Siebel

Financial Services Connector for ACORD P and C and Surety Guide

January 2020

Siebel
Financial Services Connector for ACORD P and C and Surety Guide

January 2020

F87420-02

Author: Sukanya Mishra

Contents

Get Help

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *https://docs.oracle.com/*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program website*.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What's New in Siebel Financial Services Connector for ACORD P&C and Surety

What’s New in Siebel Financial Services Connector for ACORD P&C and Surety Guide, Siebel CRM 20.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

What’s New in Siebel Financial Services Connector for ACORD P&C and Surety Guide, Siebel CRM 19.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

2 Overview of Siebel Connector for ACORD XML

Overview of Siebel Connector for ACORD XML

Oracle's Siebel Connector for ACORD XML provides integration between Oracle's Siebel Business Applications and other insurance application systems, such as a policy administration system. The connector supports the ACORD XML Business Message Specification for P&C Insurance and Surety, an insurance industry-standard XML specification.

The Siebel Connector for ACORD XML receives, parses, and processes the business operations specified in the XML message. It handles both outbound and inbound messages.

This integration offers capabilities designed to meet all Property and Casualty message specification requirements. This solution allows you to harness the synergies between Siebel front office applications and ACORD-based applications. Siebel Connector for ACORD XML extends Siebel applications to integrate with back office data and business processes.

The Siebel Connector for ACORD XML supports both synchronous and asynchronous transactions across application boundaries. The resulting consistency and sharing of data allows coordination between front and back office operations. For example, sales and service professionals can enter basic policy information in Oracle's Siebel Financial Services applications and receive a real-time response with a quote for the policy entered. The sales or service professional can then enter the policy details, without ever leaving the Siebel application interface, by completing the requirements and issuing the policy to the customer.

Using the Siebel Connector for ACORD XML

This chapter provides a brief overview of the capabilities of the Siebel Connector for ACORD XML. Additional information about integration with Siebel Business Applications is available in:

- *Overview: Siebel Enterprise Application Integration*
- *Siebel Financial Services Enterprise Application Integration Guide*

Your work with the Siebel Connector for ACORD XML consists of:

- Using the FINS ACORD Wizard to create integration objects to map data between Siebel and ACORD-based external applications.
- Creating integration workflows based on the mapped objects.

You can learn how to build the transformation maps and create workflows from this guide. You can also use some out-of-the-box ACORD messages and workflows defined in this guide as your reference for implementation. Some information on customizing your integration is included in this guide, but you will also need to consult additional guides specified in the text.

Major chapters in this guide provide a description of ACORD rules and syntax, the methods and arguments for configuring Siebel Connector for ACORD XML to customize your integration solution, and a sample implementation showing the steps involved to configure and use the connector.

Required Components

The Siebel Connector for ACORD XML requires the following components in order to implement message exchanges between Siebel Business Applications and ACORD-compliant applications:

- Siebel Financial Services
- A license to use the Siebel Connector for ACORD XML

  For help with The Siebel Connector for ACORD XML license key, create a service request (SR) on My Oracle Support. Alternatively, you can phone Oracle Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

  > **Note:** Siebel Connector for ACORD XML is not automatically available as part of Siebel Financial Services, but must be purchased separately.

- Oracle's Siebel Event Manager to initiate a workflow process through a Siebel workflow manager (optional). In the absence of the event manager, an eScript can initiate a workflow process. Siebel Workflow is delivered as a part of Siebel Financial Services.

> **Note:** You should also be familiar with ACORD XML models. Additional information about these models can be obtained by visiting www.acord.org.

