse Date"/>
<xsd:enumeration value="Last Manager Review Date"/>
<xsd:enumeration value="Last Review Manager Id"/>
<xsd:enumeration value="Line of Business"/>
<xsd:enumeration value="Location"/>
<xsd:enumeration value="Location Level"/>
<xsd:enumeration value="Main Address Flag"/>
<xsd:enumeration value="Main Fax Number"/>
<xsd:enumeration value="Main Phone Number"/>
<xsd:enumeration value="Managers Review"/>
<xsd:enumeration value="Marketing"/>
<xsd:enumeration value="Merge Sequence Number"/>
<xsd:enumeration value="Mission"/>
<xsd:enumeration value="Name"/>
<xsd:enumeration value="Name and Location"/>
<xsd:enumeration value="Not Manager Flag"/>
<xsd:enumeration value="Notes"/>
<xsd:enumeration value="Objectives"/>
<xsd:enumeration value="Organization Id"/>
<xsd:enumeration value="Organization Integration Id"/>
<xsd:enumeration value="Our Position"/>
<xsd:enumeration value="Outline Number"/>
<xsd:enumeration value="PO Approved Flag"/>
<xsd:enumeration value="PO Auto Approval Currency Code"/>
<xsd:enumeration value="PO Auto Approval Date"/>
<xsd:enumeration value="PO Auto Approval Limit"/>
<xsd:enumeration value="Parent Account Division"/>
<xsd:enumeration value="Parent Account Id"/>
<xsd:enumeration value="Parent Account Integration Id"/>
<xsd:enumeration value="Parent Account Location"/>
<xsd:enumeration value="Parent Account Location Level"/>
<xsd:enumeration value="Parent Account Name"/>
<xsd:enumeration value="Parent Account Region"/>
<xsd:enumeration value="Parent HQ DUNS"/>
<xsd:enumeration value="Partner Flag"/>
<xsd:enumeration value="Partners"/>
<xsd:enumeration value="Party Name"/>
<xsd:enumeration value="Party Type Code"/>
<xsd:enumeration value="Party UId"/>
<xsd:enumeration value="Philosophy"/>
<xsd:enumeration value="Phone Number"/>
```

```
<xsd:enumeration value="Position Integration Id"/>
<xsd:enumeration value="Postal Code"/>
<xsd:enumeration value="Price List"/>
<xsd:enumeration value="Price List End Date"/>
<xsd:enumeration value="Price List Id"/>
<xsd:enumeration value="Price List Integration Id"/>
<xsd:enumeration value="Price List Start Date"/>
<xsd:enumeration value="Primary Account City"/>
<xsd:enumeration value="Primary Account Country"/>
<xsd:enumeration value="Primary Account Postal Code"/>
<xsd:enumeration value="Primary Account State"/>
<xsd:enumeration value="Primary Account Street Address"/>
<xsd:enumeration value="Primary Address Id"/>
<xsd:enumeration value="Primary Assignment Denorm Flag"/>
<xsd:enumeration value="Primary Assignment Manual Flag"/>
<xsd:enumeration value="Primary Assignment System Flag"/>
<xsd:enumeration value="Primary Assignment Type"/>
<xsd:enumeration value="Primary Bill To Address Id"/>
<xsd:enumeration value="Primary Bill To City"/>
<xsd:enumeration value="Primary Bill To Country"/>
<xsd:enumeration value="Primary Bill To First Name"/>
<xsd:enumeration value="Primary Bill To Job Title"/>
<xsd:enumeration value="Primary Bill To Last Name"/>
<xsd:enumeration value="Primary Bill To Person Id"/>
<xsd:enumeration value="Primary Bill To Postal Code"/>
<xsd:enumeration value="Primary Bill To State"/>
<xsd:enumeration value="Primary Bill To Street Address"/>
<xsd:enumeration value="Primary Category Id"/>
<xsd:enumeration value="Primary Fulfill InvLoc Integration Id"/>
<xsd:enumeration value="Primary Fulfillment InvLoc ID"/>
<xsd:enumeration value="Primary Fulfillment Inventory Location"/>
<xsd:enumeration value="Primary Industry Id"/>
<xsd:enumeration value="Primary Organization"/>
<xsd:enumeration value="Primary Organization Id"/>
<xsd:enumeration value="Primary Payer Account"/>
<xsd:enumeration value="Primary Payer Account Id"/>
<xsd:enumeration value="Primary Position Id"/>
<xsd:enumeration value="Primary Service Agreement Id"/>
<xsd:enumeration value="Primary Ship To Address Id"/>
<xsd:enumeration value="Primary Ship To City"/>
<xsd:enumeration value="Primary Ship To Country"/>
<xsd:enumeration value="Primary Ship To First Name"/>
<xsd:enumeration value="Primary Ship To Job Title"/>
<xsd:enumeration value="Primary Ship To Last Name"/>
<xsd:enumeration value="Primary Ship To Person Id"/>
<xsd:enumeration value="Primary Ship To Postal Code"/>
<xsd:enumeration value="Primary Ship To State"/>
<xsd:enumeration value="Primary Ship To Street Address"/>
<xsd:enumeration value="Primary Synonym Id"/>
```

```
<xsd:enumeration value="Primary Territory Id"/>
<xsd:enumeration value="Primary Type Id"/>
<xsd:enumeration value="Profit"/>
<xsd:enumeration value="Project Bill Type"/>
<xsd:enumeration value="Project Comments"/>
<xsd:enumeration value="Project Fix Fee"/>
<xsd:enumeration value="Project Hour Limit"/>
<xsd:enumeration value="Project Id"/>
<xsd:enumeration value="Project Name"/>
<xsd:enumeration value="Project Percentage of Fee"/>
<xsd:enumeration value="Project Purchase Order"/>
<xsd:enumeration value="Project Relationship Type"/>
<xsd:enumeration value="Project Role"/>
<xsd:enumeration value="Prospect Flag"/>
<xsd:enumeration value="Province"/>
<xsd:enumeration value="Public"/>
<xsd:enumeration value="Reference Date"/>
<xsd:enumeration value="Reference Flag"/>
<xsd:enumeration value="Reference Stage"/>
<xsd:enumeration value="Region"/>
<xsd:enumeration value="Relationship Level"/>
<xsd:enumeration value="Relationship Type"/>
<xsd:enumeration value="Response Time"/>
<xsd:enumeration value="Revenue"/>
<xsd:enumeration value="Revenue Growth"/>
<xsd:enumeration value="Revision Number"/>
<xsd:enumeration value="Row Status"/>
<xsd:enumeration value="Row Status Asterisk"/>
<xsd:enumeration value="S-S Instance"/>
<xsd:enumeration value="S-S Instance Id"/>
<xsd:enumeration value="S-S Key Id"/>
<xsd:enumeration value="Service Calendar"/>
<xsd:enumeration value="Service Type"/>
<xsd:enumeration value="Ship Address Flag"/>
<xsd:enumeration value="Ship To City"/>
<xsd:enumeration value="Ship To Country"/>
<xsd:enumeration value="Ship To First Name"/>
<xsd:enumeration value="Ship To Job Title"/>
<xsd:enumeration value="Ship To Last Name"/>
<xsd:enumeration value="Ship To Postal Code"/>
<xsd:enumeration value="Ship To State"/>
<xsd:enumeration value="Ship To Street Address"/>
<xsd:enumeration value="Start Date"/>
<xsd:enumeration value="State"/>
<xsd:enumeration value="Strategies"/>
<xsd:enumeration value="Strategy"/>
<xsd:enumeration value="Street Address"/>
<xsd:enumeration value="Street Address 2"/>
<xsd:enumeration value="Success Factors"/>
```

```
                    <xsd:enumeration value="Synonym"/>
                    <xsd:enumeration value="Territory"/>
                    <xsd:enumeration value="Territory Id"/>
                    <xsd:enumeration value="Timestamp"/>
                    <xsd:enumeration value="Today"/>
                    <xsd:enumeration value="Total Potential Volume"/>
                  <xsd:enumeration value="Total Potential Volume Currency Code"/>
                  <xsd:enumeration value="Total Potential Volume Exchange Date"/>
                    <xsd:enumeration value="Type"/>
                    <xsd:enumeration value="Type MVF"/>
                    <xsd:enumeration value="VAT registration number"/>
                    <xsd:enumeration value="Value Proposition"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:element>
      </xsd:sequence>
      <xsd:attribute name="boname" type="xsd:string" use="required"
fixed="Account"/>
      <xsd:attribute name="bcname" type="xsd:string" use="required"
fixed="Account"/>
     <xsd:attribute name="operation" use="required">
     <xsd:simpleType>
     <xsd:restriction base="xsd:string">
     <xsd:enumeration value="insert"/>
     <xsd:enumeration value="update"/>
     <xsd:enumeration value="delete"/>
     <xsd:enumeration value="query"/>
 </xsd:restriction>
</xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Account Response Schema

Listing A-2 Account Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
```

```
  <xsd:element name="BusinessComponent">
    <xsd:complexType>
      <xsd:sequence>
<xsd:element name="Record" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence>
<xsd:element name="Field" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Account Competitors"/>
                    <xsd:enumeration value="Account Condition"/>
                    <xsd:enumeration value="Account Markets"/>
                   <xsd:enumeration value="Account Organization Integration Id"/>
                    <xsd:enumeration value="Account Products"/>
                    <xsd:enumeration value="Account Role"/>
                    <xsd:enumeration value="Account Status"/>
                    <xsd:enumeration value="Account Trend"/>
                    <xsd:enumeration value="Address Active Status"/>
                    <xsd:enumeration value="Address Id"/>
                    <xsd:enumeration value="Address Integration Id"/>
                    <xsd:enumeration value="Agreement End Date"/>
                    <xsd:enumeration value="Agreement Name"/>
                    <xsd:enumeration value="Agreement Start Date"/>
                    <xsd:enumeration value="Agreement Status"/>
                    <xsd:enumeration value="Algorithm Type"/>
                    <xsd:enumeration value="Alias"/>
                    <xsd:enumeration value="Annual Revenue"/>
                    <xsd:enumeration value="Assignment Area Code"/>
                    <xsd:enumeration value="Assignment Country Code"/>
                    <xsd:enumeration value="Assignment Excluded"/>
                    <xsd:enumeration value="Assignment Manual Flag"/>
                    <xsd:enumeration value="Back Office Distribution Channel"/>
                    <xsd:enumeration value="Back Office Order Query End  Dt"/>
                    <xsd:enumeration value="Back Office Order Query Start  Dt"/>
                  <xsd:enumeration value="Back Office Sales Area Division Code"/>
                    <xsd:enumeration value="Back Office Sales Organization"/>
                    <xsd:enumeration value="Bill Address Flag"/>
                    <xsd:enumeration value="Bill To City"/>
                    <xsd:enumeration value="Bill To Country"/>
                    <xsd:enumeration value="Bill To First Name"/>
                    <xsd:enumeration value="Bill To Id"/>
                    <xsd:enumeration value="Bill To Job Title"/>
                    <xsd:enumeration value="Bill To Last Name"/>
                    <xsd:enumeration value="Bill To Postal Code"/>
                    <xsd:enumeration value="Bill To State"/>
                    <xsd:enumeration value="Bill To Street Address"/>
```

```
<xsd:enumeration value="Block Credit Flag"/>
<xsd:enumeration value="Business Profile"/>
<xsd:enumeration value="CSN"/>
<xsd:enumeration value="City"/>
<xsd:enumeration value="City State"/>
<xsd:enumeration value="Competitor"/>
<xsd:enumeration value="Country"/>
<xsd:enumeration value="County"/>
<xsd:enumeration value="Credit Control Area Code"/>
<xsd:enumeration value="Credit Currency Code"/>
<xsd:enumeration value="Credit Limit Amount"/>
<xsd:enumeration value="Credit Profile Id"/>
<xsd:enumeration value="Culture"/>
<xsd:enumeration value="Currency Code"/>
<xsd:enumeration value="Current Volume"/>
<xsd:enumeration value="Current Volume Currency Code"/>
<xsd:enumeration value="Current Volume Exchange Date"/>
<xsd:enumeration value="Customer Account Group"/>
<xsd:enumeration value="DNBReport"/>
<xsd:enumeration value="DUNS Intcode"/>
<xsd:enumeration value="DUNS Number"/>
<xsd:enumeration value="Date Formed"/>
<xsd:enumeration value="DeDup Key Modification Date"/>
<xsd:enumeration value="DeDup Key Update"/>
<xsd:enumeration value="DeDup Keys"/>
<xsd:enumeration value="DeDup Last Match Date"/>
<xsd:enumeration value="DeDup Token"/>
<xsd:enumeration value="Deduplication Match Score"/>
<xsd:enumeration value="Deduplication Object Id"/>
<xsd:enumeration value="Description"/>
<xsd:enumeration value="Disable DataCleansing"/>
<xsd:enumeration value="Division"/>
<xsd:enumeration value="Domestic Ultimate DUNS"/>
<xsd:enumeration value="Dummy"/>
<xsd:enumeration value="EAI Sync Date"/>
<xsd:enumeration value="EAI Sync Error Text"/>
<xsd:enumeration value="EAI Sync Status Code"/>
<xsd:enumeration value="Email Address"/>
<xsd:enumeration value="Employee Here"/>
<xsd:enumeration value="Employees"/>
<xsd:enumeration value="Expertise"/>
<xsd:enumeration value="Explorer Label"/>
<xsd:enumeration value="Fax Number"/>
<xsd:enumeration value="Fiscal Year End"/>
<xsd:enumeration value="Freight Terms"/>
<xsd:enumeration value="Freight Terms Info"/>
<xsd:enumeration value="Full Address"/>
<xsd:enumeration value="GSA Flag"/>
<xsd:enumeration value="Global Ultimate DUNS"/>
```

```
<xsd:enumeration value="Goals"/>
<xsd:enumeration value="Group Type Code"/>
<xsd:enumeration value="Home Page"/>
<xsd:enumeration value="Industry Condition"/>
<xsd:enumeration value="Industry Trend"/>
<xsd:enumeration value="Integration Id"/>
<xsd:enumeration value="Internal Org Flag"/>
<xsd:enumeration value="Joined Synonym"/>
<xsd:enumeration value="Key Competitors"/>
<xsd:enumeration value="Language Code"/>
<xsd:enumeration value="Last Clnse Date"/>
<xsd:enumeration value="Last Manager Review Date"/>
<xsd:enumeration value="Last Review Manager Id"/>
<xsd:enumeration value="Line of Business"/>
<xsd:enumeration value="Location"/>
<xsd:enumeration value="Location Level"/>
<xsd:enumeration value="Main Address Flag"/>
<xsd:enumeration value="Main Fax Number"/>
<xsd:enumeration value="Main Phone Number"/>
<xsd:enumeration value="Managers Review"/>
<xsd:enumeration value="Marketing"/>
<xsd:enumeration value="Merge Sequence Number"/>
<xsd:enumeration value="Mission"/>
<xsd:enumeration value="Name"/>
<xsd:enumeration value="Name and Location"/>
<xsd:enumeration value="Not Manager Flag"/>
<xsd:enumeration value="Notes"/>
<xsd:enumeration value="Objectives"/>
<xsd:enumeration value="Organization Id"/>
<xsd:enumeration value="Organization Integration Id"/>
<xsd:enumeration value="Our Position"/>
<xsd:enumeration value="Outline Number"/>
<xsd:enumeration value="PO Approved Flag"/>
<xsd:enumeration value="PO Auto Approval Currency Code"/>
<xsd:enumeration value="PO Auto Approval Date"/>
<xsd:enumeration value="PO Auto Approval Limit"/>
<xsd:enumeration value="Parent Account Division"/>
<xsd:enumeration value="Parent Account Id"/>
<xsd:enumeration value="Parent Account Integration Id"/>
<xsd:enumeration value="Parent Account Location"/>
<xsd:enumeration value="Parent Account Location Level"/>
<xsd:enumeration value="Parent Account Name"/>
<xsd:enumeration value="Parent Account Region"/>
<xsd:enumeration value="Parent HQ DUNS"/>
<xsd:enumeration value="Partner Flag"/>
<xsd:enumeration value="Partners"/>
<xsd:enumeration value="Party Name"/>
<xsd:enumeration value="Party Type Code"/>
<xsd:enumeration value="Party UId"/>
```

```
<xsd:enumeration value="Philosophy"/>
<xsd:enumeration value="Phone Number"/>
<xsd:enumeration value="Position Integration Id"/>
<xsd:enumeration value="Postal Code"/>
<xsd:enumeration value="Price List"/>
<xsd:enumeration value="Price List End Date"/>
<xsd:enumeration value="Price List Id"/>
<xsd:enumeration value="Price List Integration Id"/>
<xsd:enumeration value="Price List Start Date"/>
<xsd:enumeration value="Primary Account City"/>
<xsd:enumeration value="Primary Account Country"/>
<xsd:enumeration value="Primary Account Postal Code"/>
<xsd:enumeration value="Primary Account State"/>
<xsd:enumeration value="Primary Account Street Address"/>
<xsd:enumeration value="Primary Address Id"/>
<xsd:enumeration value="Primary Assignment Denorm Flag"/>
<xsd:enumeration value="Primary Assignment Manual Flag"/>
<xsd:enumeration value="Primary Assignment System Flag"/>
<xsd:enumeration value="Primary Assignment Type"/>
<xsd:enumeration value="Primary Bill To Address Id"/>
<xsd:enumeration value="Primary Bill To City"/>
<xsd:enumeration value="Primary Bill To Country"/>
<xsd:enumeration value="Primary Bill To First Name"/>
<xsd:enumeration value="Primary Bill To Job Title"/>
<xsd:enumeration value="Primary Bill To Last Name"/>
<xsd:enumeration value="Primary Bill To Person Id"/>
<xsd:enumeration value="Primary Bill To Postal Code"/>
<xsd:enumeration value="Primary Bill To State"/>
<xsd:enumeration value="Primary Bill To Street Address"/>
<xsd:enumeration value="Primary Category Id"/>
<xsd:enumeration value="Primary Fulfill InvLoc Integration Id"/>
<xsd:enumeration value="Primary Fulfillment InvLoc ID"/>
<xsd:enumeration value="Primary Fulfillment Inventory Location"/>
<xsd:enumeration value="Primary Industry Id"/>
<xsd:enumeration value="Primary Organization"/>
<xsd:enumeration value="Primary Organization Id"/>
<xsd:enumeration value="Primary Payer Account"/>
<xsd:enumeration value="Primary Payer Account Id"/>
<xsd:enumeration value="Primary Position Id"/>
<xsd:enumeration value="Primary Service Agreement Id"/>
<xsd:enumeration value="Primary Ship To Address Id"/>
<xsd:enumeration value="Primary Ship To City"/>
<xsd:enumeration value="Primary Ship To Country"/>
<xsd:enumeration value="Primary Ship To First Name"/>
<xsd:enumeration value="Primary Ship To Job Title"/>
<xsd:enumeration value="Primary Ship To Last Name"/>
<xsd:enumeration value="Primary Ship To Person Id"/>
<xsd:enumeration value="Primary Ship To Postal Code"/>
<xsd:enumeration value="Primary Ship To State"/>
```

```
<xsd:enumeration value="Primary Ship To Street Address"/>
<xsd:enumeration value="Primary Synonym Id"/>
<xsd:enumeration value="Primary Territory Id"/>
<xsd:enumeration value="Primary Type Id"/>
<xsd:enumeration value="Profit"/>
<xsd:enumeration value="Project Bill Type"/>
<xsd:enumeration value="Project Comments"/>
<xsd:enumeration value="Project Fix Fee"/>
<xsd:enumeration value="Project Hour Limit"/>
<xsd:enumeration value="Project Id"/>
<xsd:enumeration value="Project Name"/>
<xsd:enumeration value="Project Percentage of Fee"/>
<xsd:enumeration value="Project Purchase Order"/>
<xsd:enumeration value="Project Relationship Type"/>
<xsd:enumeration value="Project Role"/>
<xsd:enumeration value="Prospect Flag"/>
<xsd:enumeration value="Province"/>
<xsd:enumeration value="Public"/>
<xsd:enumeration value="Reference Date"/>
<xsd:enumeration value="Reference Flag"/>
<xsd:enumeration value="Reference Stage"/>
<xsd:enumeration value="Region"/>
<xsd:enumeration value="Relationship Level"/>
<xsd:enumeration value="Relationship Type"/>
<xsd:enumeration value="Response Time"/>
<xsd:enumeration value="Revenue"/>
<xsd:enumeration value="Revenue Growth"/>
<xsd:enumeration value="Revision Number"/>
<xsd:enumeration value="Row Status"/>
<xsd:enumeration value="Row Status Asterisk"/>
<xsd:enumeration value="S-S Instance"/>
<xsd:enumeration value="S-S Instance Id"/>
<xsd:enumeration value="S-S Key Id"/>
<xsd:enumeration value="Service Calendar"/>
<xsd:enumeration value="Service Type"/>
<xsd:enumeration value="Ship Address Flag"/>
<xsd:enumeration value="Ship To City"/>
<xsd:enumeration value="Ship To Country"/>
<xsd:enumeration value="Ship To First Name"/>
<xsd:enumeration value="Ship To Job Title"/>
<xsd:enumeration value="Ship To Last Name"/>
<xsd:enumeration value="Ship To Postal Code"/>
<xsd:enumeration value="Ship To State"/>
<xsd:enumeration value="Ship To Street Address"/>
<xsd:enumeration value="Start Date"/>
<xsd:enumeration value="State"/>
<xsd:enumeration value="Strategies"/>
<xsd:enumeration value="Strategy"/>
<xsd:enumeration value="Street Address"/>
```

```
                    <xsd:enumeration value="Street Address 2"/>
                    <xsd:enumeration value="Success Factors"/>
                    <xsd:enumeration value="Synonym"/>
                    <xsd:enumeration value="Territory"/>
                    <xsd:enumeration value="Territory Id"/>
                    <xsd:enumeration value="Timestamp"/>
                    <xsd:enumeration value="Today"/>
                    <xsd:enumeration value="Total Potential Volume"/>
                  <xsd:enumeration value="Total Potential Volume Currency Code"/>
                  <xsd:enumeration value="Total Potential Volume Exchange Date"/>
                    <xsd:enumeration value="Type"/>
                    <xsd:enumeration value="Type MVF"/>
                    <xsd:enumeration value="VAT registration number"/>
                    <xsd:enumeration value="Value Proposition"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
      </xsd:sequence>
      <xsd:attribute name="status" use="required">
      <xsd:simpleType>
      <xsd:restriction base="xsd:string">
      <xsd:enumeration value="success"/>
      <xsd:enumeration value="failure"/>
      <xsd:restriction>
      </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="reason" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Sample XML for Account Add Request

Listing A-3 Sample XML for Account Add Request

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

```
 edited with XML Spy v4.4 U (http://www.xmlspy.com) by jerry
(Information Builders Inc.)
-->
- <!--
Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)
-->
 <BusinessComponent
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="E:\edaxml-home\SiebelAdapter\test\
schema4\Account.xsd" boname="Account" bcname="Account"
operation="insert">
<Field name="Currency Code" value="USD" />
<Field name="Name" value="JDB 5" />
</BusinessComponent>
```

Sample XML for Account Add Response

Listing A-4 Sample XML for Account Add Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<BusinessComponent status="success" reason="" />
```

Sample XML for Account Delete Request

Listing A-5 Sample XML for Account Delete Request

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XML Spy v4.4 U (http://www.xmlspy.com)
-->
- <!--
Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)
-->
 <BusinessComponent boname="Account" bcname="Account" operation="delete">
<Select name="Name" value="JD*" />
</BusinessComponent>
```

Sample XML for Account Delete Response

Listing A-6 Sample XML for Account Delete Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<BusinessComponent status="success" reason="" />
```

Sample XML for Account Query Request

Listing A-7 Sample XML for Account Query Request

```
 <BusinessComponent boname="Account" bcname="Account"
operation="query">
<Select name="Name" value="Ja*" />
<Field name="Name" value="" />
<Field name="City" value="" />
<Field name="Street Address" value="" />
<Field name="Type" value="" />
<Field name="Account Status" value="Active" />
</BusinessComponent>
```

Sample XML for Account Query Response

Listing A-8 Sample XML for Account Query Response

```
<?xml version="1.0" encoding="UTF-8" ?>
 <BusinessComponent status="success" reason="">
 <Record>
<Field name="Name" value="Java Data Bean Account3" />
<Field name="City" value="Norwood City3" />
<Field name="Street Address" value="201 Wickham Way3" />
<Field name="Type" value="" />
<Field name="Account Status" value="" />
</Record>
 <Record>
<Field name="Name" value="Java Data Bean Account4" />
```

```
<Field name="City" value="Norwood City4" />
<Field name="Street Address" value="201 Wickham Way4" />
<Field name="Type" value="" />
<Field name="Account Status" value="" />
</Record>
 <Record>
<Field name="Name" value="Java Data Bean Account5" />
<Field name="City" value="Norwood City5" />
<Field name="Street Address" value="201 Wickham Way5" />
<Field name="Type" value="" />
<Field name="Account Status" value="" />
</Record>
 <Record>
<Field name="Name" value="Java Data Bean Account6" />
<Field name="City" value="Norwood City6" />
<Field name="Street Address" value="201 Wickham Way6" />
<Field name="Type" value="OEM" />
<Field name="Account Status" value="Active" />
</Record>
</BusinessComponent>
```

PGMAVV Account Business Service

PGMAVV Account Add Request Schema

Listing A-9 PGMAVV Account Add Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="BusinessService">
    <xsd:complexType>
      <xsd:sequence>
<xsd:element name="Param" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Param1"/>
```

```
                    <xsd:enumeration value="Param2"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
      <xsd:attribute name="value" type="xsd:string" use="required"
/>
    </xsd:complexType>
</xsd:element>
      </xsd:sequence>
      <xsd:attribute name="servicename" type="xsd:string"
use="required" fixed="PGMAVV Calculator"/>
      <xsd:attribute name="methodname" type="xsd:string"
use="required" fixed="Add"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

PGMAVV Account Add Response Schema

Listing A-10 PGMAVV Account Add Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="BusinessService">
    <xsd:complexType>
      <xsd:sequence>
<xsd:element name="Result" minOccurs="0" maxOccurs="unbounded" >
    <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Value"/>
                    <xsd:enumeration value="Operator"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
      <xsd:attribute name="value" type="xsd:string" use="required"
/>
    </xsd:complexType>
</xsd:element>
      </xsd:sequence>
      <xsd:attribute name="status" use="required">
```

```
      <xsd:simpleType>
      <xsd:restriction base="xsd:string">
      <xsd:enumeration value="success"/>
      <xsd:enumeration value="failure"/>
      </xsd:restriction>
      </xsd:simpleType>
      </xsd:attribute>
     <xsd:attribute name="reason" type="xsd:string" use="required"
/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Sample XML for PGMAVV Account Add Request

Listing A-11 Sample XML for PGMAVV Account Add Request

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)
-->
 <BusinessService servicename="PGMAVV Calculator"
methodname="Add">
<Param name="Param1" value="1" />
<Param name="Param2" value="3" />
</BusinessService>
```