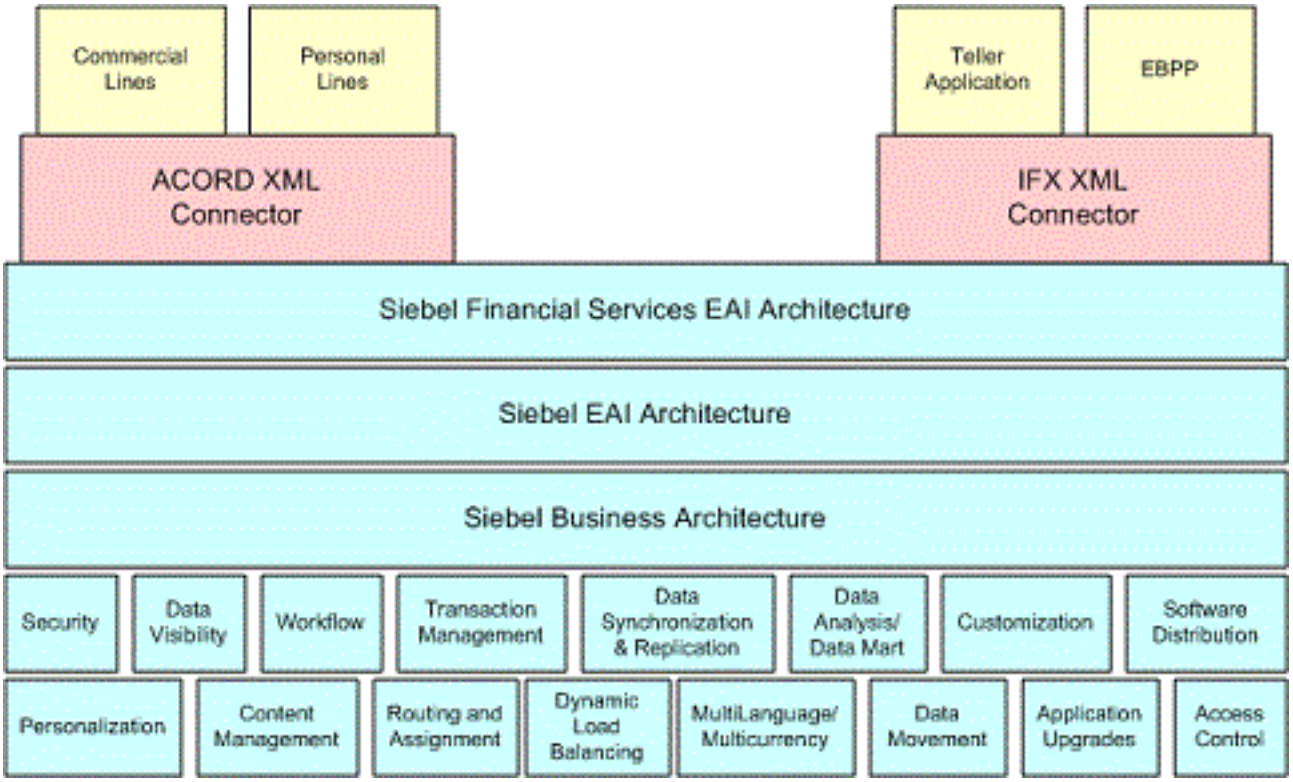
For the purposes of this document, we assume that all these products have been successfully installed and tested for completeness by trained personnel before starting to use the Siebel Connector for ACORD XML for integration. Please refer to *Configuration Roadmap* for implementing integration.

Siebel Connector for ACORD XML Architectural Overview

The Siebel Connector for ACORD XML is a configurable set of components that allow data to be exchanged between your Siebel application and external ACORD-based applications and databases. As shown in the following image, the Siebel Connector for ACORD XML is built on the Siebel Financial Services EAI Architecture, which in turn is built on Oracle's Siebel Enterprise Application Integration (EAI) Architecture. The Siebel Financial Services EAI framework has been built to support XML messaging-based communication infrastructure.

Generally speaking, users of Siebel Financial Services must integrate with many different applications through messaging mechanisms. In order to fulfill this requirement, many connectors have to be built in order to support various industry standards. Siebel Financial Services is in a position to quickly and easily build and deploy multiple connectors based on the flexible Siebel EAI Architecture.

To demonstrate such flexibility, Oracle has built two connectors —ACORD P&C Connector and IFX Connector—both based on the Siebel Financial Services EAI framework. Please refer to *Siebel Financial Services Enterprise Application Integration Guide* for more information about the flexible Siebel Financial Services architecture.



The Siebel Connector for ACORD XML is based on the ACORD XML standard for insurance industry exchange. In the Property and Casualty business, the main driver to the Internet is the real-time exchange of data between producers, carriers, rating bureaus, service providers, and others. The ACORD XML standard is designed to address these requirements by defining P&C transactions that include both a request and a response message.

ACORD partially leverages from the existing Interactive Financial Exchange (IFX) specification as the base protocol while we define an Insurance service containing Personal Lines, Commercial Lines, Surety, Claims, and Accounting transactions. It provides functions such as:

- Handling the XML message header
- Handling heterogeneous commands in the body section of an XML message
- Data type formatting and conversions
- Data model mapping through the various connector modules

These Siebel Connector for ACORD XML modules include the FINS ACORD Wizard, the FINS ACORD XML Dispatcher, the FINS ACORD XML Converter, the FINS ACORD XML Data Transformation Engine (DTE), and the FINS ACORD XML Transaction Manager.