Sample XML for PGMAVV Account Add Response

Listing A-12 Sample XML for PGMAVV Account Add Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<BusinessService status="success" reason="">
<Result name="Value" value="4" />
<Result name="Operator" value="+" />
</BusinessService>
```

CHAPTER

B Creating Siebel Workflows

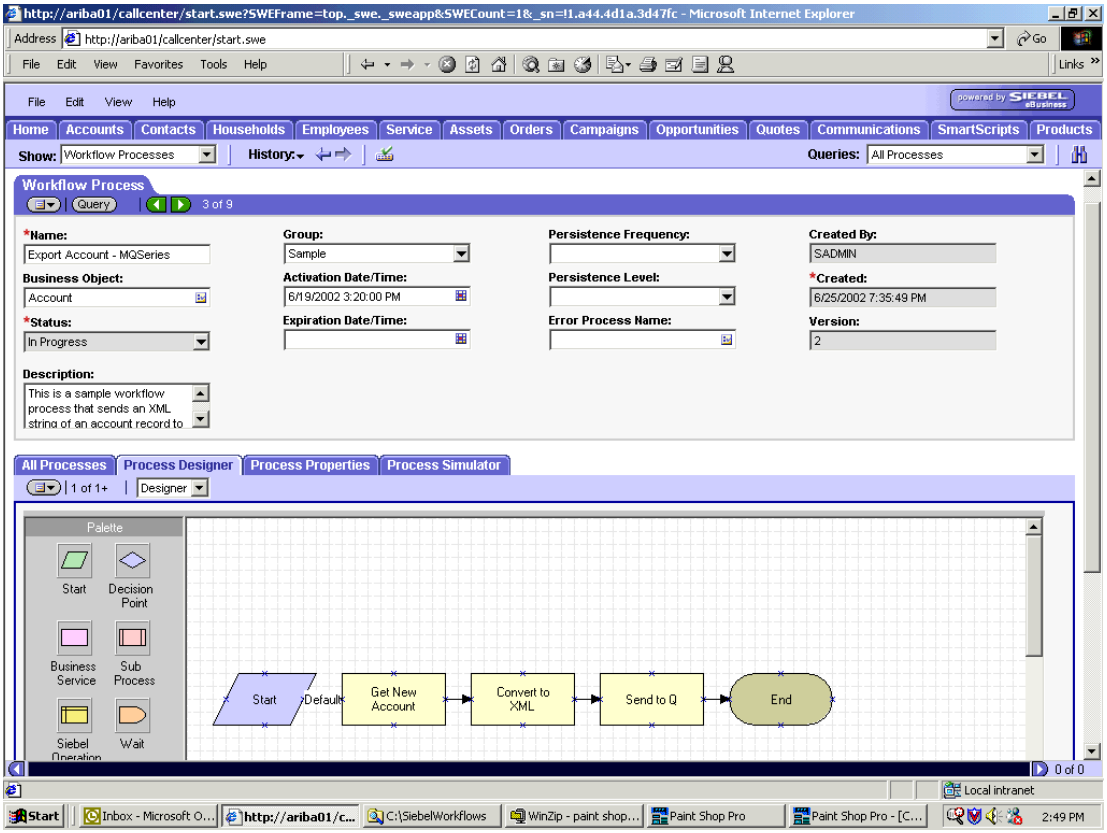
This section provides sample Siebel Workflows. It includes the following topics:

- Creating a Siebel Workflow for Event Using MQSeries Transport
- Creating a Siebel Workflow for Event Using File Transport
- Creating a Siebel Workflow for Event Using HTTP Transport
- Creating a Siebel Workflow for Service Using MQSeries Transport
- Creating a Siebel Workflow for Service Using File Transport
- Creating a Siebel Workflow for Service Using HTTP Transport

Creating a Siebel Workflow for Event Using MQSeries Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for exporting Siebel Account record information using the MQSeries transport.

Figure B-1 Workflow Process Window



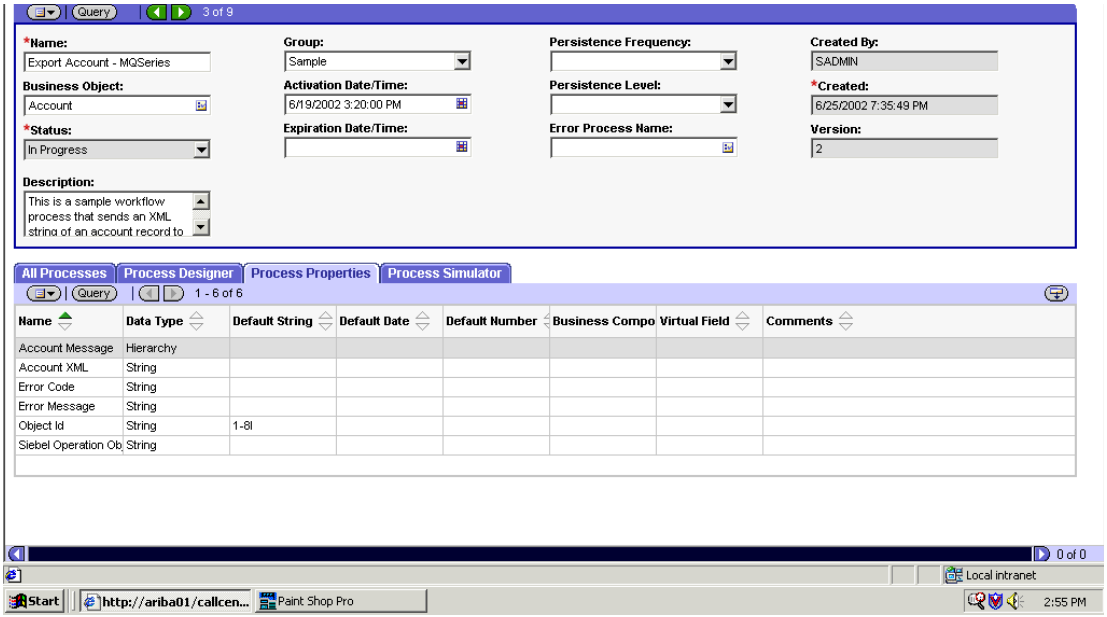
The following is an example of the steps required to create a Siebel Workflow that generates Siebel XML when an Account record is updated in the Siebel Call Center. It is then placed on an IBM MQSeries message queue.

1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   The Account message contains Siebel Account data in hierarchical format.

   Account XML specifies the Siebel Account data that the workflow has converted to XML.

Figure B-2 Workflow Process Window - Process Properties Tab

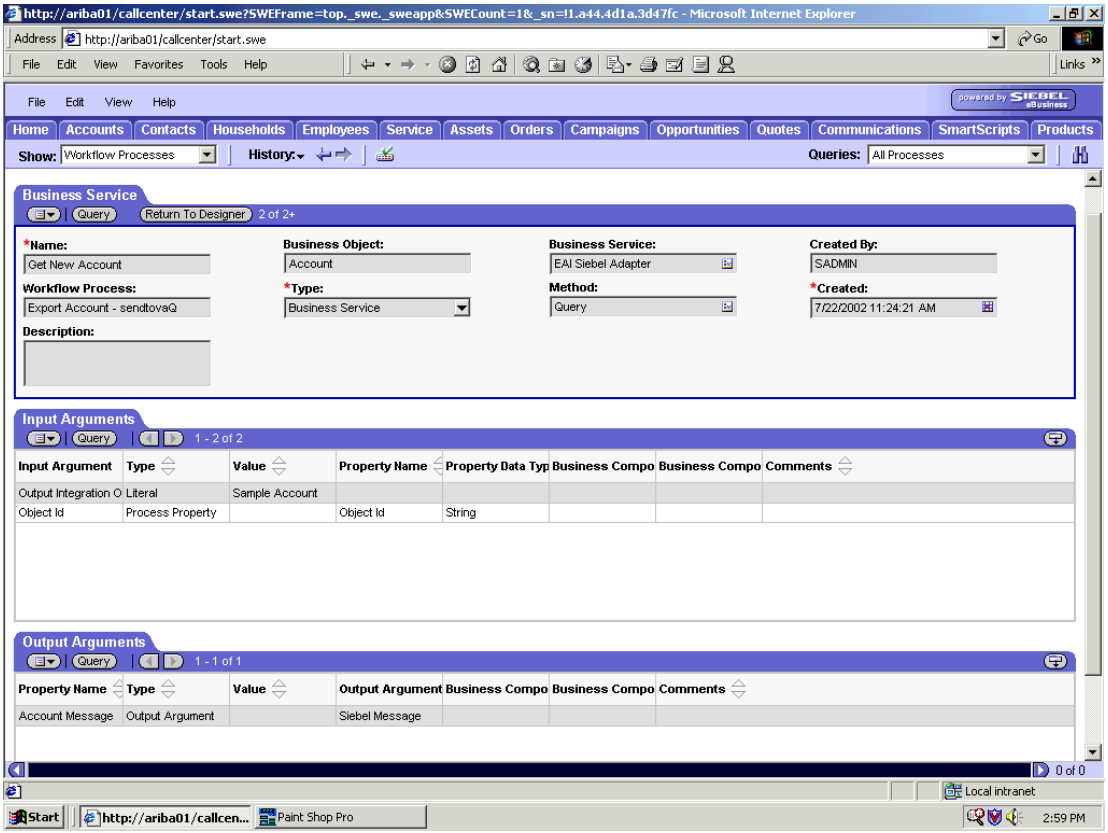


2. Use the Siebel Workflow Administration windows to create a workflow. Define an EAI Siebel Adapter business service step to receive an instance of Account data and call it Get New Account.

   The business service obtains the Account information from Siebel using the Query method.

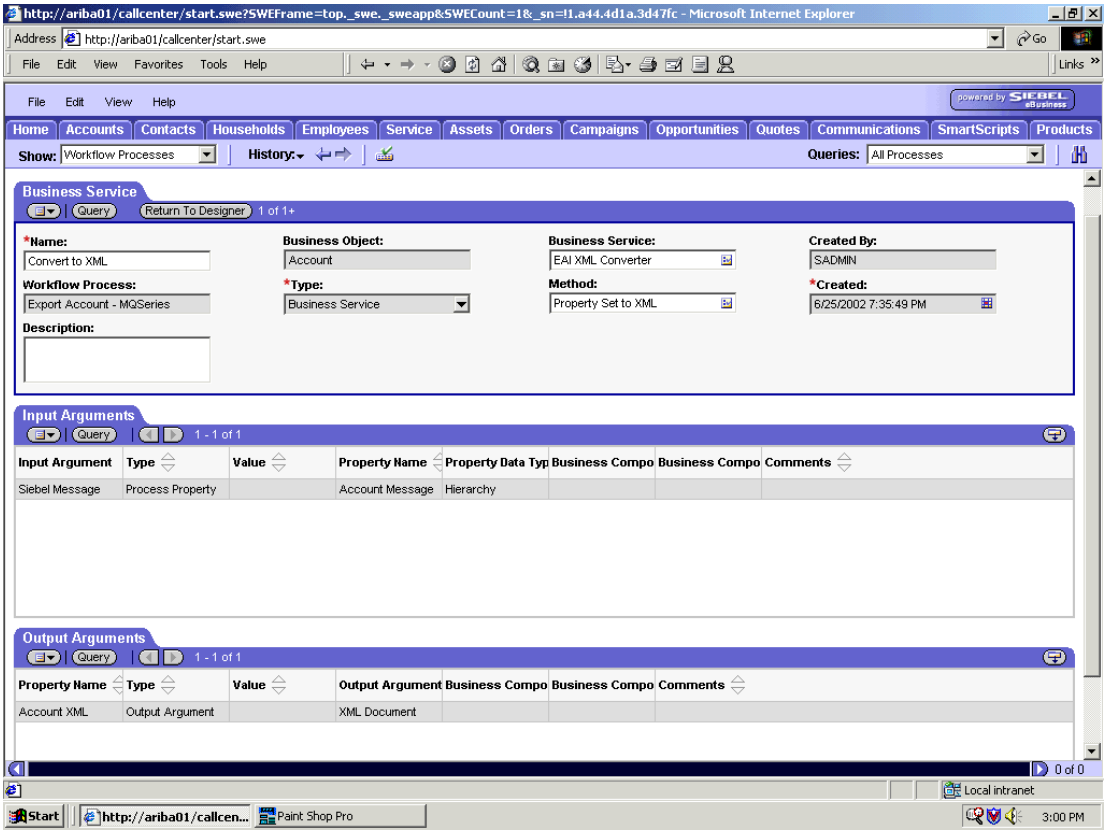
   Output from this business service is generated in hierarchical format.

Figure B-3 Business Service Window



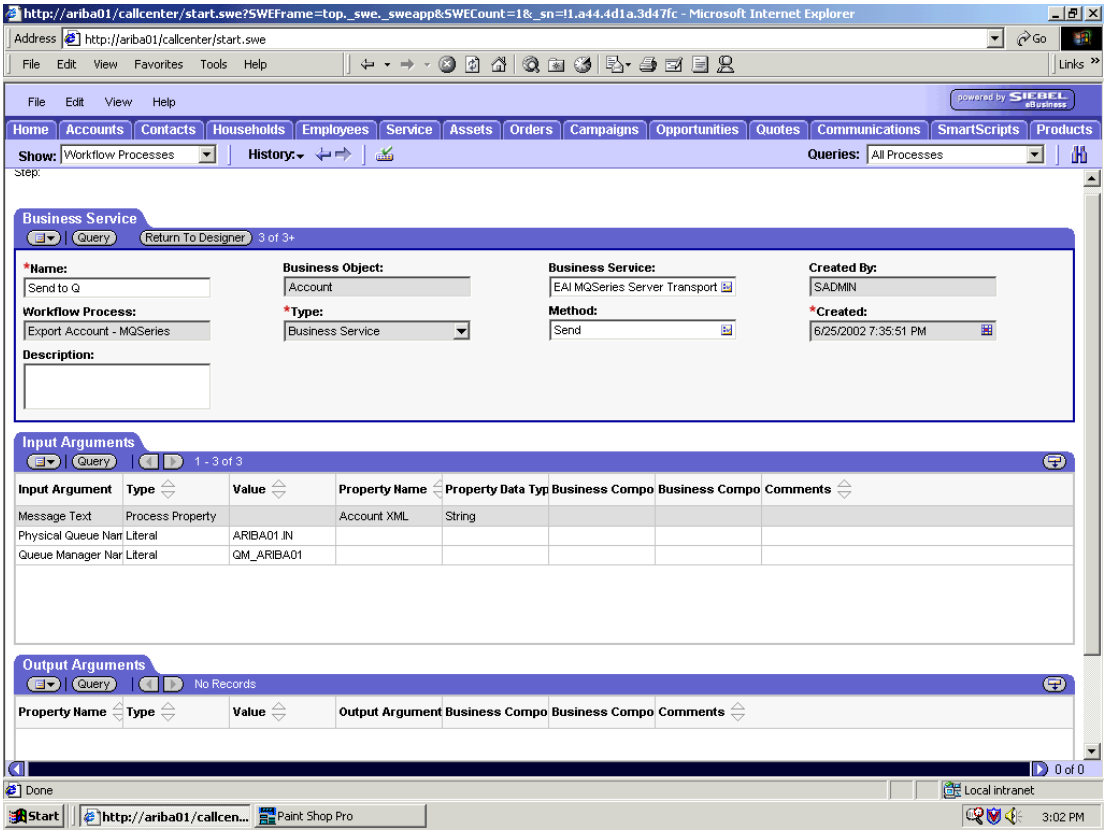
3. Define an EAI XML Converter business service step and call it Convert to XML. It should be defined to receive the Account data from the EAI Siebel Adapter business service in hierarchical format and convert it to XML format.

Figure B-4 Business Service Window - Input and Output Arguments



4. Define an EAI MQSeries server transport business service step and call it Send to Q. It should be defined to receive the Account data from the EAI XML Converter business service in Siebel XML format and send the Account XML to MQSeries using the Send method.

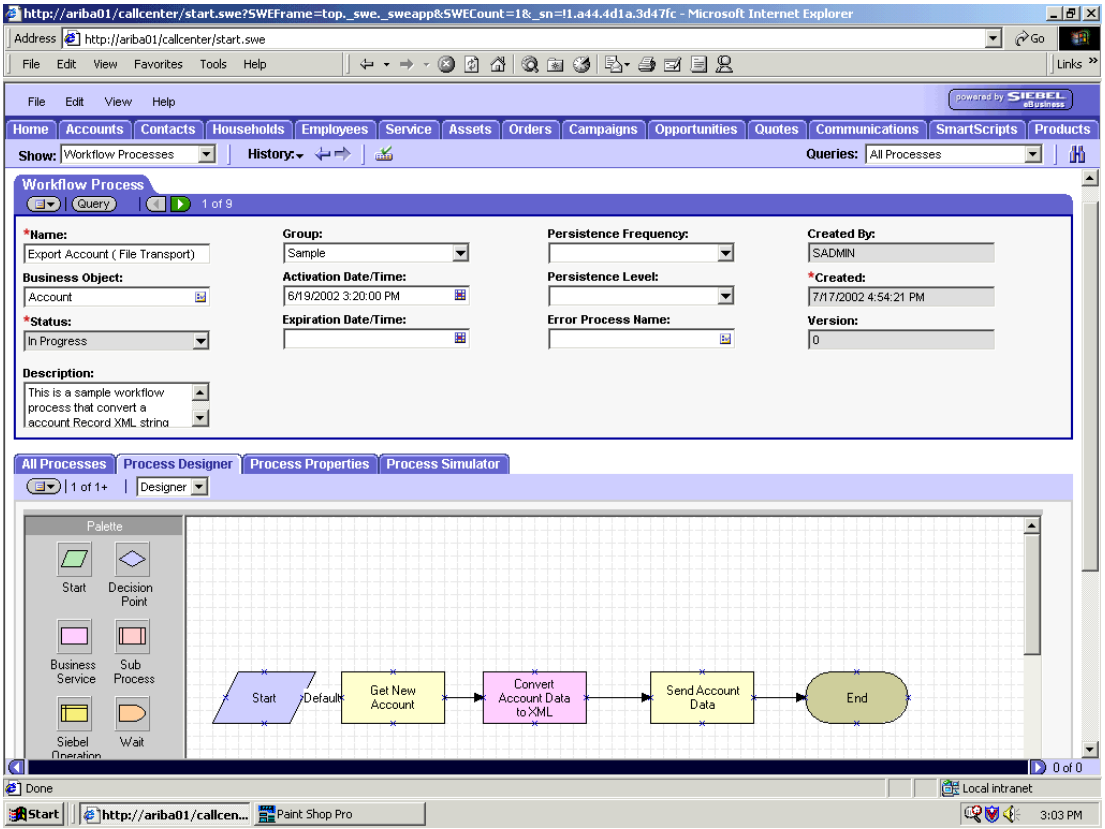
Figure B-5 Window for Input Arguments



Creating a Siebel Workflow for Event Using File Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for exporting Siebel Account record information using the File transport.

Figure B-6 Workflow Process Window



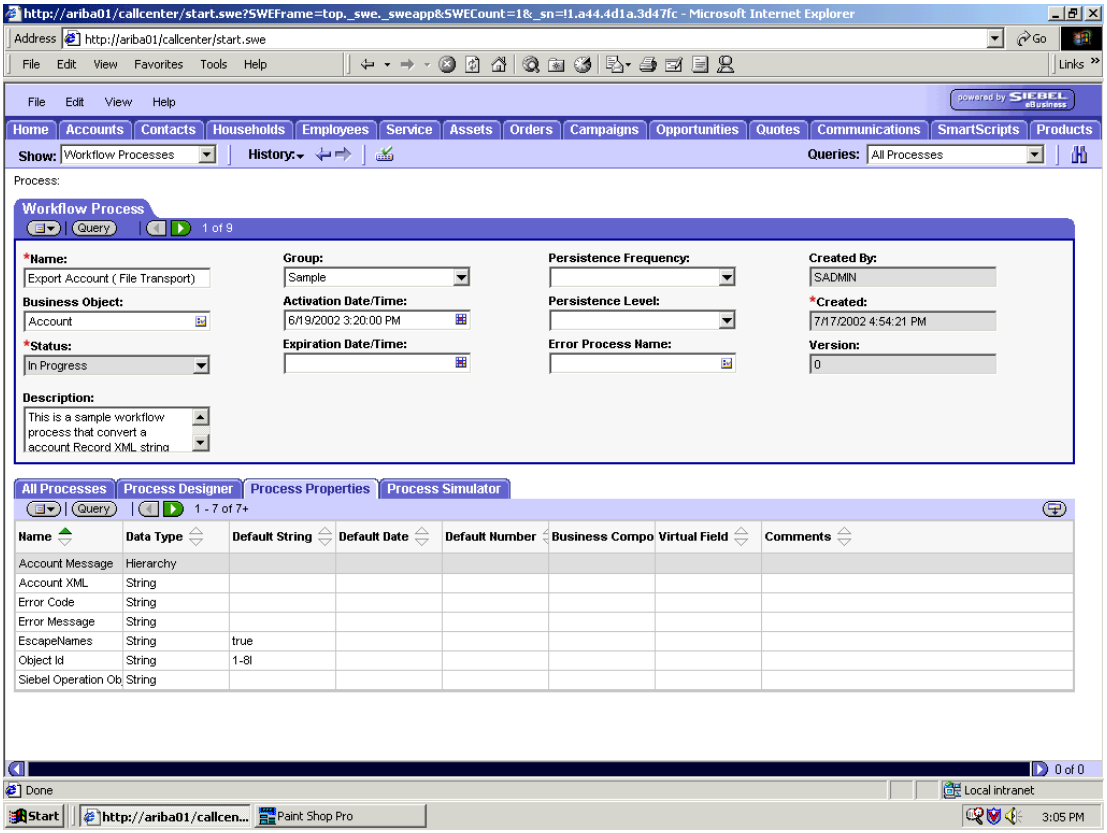
The following is an example of the steps required to create a Siebel Workflow that generates Siebel XML when an Account record is updated in Siebel Call Center and then places it on the file system.

1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   Account message contains the Siebel Account data in hierarchical format.

   Account XML specifies which Siebel Account data the workflow has converted to XML.

Figure B-7 Workflow Process Window - Process Properties Tab

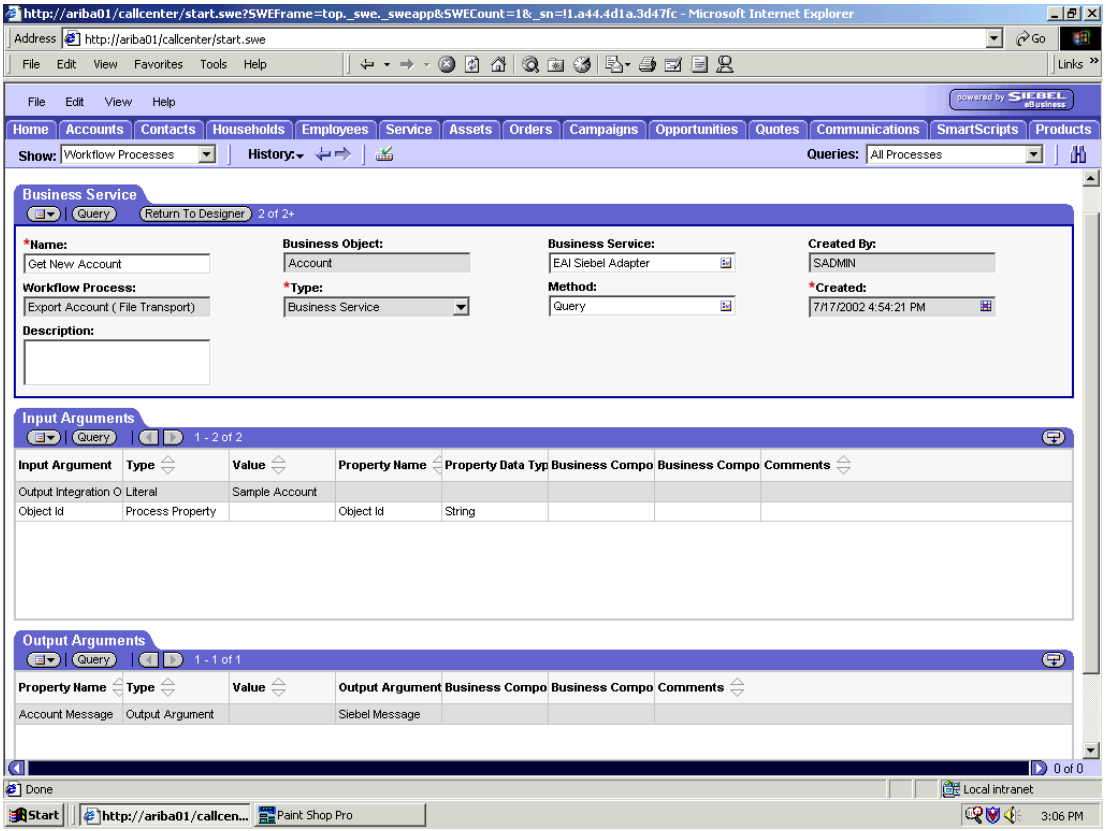


2. Use the Siebel Workflow Administration windows to create a workflow. Define an EAI Siebel Adapter business service step to receive an instance of Account data and call it Get New Account.

   The business service obtains the Account information from Siebel using the Query method.