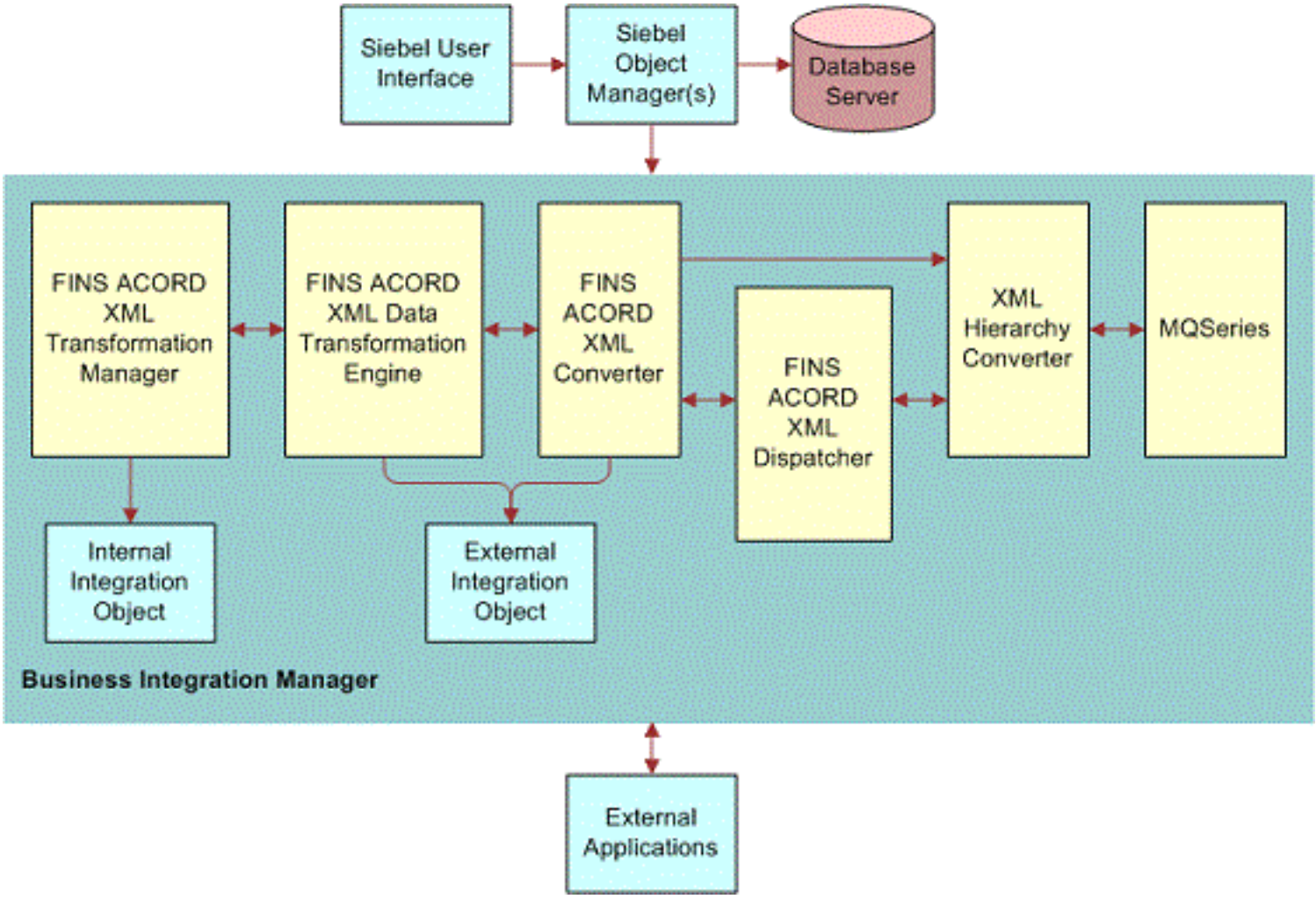
Business Data Flows

Each standard integration or custom integration is based on business data flows. A business data flow controls the transformation of an ACORD-based data object to a Siebel data object and a Siebel data object to an ACORD-based data object.

There are two types of business data flows:

- Outbound to an external ACORD-based application (Send)
- Inbound from an external ACORD-based application (Receive)

The following illustrates both inbound and outbound business data flows.



The business data flows consist of XML messages in the format published by ACORD, such as<HomePolicyAddRq>, <HomePolicyQuoteInqRq>, <PersAutoPolicyAddRq>, <PerAutoPolicyQuoteInqRq>, <PersUmbrellaPolicyAddRq>, <PersUmbrellaPolicyQuoteInqRs>. A significant portion of the ACORD messages are provided as examples for your reference in the sample database.

The processing for each type of data flow is contained within a Siebel workflow. The workflow process is initiated by Oracle's Siebel Event Manager or by a call from Oracle's Siebel eScript.