   Output from this business service is generated in hierarchical format.

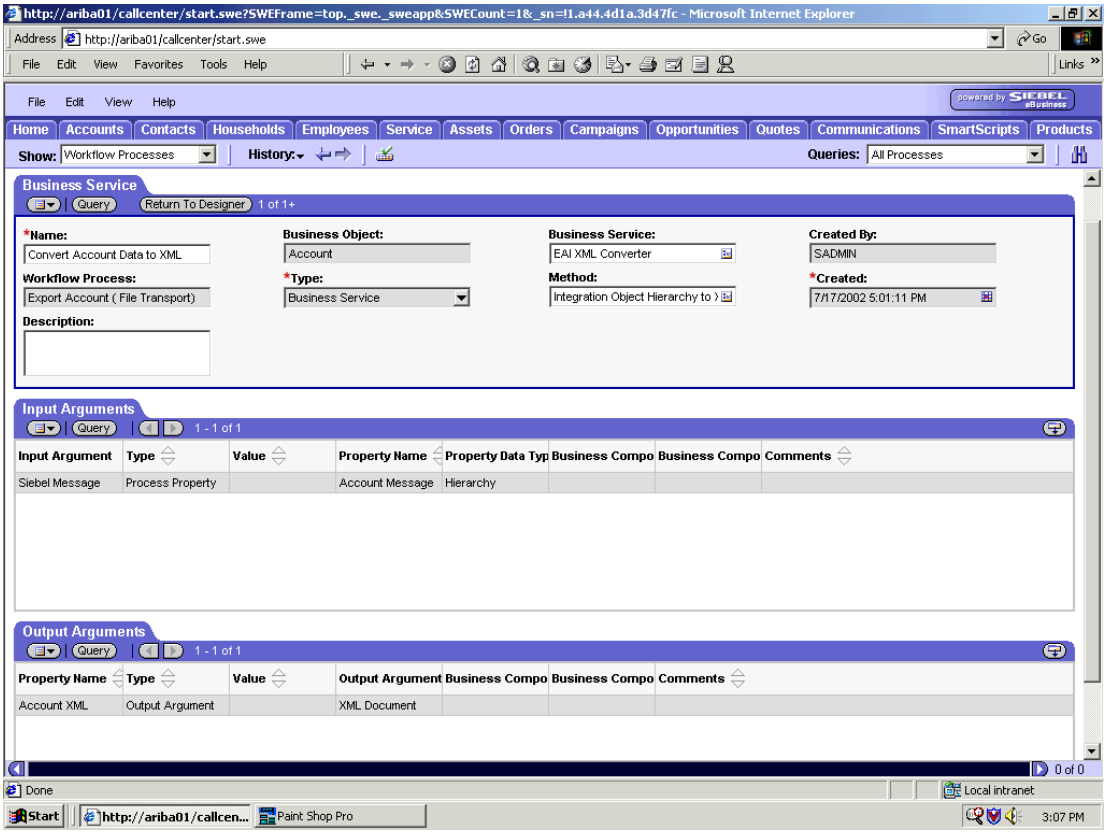
Figure B-8 Business Service Output Window



3. Define an EAI XML Converter business service step and call it Convert Account Data to XML.

   It should be defined to receive the Account data from the EAI Siebel Adapter business service in hierarchical format and convert it to XML format.

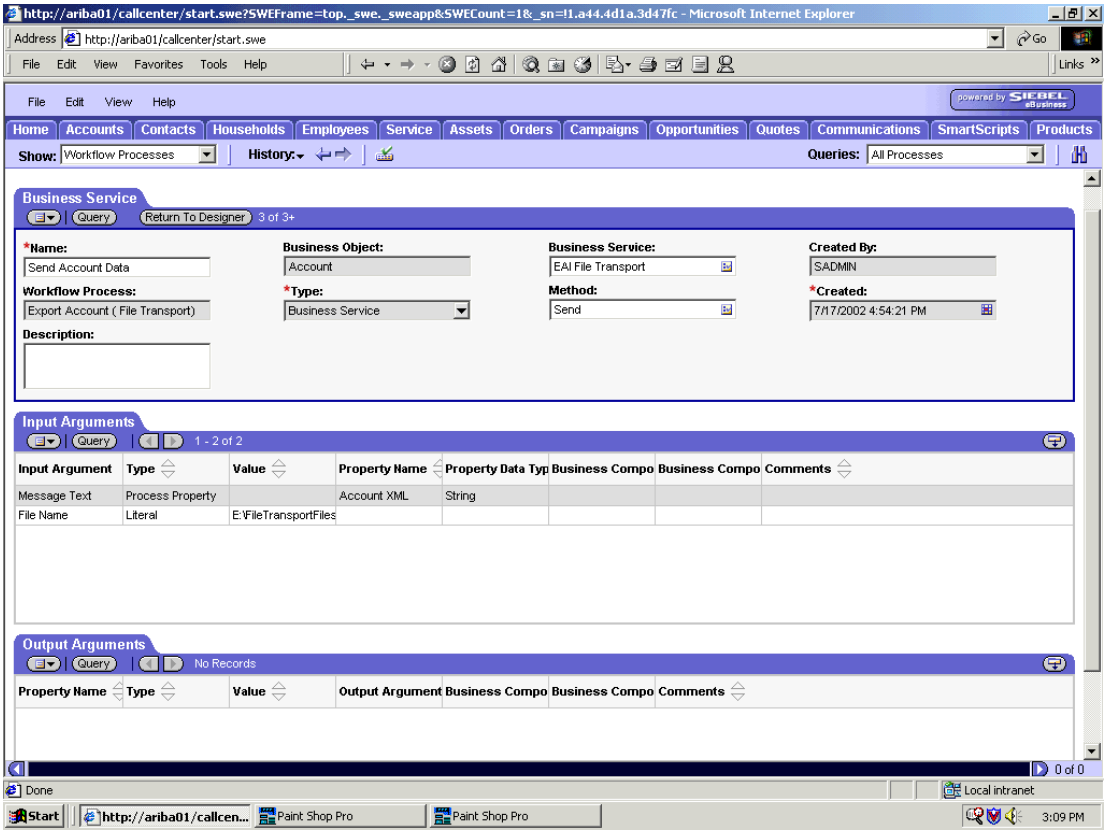
Figure B-9 EAI XML Converter Business Service Window



4. Define an EAI File transport business service step and call it Send Account Data.

   It should be defined to receive the Account data from the EAI XML Converter business service in Siebel XML format and Send the Account XML to the file system in a specified directory using the Send method.

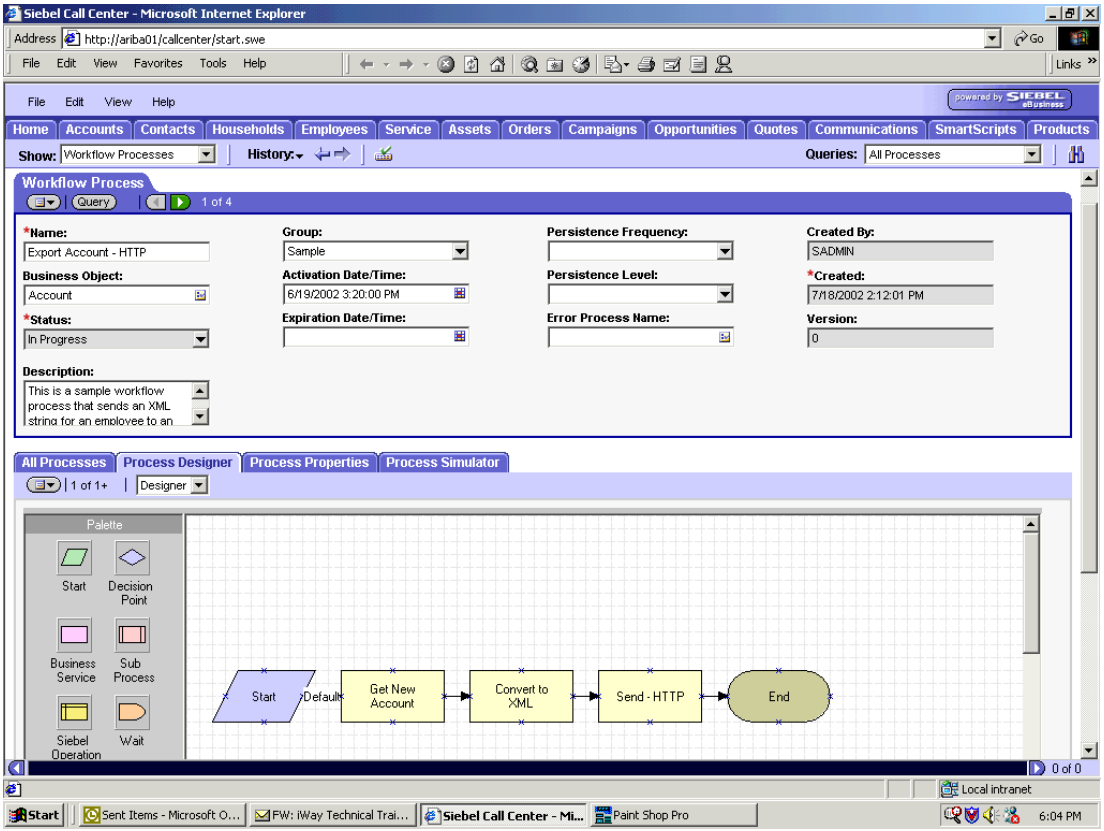
Figure B-10 EAI File Transport Business Service Window



Creating a Siebel Workflow for Event Using HTTP Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for exporting Siebel Account record information using the HTTP transport.

Figure B-11 Workflow Process Window

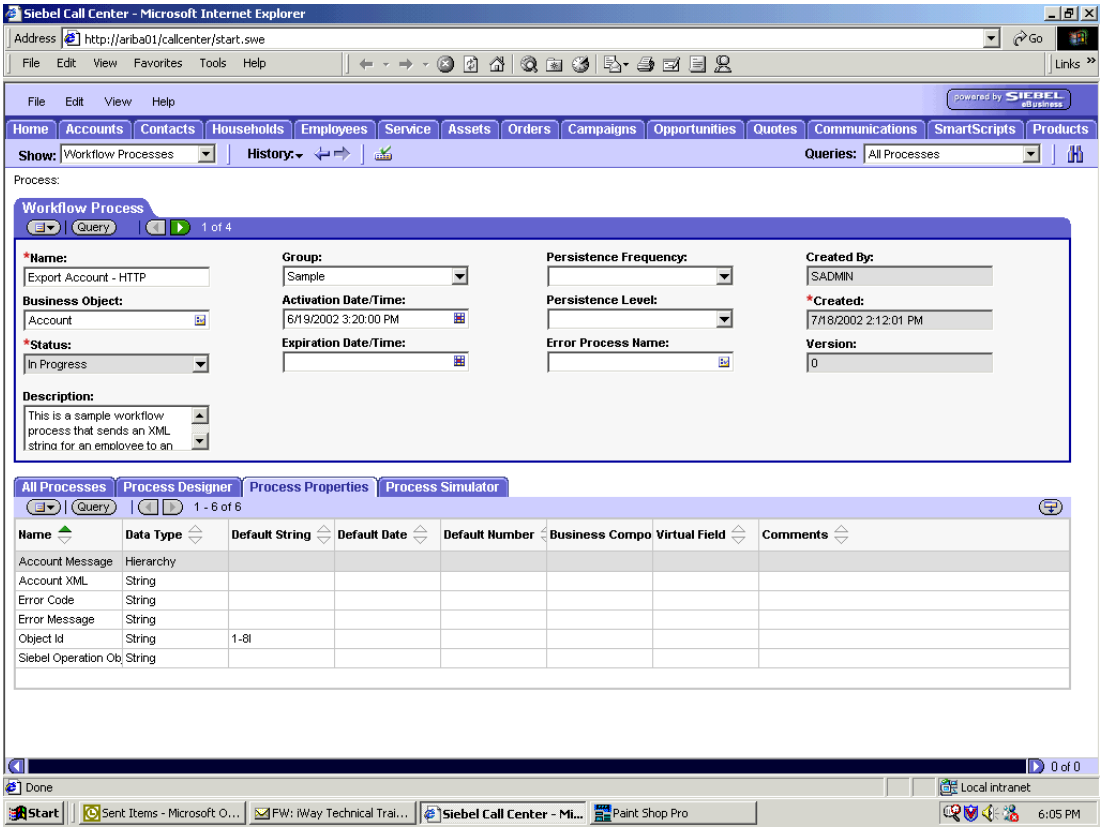


1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   Account message contains the Siebel Account data in hierarchical format.

   Account XML specifies the Siebel Account data that the workflow has converted to XML.

Figure B-12 Workflow Process Window - Process Properties Tab

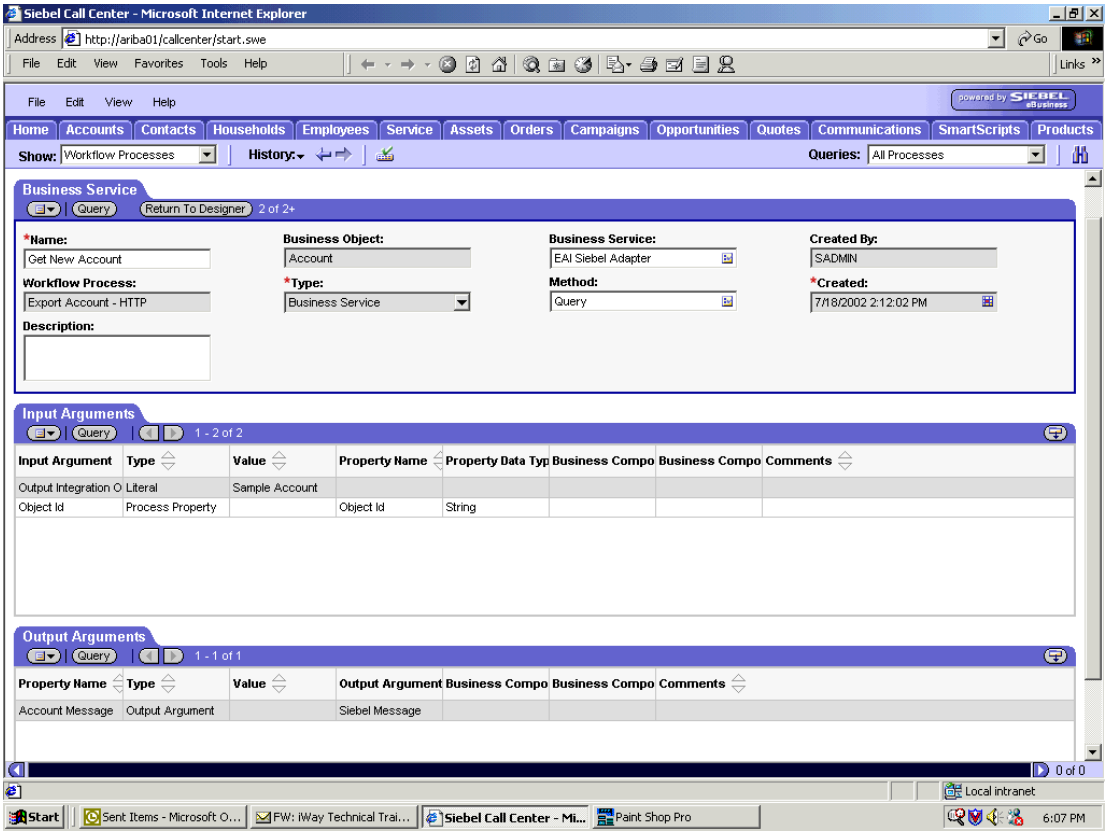


2. Use the Siebel Workflow Administration windows to create a workflow. Define an EAI Siebel Adapter business service step to receive an instance of Account data and call it Get New Account.

   The business service obtains the Account information from Siebel using the Query method.

   Output from this business service is generated in hierarchical format.

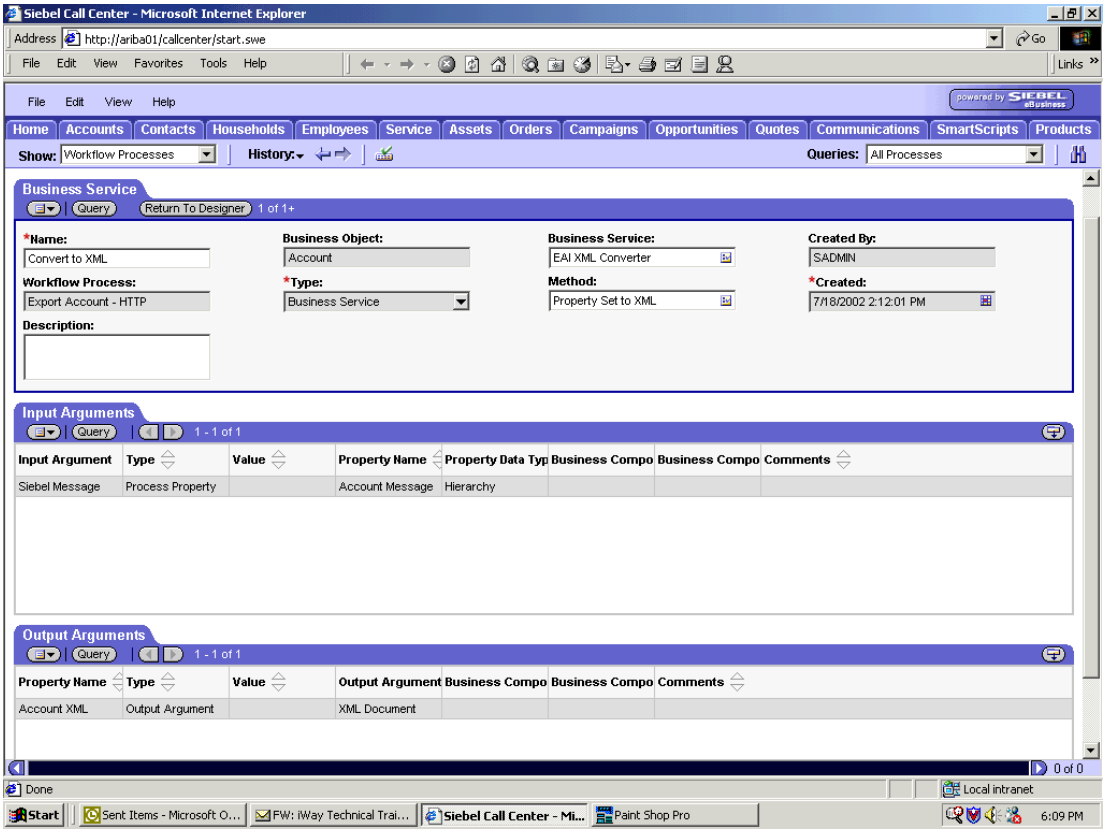
Figure B-13 Business Service Window for EAI Siebel Adapter



3. Define an EAI XML Converter business service step and call it Convert to XML.

   It should be defined to receive the Account data from the EAI Siebel Adapter business service in hierarchical format and convert it to XML format.

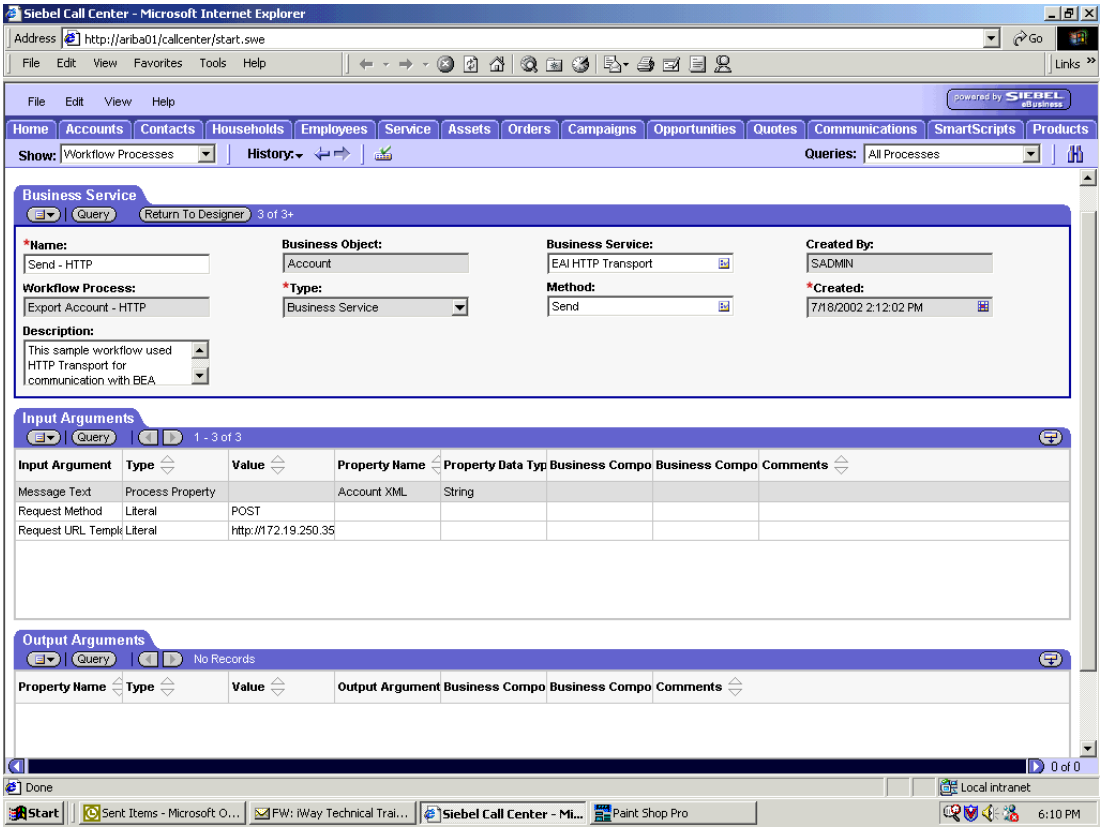
Figure B-14 Business Service Window for EAI XML Converter



4. Define an EAI HTTP Transport business service step and call it Send - HTTP.

   It should be defined to receive the Account data from the EAI XML Converter business service in Siebel XML format and send the Account XML to HTTP using the Send method.

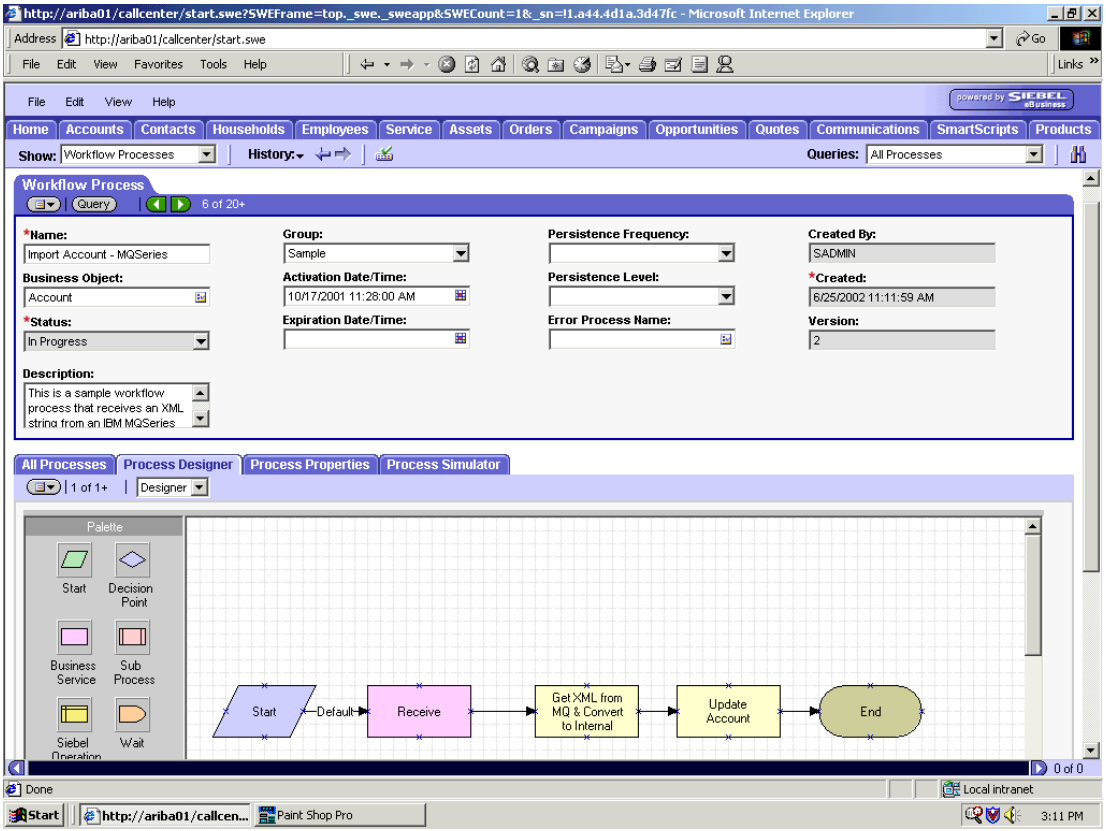
Figure B-15 Business Service Window for EAI HTTP Transport



Creating a Siebel Workflow for Service Using MQSeries Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for importing Siebel Account record information through the MQSeries Transport.

Figure B-16 Workflow Process Window

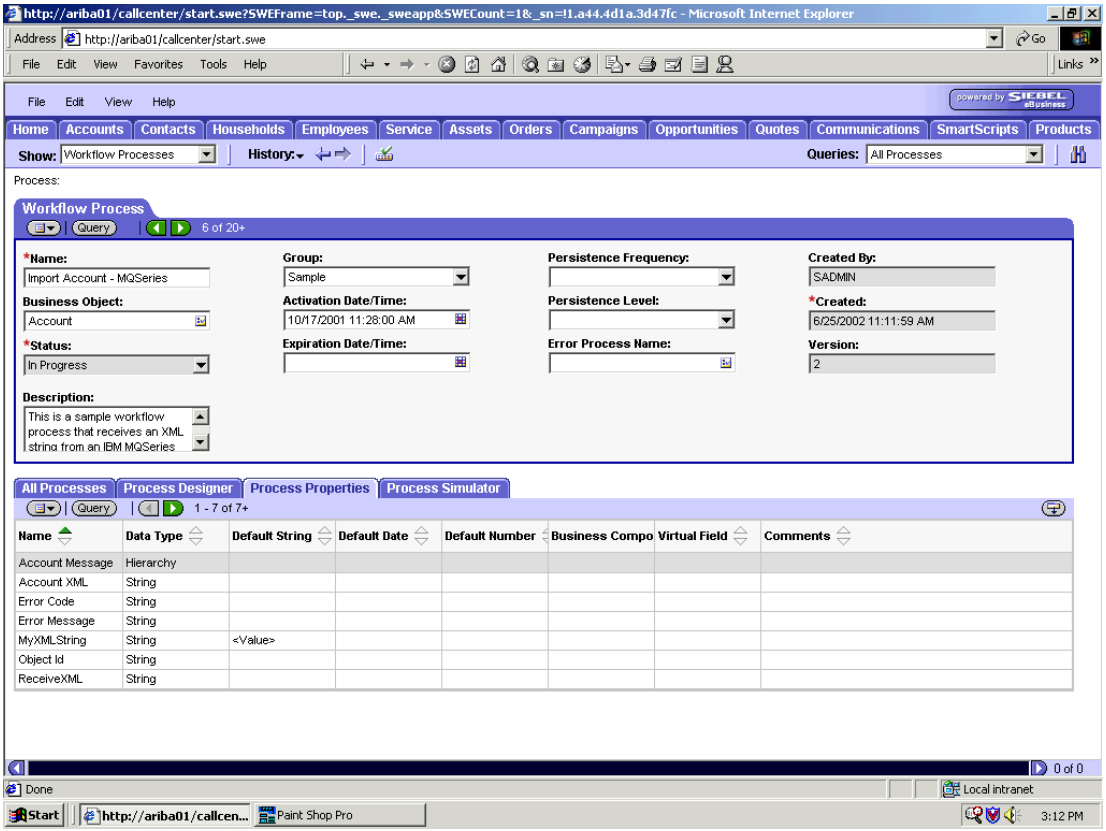


1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   Account message contains the Siebel Account data in hierarchical format.

   Account XML specifies the Siebel Account data that the workflow has converted to XML.

Figure B-17 Workflow Process Window - Process Properties Tab

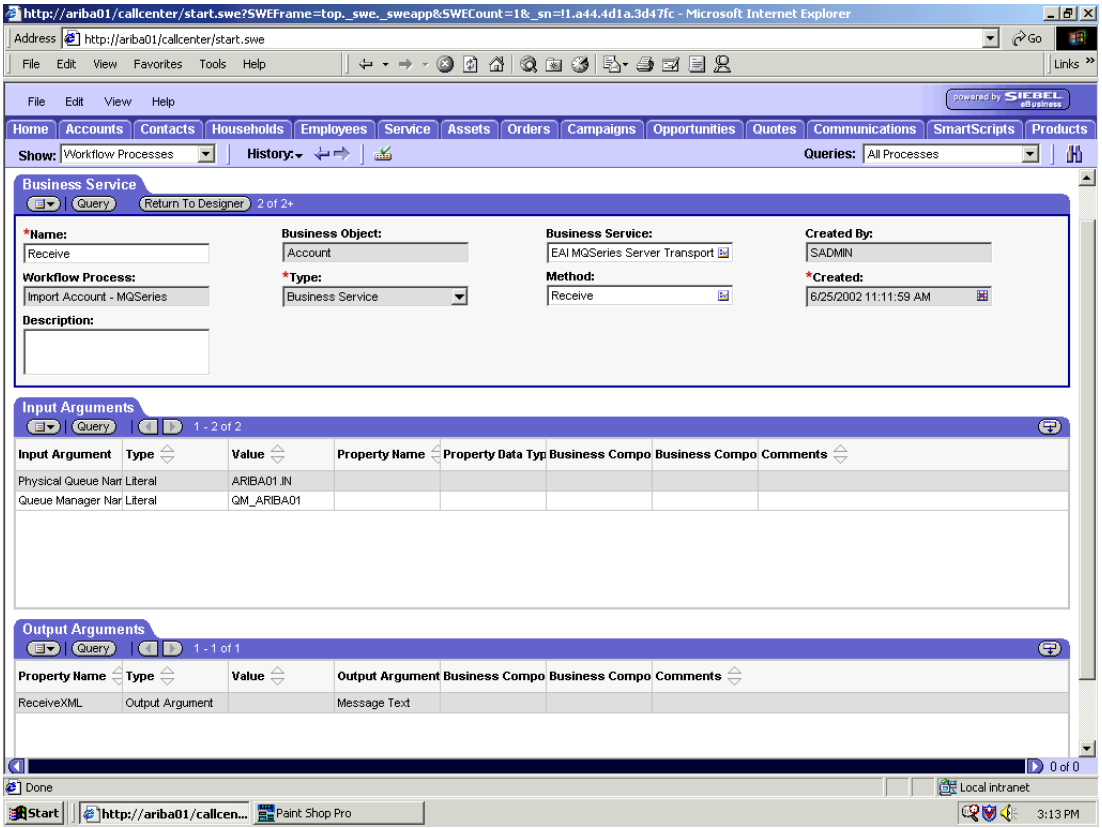


2. Define an EAI MQSeries Server Transport business service step and call it Receive.

   It should be defined to receive the Account data from the IBM MQSeries message queue.

   The EAI MQSeries Server Transport business service receives the Account data in Siebel XML format and sends it to the EAI XML Converter business service.

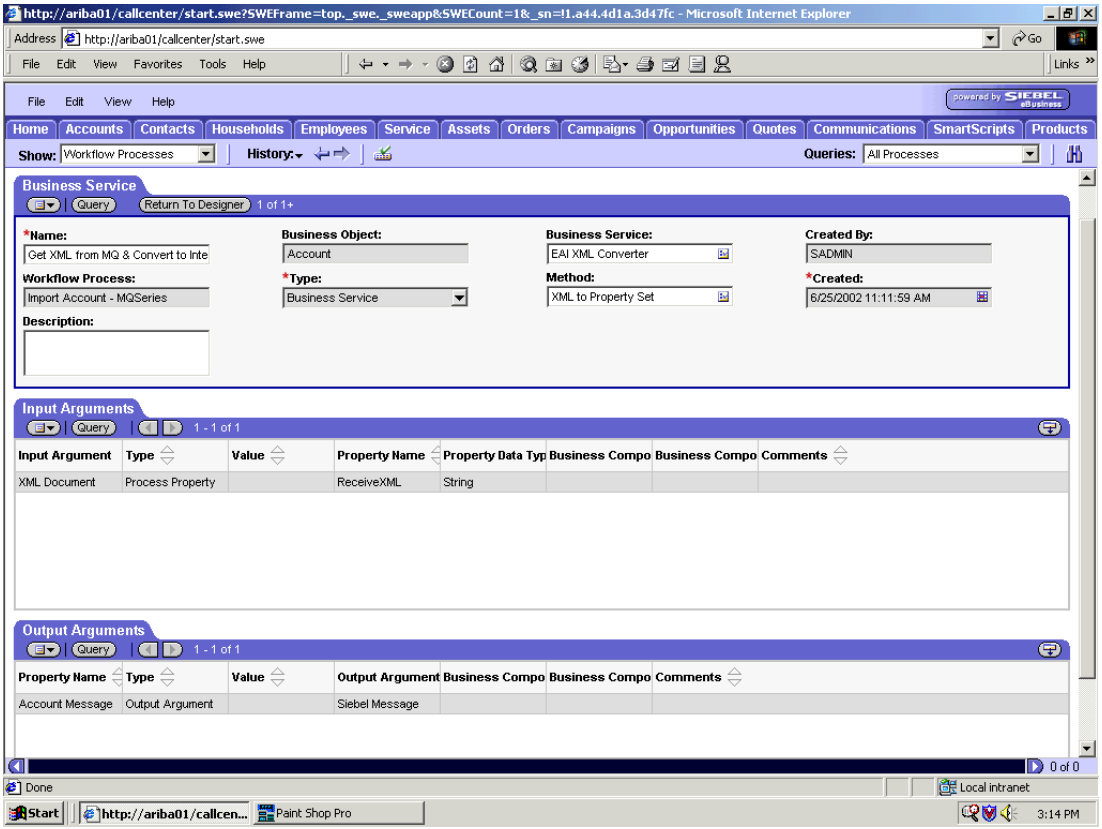
Figure B-18 Business Service Window for EAI MQSeries Server Transport



3. Define an EAI XML Converter business service step and call it Get XML from MQ & Convert to XML.