Outbound Data Flow

The image in the Business Data Flows illustrates an outbound data flow as well as an inbound data flow. During an outbound data flow:

1. When the workflow is initiated, the FINS ACORD XML Transaction Manager extracts data from the Siebel database.

   The transaction manager takes as input all the ROW_IDs of the objects. This data is then used to instantiate the internal integration objects based on the Siebel business objects.

   The transaction manager returns all the instances retrieved as Siebel property sets. A property set is a representation of data in memory in the Siebel internal format. It is used widely by the business services that constitute the connector components.
2. The internal integration object instances are then passed to the FINS ACORD XML Data Transformation Engine (DTE) to transform the internal integration object instances into external integration object instances.

   The DTE also adds all necessary ACORD-specific command layer attributes into the instances transformed.
3. The FINS ACORD XML Converter converts all external integration object instances into proper XML integration object instances. It also adds the envelope, header, and other sections to the newly converted instance.
4. Lastly, the XML Hierarchy Converter converts the XML integration object instance from a property set format into a text format.
5. The message is then sent to external systems using any transport mechanism supported by Siebel EAI.

Note: *Business Data Flows* depicts the connector process using the MQ Series transport adapter. However, the transport mechanism can be HTTP, MSMQ, or any other transport mechanisms supported by Siebel EAI.

Inbound Data Flow

The image in *Business Data Flows* illustrates an inbound data flow as well as an outbound data flow.

Inbound business data flows start with a receiver server component such as the MQSeries, HTTP, or MSQM.

The receiver runs in the background continuously, waiting for data from external ACORD-based applications. When the receiver receives an ACORD message, it invokes the workflow process configured to handle and process the data. The workflow typically dictates the whole Siebel Connector for ACORD XML business logic.

1. The raw XML text string is passed through the XML Hierarchy Converter to be converted into an XML integration object instance.
2. The FINS ACORD XML Dispatcher then takes in the XML instance, parses it and identifies the messages received according to the rules set forth in the dispatcher map. The FINS ACORD XML Dispatcher identifies the envelope, header and body sections. The dispatcher then associates the appropriate internal and external integration objects to the message so that it can be processed by the converter.

   The dispatcher map is an integration object created by the FINS ACORD Wizard.
3. The FINS ACORD XML Converter then takes the XML instance, and processes individual sections of the instance while converting each sub-tree into external integration object instances.
4. The FINS ACORD XML DTE transforms the external integration object instances into internal integration object instances.

5. The internal integration object instances are passed to the FINS ACORD XML Transaction Manager which performs the operation specified in the instance (such as insert/update/delete) through the invocation of other business services configured in its user properties.

Workflow Integration

Siebel workflows control the flow and transformation of data into and out of Siebel applications. You create a workflow using the Workflow Designer, a graphical user interface provided within the Siebel applications. Siebel workflows provide many more capabilities than those described in this guide. For more information about Siebel Workflow, see *Siebel Business Process Framework: Workflow Guide* .

Integration Objects

Integration objects are the data containers used within the workflow environment. They represent the data structure of either a Siebel business object or an external application's data object.

You can create integration objects with the integration object wizard provided in Oracle's Siebel Tools. The integration object wizard can create Siebel integration objects from Siebel business objects.

For ACORD integration work, please use the FINS ACORD Wizard in Siebel Tools that reads an ACORD Document Type Definition (DTD) and creates the required external integration objects, pairs them with the internal integration objects, creates the envelope and header integration objects, and finally associates all of these in the rule-based dispatcher map.

This document describes how to use the FINS ACORD Wizard to complete design time requirements. For more information about the FINS ACORD Wizard see *Siebel Connector for ACORD XML* in this document. For more information on the integration objects read *Overview: Siebel Enterprise Application Integration* .

Business Services

All of the connector components are Siebel business services. Business services execute predefined or customized actions in a workflow process. Examples of business services include the FINS ACORD XML Transaction Manager, Siebel EAI Adapter, and the FINS ACORD XML Converter.

Siebel business services act on property sets passed to them. They perform business logic operations such as interfacing with a database, interfacing to ACORD-based systems, or transforming one integration object into another.

Business services have object-like qualities, such as methods, method arguments, and user properties. These elements define how a business service can be used. Although business services can be used to perform many different functions, they all have a standard interface.

Oracle provides many business services, and you can create your own. Business services are defined in Siebel Tools.

This guide describes those business services used to interface to ACORD-based systems. For more information on business services in general, read *Integration Platform Technologies: Siebel Enterprise Application Integration .*

ACORD P&C and Surety XML Standard

When handling insurance application information, your Siebel application implements the *ACORD DTD for the ACORD XML P&C Insurance and Surety Standard* to connect with external applications. The DTD is required by Siebel Connector for ACORD XML.

You can find the ACORD DTD, along with complete documentation, at the following location: www.acord.com. Be certain to use the appropriate version of the ACORD DTD, as described in *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

3 Siebel Connector for ACORD XML

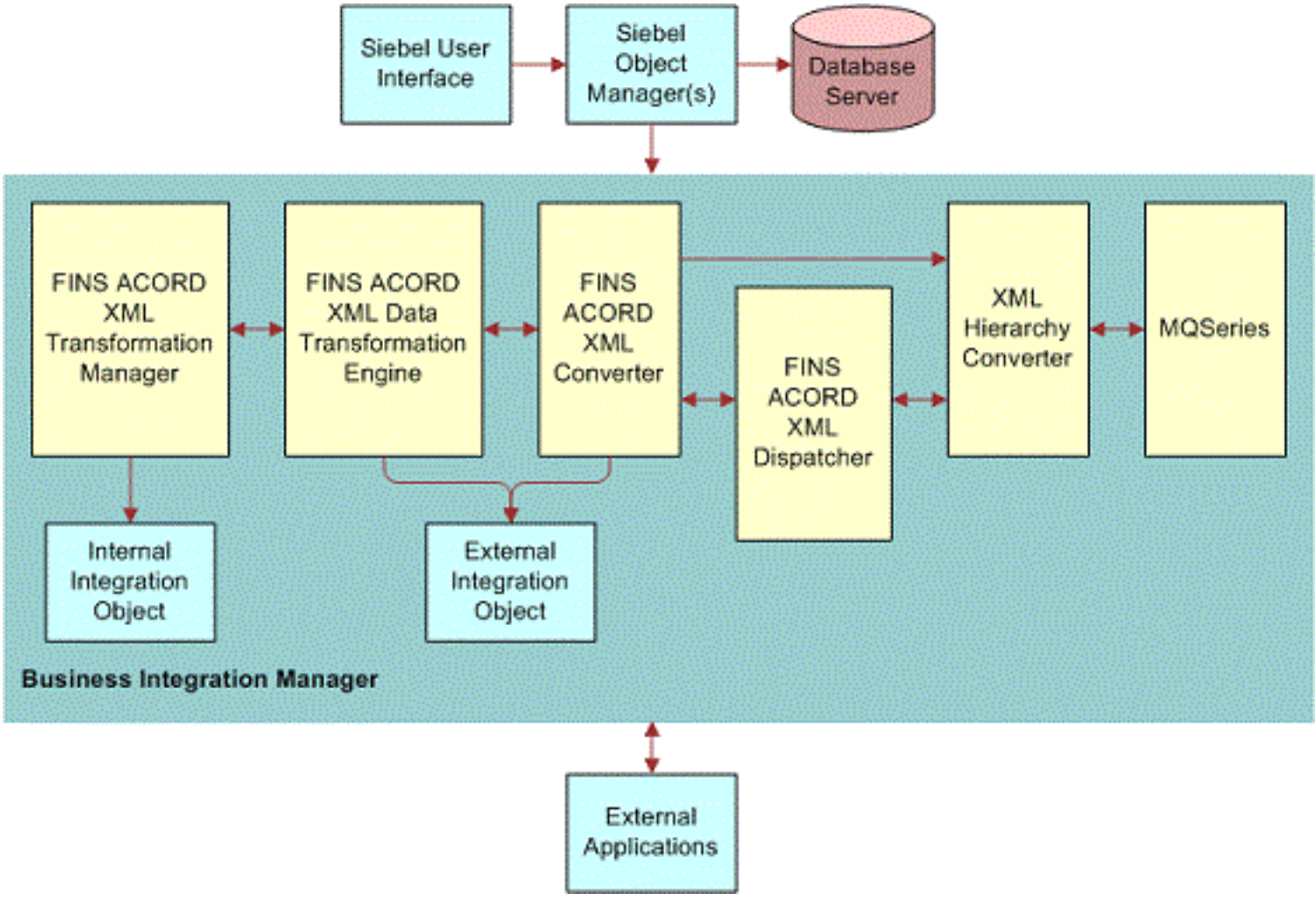
Siebel Connector for ACORD XML

This chapter describes the methods, input arguments, and output arguments for configuring the components of a Siebel Connector for ACORD XML.

The Siebel Connector for ACORD XML consists of the following components:

- Transaction manager
- Transformation engine
- Converter
- Dispatcher
- Transport adapter

The following image shows the connector components.



The connector components are Siebel business services, which are configured in the Workflow view. The integration objects are created using the FINS ACORD Wizard, and they are configured using the Data Map editor.

Note: For information about Siebel integration objects, converter elements, and XML, see *XML Reference: Siebel Enterprise Application Integration* .