   It should be defined to receive the Account data from the EAI MQSeries Server Transport business service in XML format and convert it to hierarchical format.

Figure B-19 Business Service Window for EAI XML Converter

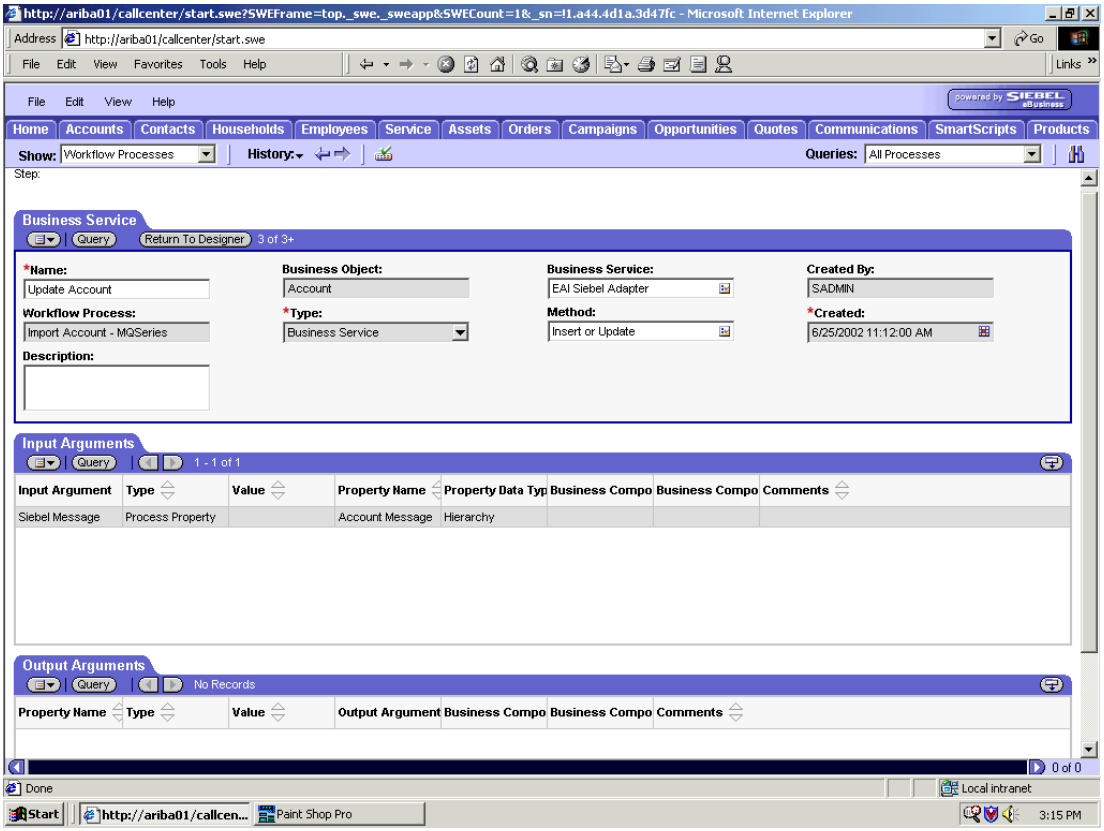


4. Define an EAI Siebel Adapter business service step and call it Update Account.

   It should be defined to receive from the EAI XML Converter business service the instance of Account data in hierarchical format.

   The business service applies the Account information into Siebel using the Insert or Update method.

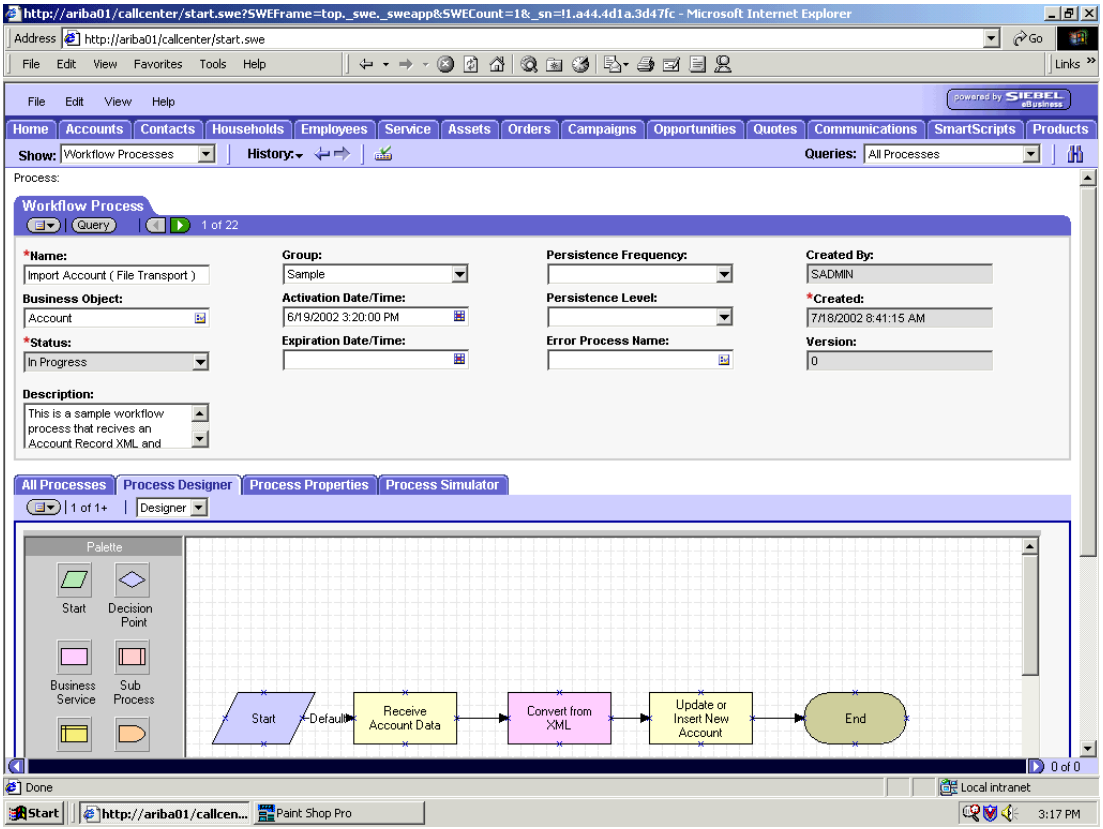
Figure B-20 Business Service Window for EAI Siebel Adapter



Creating a Siebel Workflow for Service Using File Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for importing Siebel Account record information through the File transport.

Figure B-21 Workflow Process Window

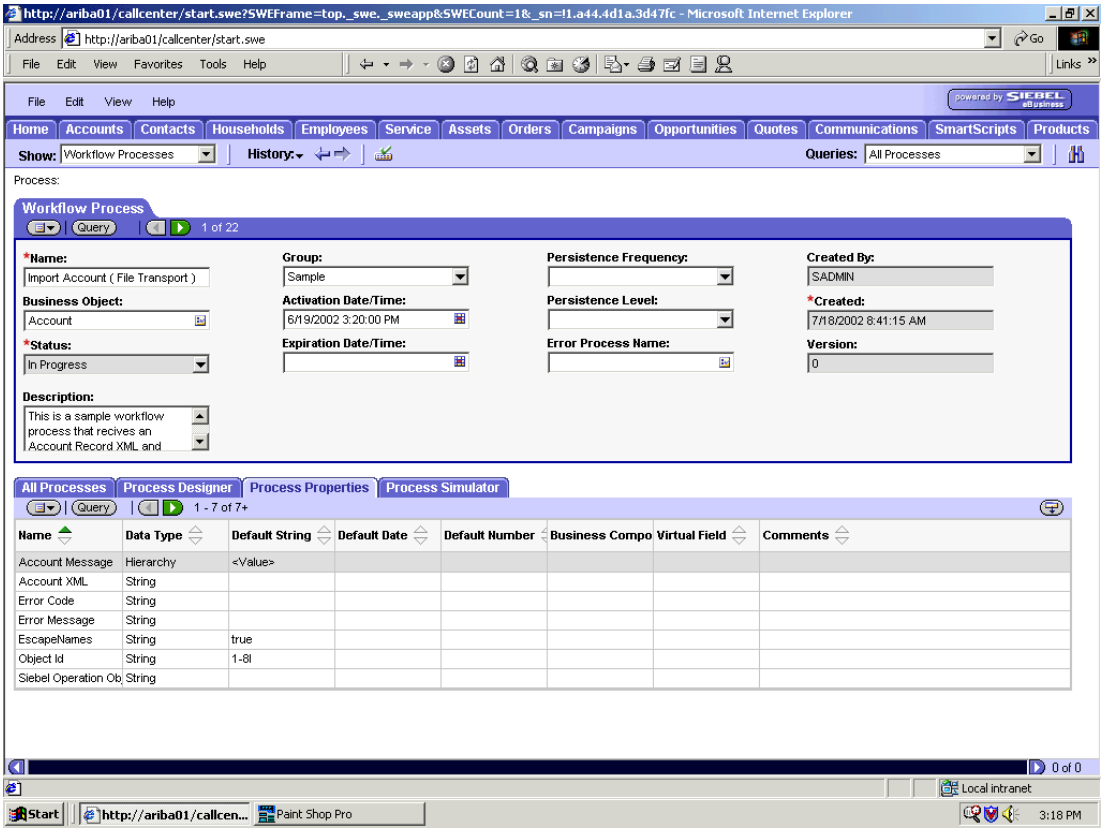


1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   Account message contains the Siebel Account data in hierarchical format.

   Account XML specifies the Siebel Account data that the workflow has converted to XML.

Figure B-22 Workflow Process Window - Process Properties Tab

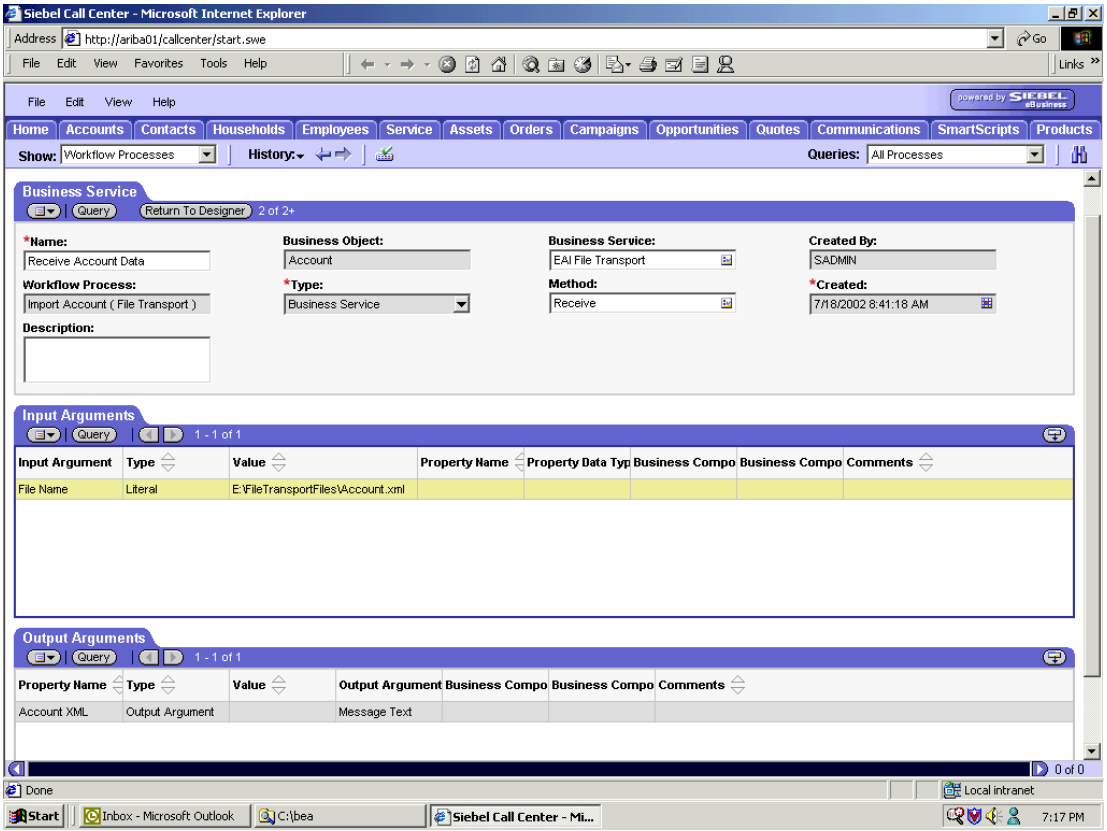


2. Define an EAI FileTransport business service step and call it Receive Account Data.

   It should be defined to receive the Account data from the file system.

   The EAI File Transport business service receives the Account data in Siebel XML format and sends it to the EAI XML Converter business service.

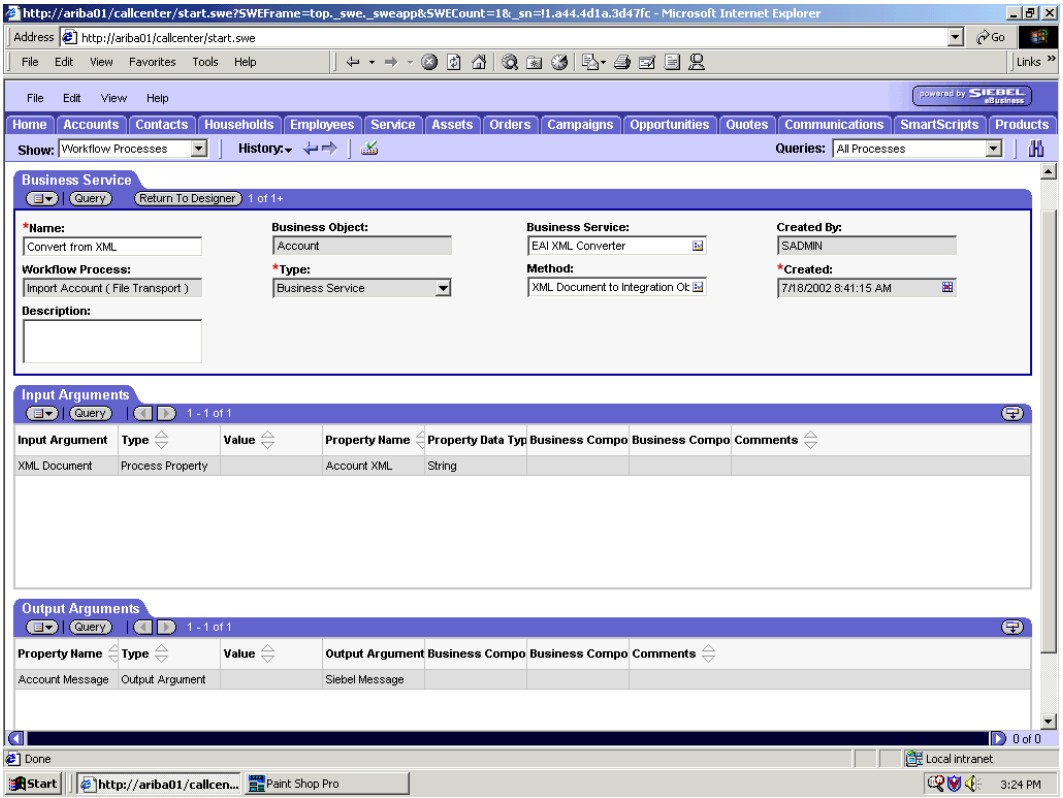
Figure B-23 Business Service Window for EAI File Transport



3. Define an EAI XML Converter business service step and call it Convert from XML.

   It should be defined to receive the Account data from the EAI File Transport business service in XML format and convert it to hierarchical format.