ACORD XML Syntax and Rules

ACORD P&C and Surety is an insurance-industry version of XML. It is used for messages that are appropriate for Property and Casualty (P&C) and Surety forms of insurance.

The ACORD standard defines the required structure and format of an XML message for use in a Siebel connector. The definition is in the ACORD DTD, and the ACORD DTD is incorporated by the Siebel connector to construct messages.

This section provides a summary of the ACORD XML syntax and rules, and provides the appropriate vocabulary for discussing ACORD XML messages. This section supplies knowledge that is basic for any troubleshooting you may need to do.

ACORD XML Documents

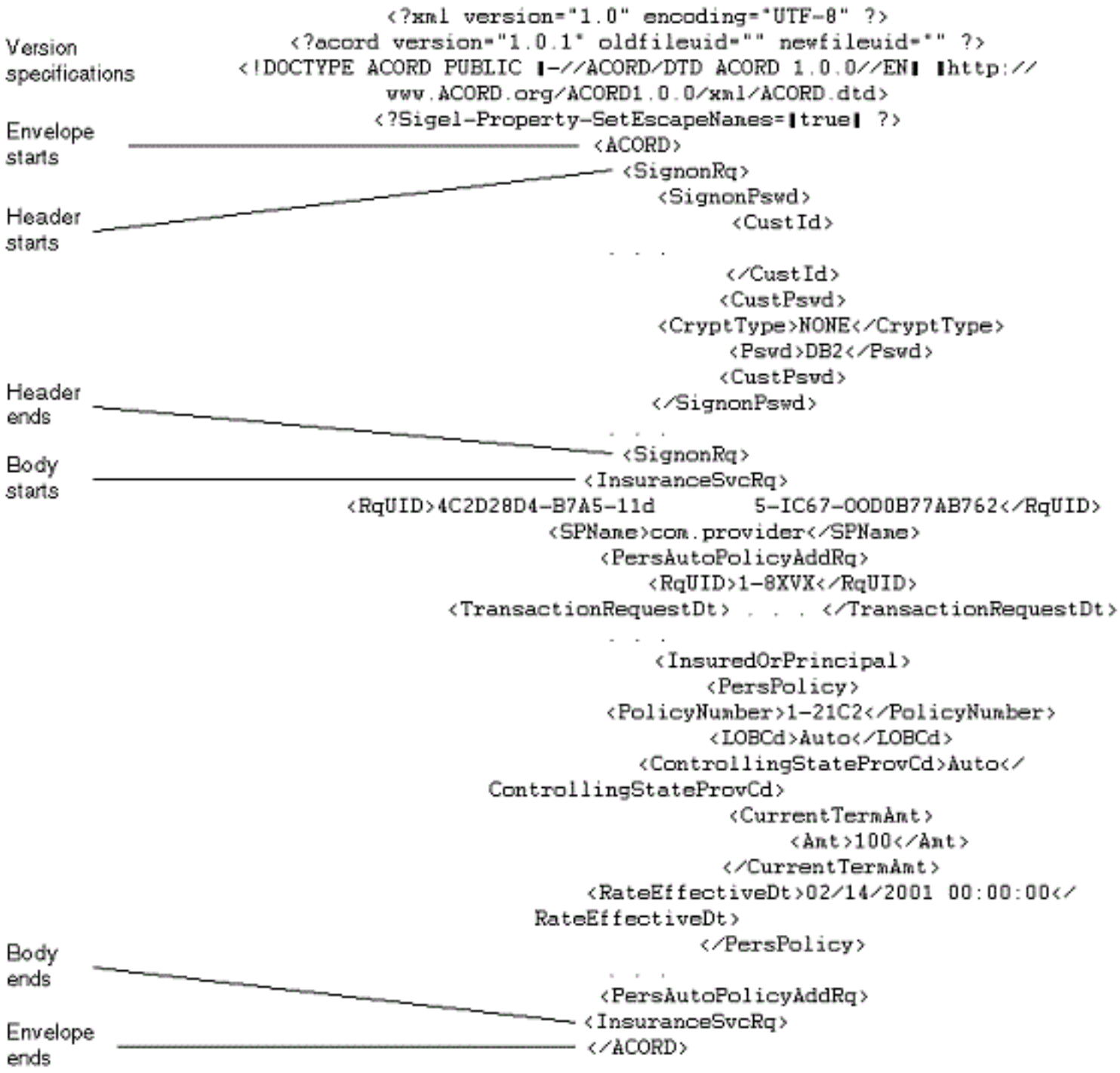
Each XML document has three distinct parts:

- Envelope
- Header
- Body

The parts are presented as a hierarchy: the envelope is the root, which contains the header and the body. Elements of an ACORD XML document that contain other elements are called aggregates.

The envelope and header provide information required by the XML converter and by other components in the connector. The services identify the kind of business service affected by the information, and the messages provide the data that is being exchanged. There are elements that precede the message proper, which specify the versions of XML and ACORD.

The following image shows a sample ACORD XML document.



Envelope

The envelope is the root element of an XML document. For an ACORD XML document, it begins with <ACORD> and ends with </ACORD>.

The indicator <ACORD> is the only item in the envelope.

Header

Every message header has a sign-on element that authenticates the message, and it may have a sign-off element that ends a particular session.

The header has five possible elements (currently supported):

- SignonRq
- SignonRs
- SignoffRq
- SignoffRs
- Status

The header for a request has the header element <SignonRq>. The header for the response has the header element <SignonRs>. Similarly, the sign-off elements are specifically for requests and responses. The <Status> element provides status and error information.

> **Note:** ACORD XML messages must be either requests or responses. Requests and responses cannot be mixed in a single message. A request uses <SignonRq>. A response uses <SignonRs>.

Signon Information

The <SignonRq> or <SignonRs> header element provides a location for authentication information, date and time stamps, language preferences, and identification of the application that will use the data. You can find complete information in the ACORD specification.

Authentication Information

The *initial* <SignonRq> for any session must provide authentication information, typically the user name and password, or a certificate ID. When the server authenticates the user, using the information in the header, the server issues a session key in the <SignonRs>. Subsequent messages use the session key as a token. After a session has finished, any subsequent session must start with the authentication information again.

The following is an example of authentication information included in a <SignonRs> element. The response includes a session key for authentication, in the <SessKey> element, issued by the server after the initial request message was received.

```
<SignonRs>

<ClientDt>1001-10-02T19:21:06.9-07:00</ClientDt>

<CustLangPref>ENU</CustLangPref>

<ClientApp>

<Org>Oracle/Org>

<Name>Siebel Financial Services</Name>

<Version>8.0</Version>

</ClientApp>

<ServerDt>1001-10-02T19:21:06.9-07:00</ServerDt>

<SessKey>SNOVICEsnoviceadmin</SessKey>
```

```
<Language>ENU</Language>

</SignonRs>
```

Signoff Information

The <SignoffRq> and <SignoffRs> header elements are used to end a session. A typical time to end a session is at the close of business for the day.

The Signoff element, <SignoffRq> or <SignoffRs>, appears at the end of the message, just before the end of the envelope </ACORD>. The Signoff element may optionally contain a <cust ID> element.

<Status>

The <Status> header element may contain error codes and error messages. For additional information about the kind of information in a <Status> element, see *Status Information and Error Codes*.

> **Note:** Siebel Connector for ACORD XML does not at this time support the <SuppressNotificationInd>, <SuppressedNotificationInd>, and <PendingResponseInfo> header tags. Support for these tags is expected in the future, depending on customer need.

Body

The body of an ACORD XML document provides the content of the information request or response. The body serves as an aggregate containing services and messages. Services and messages, in turn, are aggregates that contain smaller elements.

- **Service.** A service identifies the kind of service being requested or delivered, and identifies the business function that will be affected. For example <InsuranceSvcRq> is a request for an insurance service.
- **Message.** A message identifies the business object affected by the message and the operation that is to be performed on the data. For example, <PersAutoPolicyAddRq> is a request to add a personal automobile policy.
- **Data Element.** A data element identifies the business component or fields affected by an operation defined in the message. For example, <PersVeh> is a data element that contains information about a vehicle.

Services

The basic body element is a service, for example <InsuranceSvcRq>, <BaseSvcRq>, or <SuretySvcRq>. <BaseSvcRq> is a request for the Base service, which all service providers can provide.

An ACORD body can include multiple services. A body almost always contains at least one service. A body with no service would provide only authentication.

The same service may be included in a body more than once, but each service must be for a different service provider.