Figure B-24 Business Service Window for EAI XML Converter

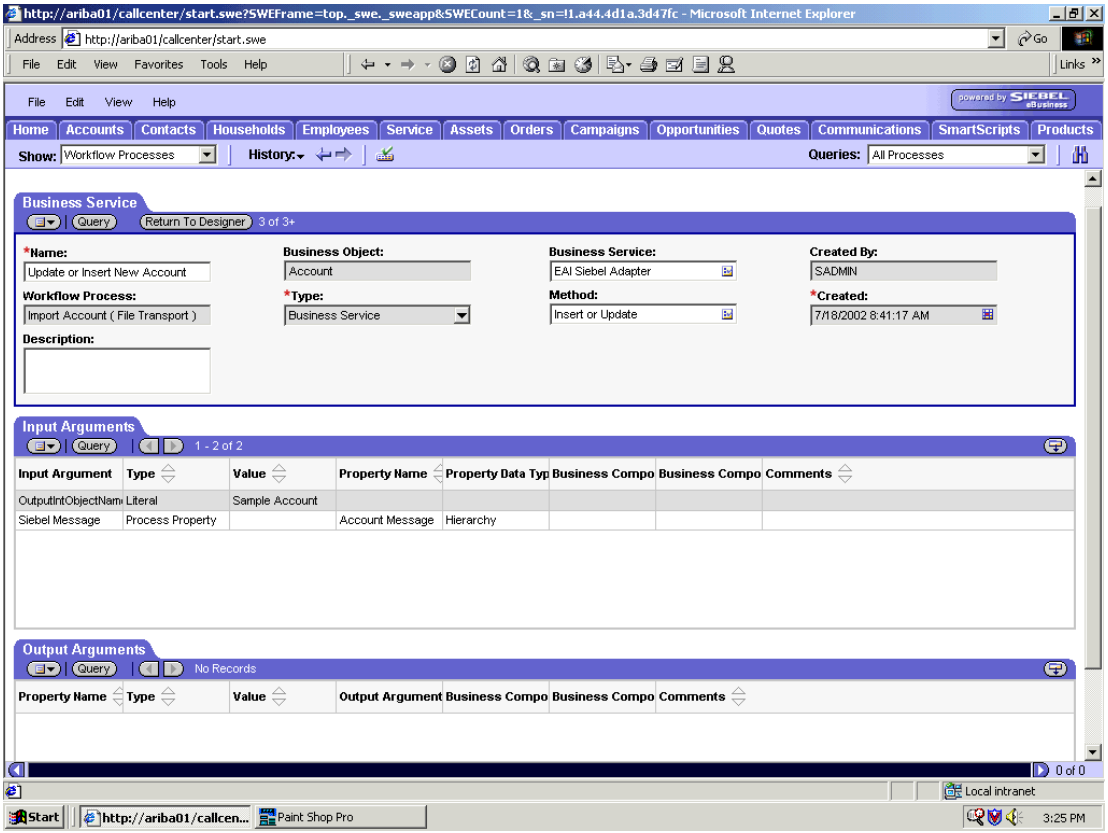


4. Define an EAI Siebel Adapter business service step and call it Update or Insert New Account.

   It should be defined to receive from the EAI XML Converter business service the instance of Account data in hierarchical format.

   The business service applies the Account information into Siebel using the Insert or Update method.

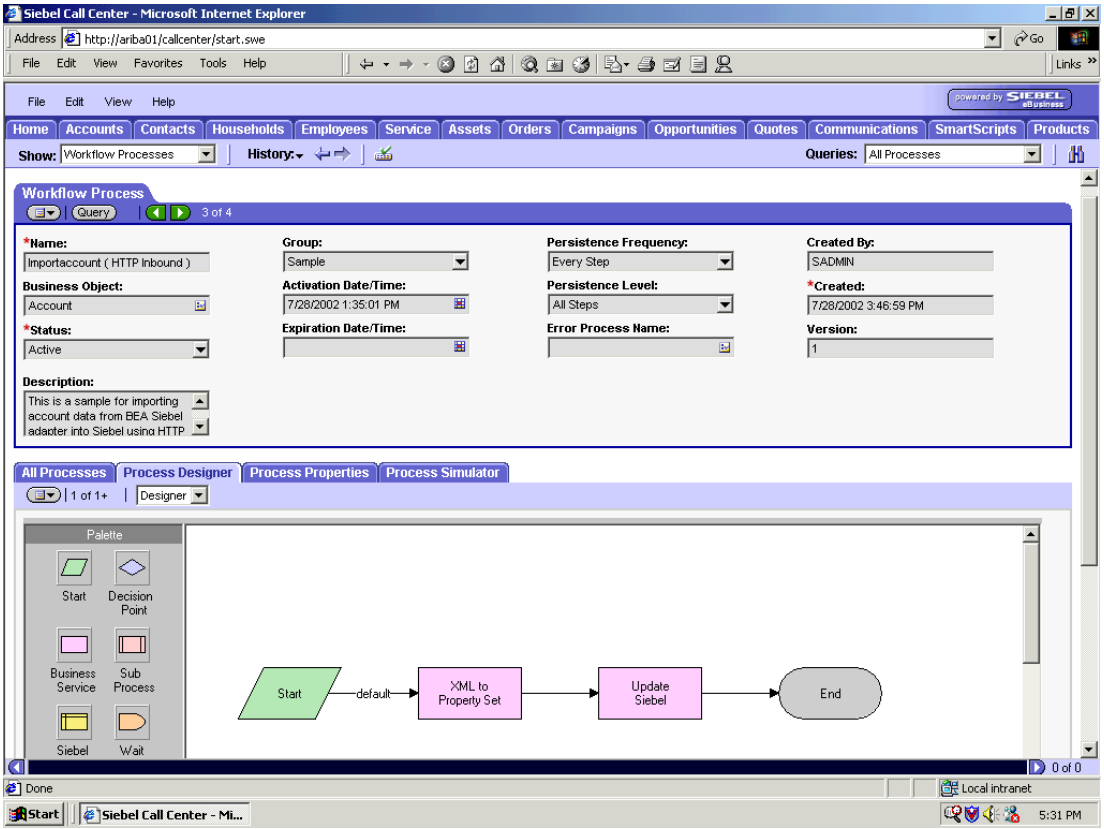
Figure B-25 Business Service Window for EAI Siebel Adapter



Creating a Siebel Workflow for Service Using HTTP Transport

The following is an example of a Siebel Workflow as seen in the Siebel Workflow Administration window. The workflow was designed for importing Siebel Account record information through the HTTP transport.

Figure B-26 Workflow Process Window

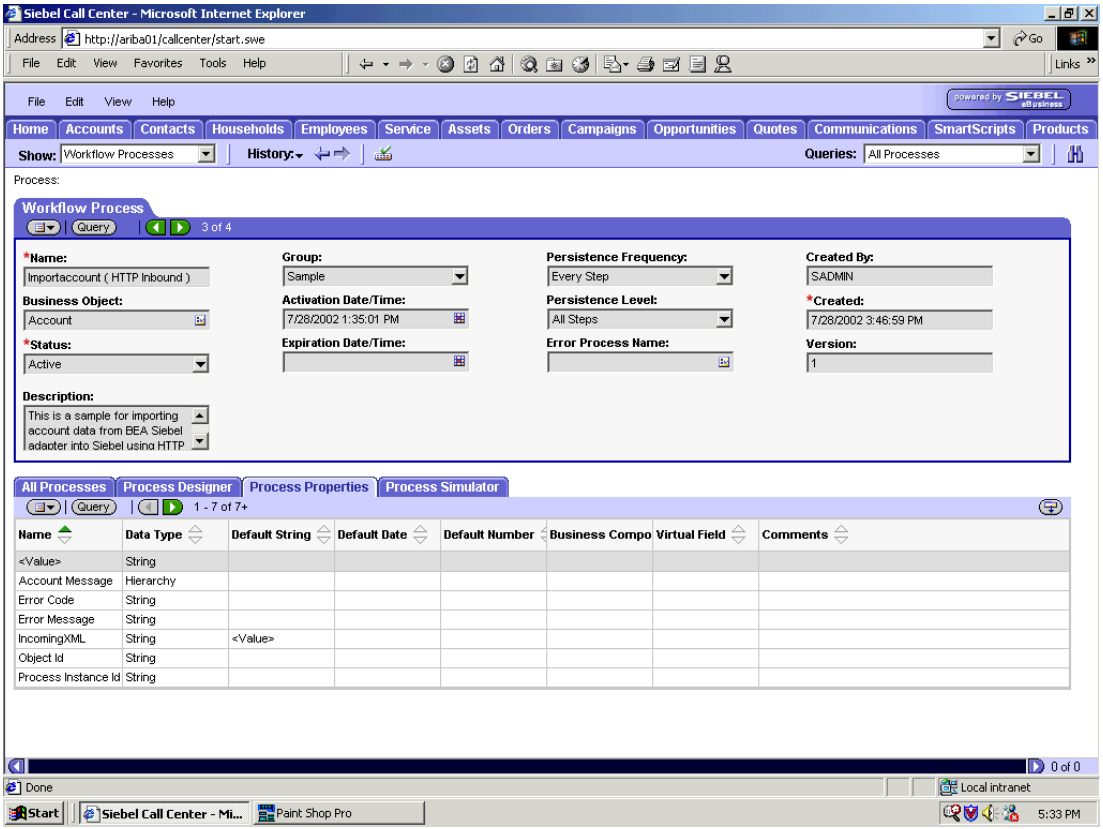


1. In the Process Properties tab of the Workflow Process window, define the Account message and Account XML process properties.

   Account message contains the Siebel Account data in hierarchical format.

   Account XML specifies the Siebel Account data that the workflow has converted to XML.

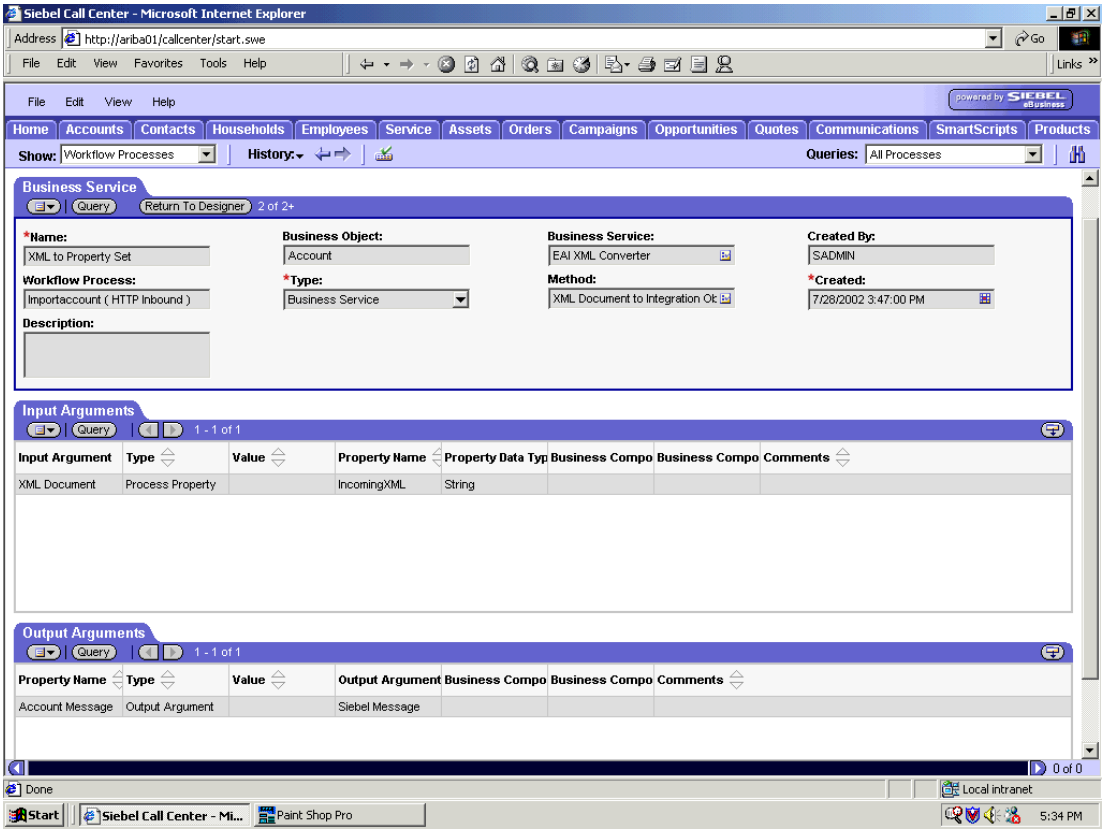
Figure B-27 Process Properties Window



2. Define an EAI XML Converter business service step and call it XML to Property Set.

   It should be defined to receive the Account data from the EAI HTTP Transport business service in XML format and convert it to hierarchical format.

Figure B-28 Business Service Window for EAI XML Converter



3. Define an EAI Siebel Adapter business service step and call it Update Siebel.

   It should be defined to receive from the EAI XML Converter business service the instance of Account data in hierarchical format.

   The business service applies the Account information into Siebel using the Insert or Update method.

Figure B-29 Business Service Window for Siebel Adapter