The following is an example of a message with a single insurance service request.

```
<InsuranceSvcRq>

 <RqUID>4C2D28D4-B7A5-11d 5-IC67-OOD0B77AB762</RqUID>

 <SPName>com.siebel</SPName>

 <PersAutoPolicyAddRq>

 <RqUID>1-8XVX</RqUID>

 <CurCd/>
```

```
<Producer>
<GenPartyInfo>
<NameInfo>
<CommlName>
<CommercialName>LOGONXNXN</CommercialName>
</CommlName>
</NameInfo>
</GenPartyInfo>
</Producer>
<InsuredOrPrincipal>
<GenPartyInfo>
<NameInfo>
<PersonName>
<Surname>Aaron</Surname>
<GiveName>Mary</GiveName>
</PersonName>
</NameInfo>
</GenPartyInfo>
<InsuredOrPrincipal>
<PersPolicy>
<PolicyNumber>1-21C2</PolicyNumber>
<LOBCd>Auto</LOBCd>
<ControllingStateProvCd>Auto</ControllingStateProvCd>
<CurrentTermAmt>
<Amt>100</Amt>
</CurrentTermAmt>
<RateEffectiveDt>02/14/2001 00:00:00</RateEffectiveDt
</PersPolicy>
. . .
</PersAutoPolicyAddRq>
</InsuranceSvcRq>
```

The service aggregate includes a universally unique identifier (UUID) to match responses to requests. The UUID is generated using an algorithm that makes it unique. It appears in the <RqUID> element. It is generated by the client (which sends out the request). It is stored at the client site, which then matches it to the UUID in the response message.

The UUID generator can be a Siebel business service or an extension provided by a third party. In any case, the UUID generator is identified by a parameter to the FINS ACORD XML Converter.

Messages

Messages (sometimes called business messages) are contained in service aggregates. Each service can contain any number of messages.

The message tag identifies the business object that is affected by the message and a command operator. A business object can be a personal auto insurance policy, or a surety policy—anything on which an operation can be performed.

A message uses one of the following operations:

- Add
- Inquiry
- Submit

Note: Additional operations—Modify, Cancel, and Delete—will be supported in future releases of Siebel Connector for ACORD XML, if future versions of the ACORD DTD supports them.

The business message name tag contains the object and the operation. For example, a business message called <PersAutoPolicyAddRq> identifies “personal auto policy” as the business object, and “add” as the operation. The details of the added policy are provided within the message.

A complete list of business messages for ACORD XML is provided in the ACORD XML implementation specification.

Data Elements

Within the business message are additional elements that identify the record that should be affected by the request or response and provide any other specifications, such as <PersonName>, <PolicyNumber>, <DriverInfo>, and <Coverage>.

The additional elements include field labels, field information, and tags that provide program access to the data.

The following illustrates data elements in an add personal auto policy request.

```
<PersVeh>
 <ItemIdInfo>
 <OtherIdentifier>
 <OtherId>1-9XX1</OtherId>
 </OtherIdentifier>
 </ItemIdInfo>
 <Manufacturer>Honda</Manufacturer>
 <Model>Civic</Model>
 <ModelYear>2000</ModelYear>
 <VehBodyTypeCd>Private Passenger Vehicle</VehBodyTypeCd>
. . .
```

```
</PersVeh>
```

The information in the add personal auto policy request is sent to the external application, which performs the request and returns a response.

Status Information and Error Codes

Status information is information about the current status of a message. It appears only in response documents. It can appear in a response header or in any body element. In the header, it appears in a <Status> tag. In messages in the body, it appears in a <MsgStatus> tag.

Status information in the header applies to the entire ACORD XML document. Status information in a service applies to that service. Status information in a message applies to that message.

The external server generates status information after processing the document. If the processing is satisfactory, status information may or may not be generated. If there is a problem in the processing, the status information identifies the problem.

For details of status codes, see the ACORD XML Business Message specification, which describes the status codes.

The following is an example of status information in a message. Note that it uses the <MsgStatus> tag.

```
<MsgStatus>
<MsgStatusCd>Error</MsgStatusCd>
<MsgErrorCd>FINS IXML Transaction Manager: Multiple matches found
for instance of integration component 'Ins Policy_Position' using
search specification '[Active Login Name]="VSILVER"' in the
business component 'Position', based on user key 'User Key:1'.
</MsgErrorCd>
</MsgStatus>
```

The <MsgStatusCd> information, such as “Error,” “Rejected,” or “Success,” shows the result of processing. If the value of <MsgStatusCd> is “Error,” <MsgErrorCd> provides additional details.

FINS ACORD Wizard

Siebel Business Applications provide wizards to guide you through the process of building integration objects and updating dispatcher maps.

You can use the FINS ACORD Wizard to build integration objects for Siebel Connector for ACORD XML. The wizard guides you through the process of selecting objects (from the Siebel repository or from an external system) on which you can base your new Siebel integration object. The wizard builds a list of valid components from which you choose the specific components to be included in your Siebel integration object.

You access Siebel wizards within Siebel Integration Object Builder in Siebel Tools. Use the FINS ACORD XML Wizard to create an appropriate elements hierarchy that works with the ACORD DTD. The wizard:

- Creates a set of integration objects to handle outbound and inbound messages and to handle internal and external integration.
- Updates the dispatcher map, which is later used by the dispatcher.

Integration Objects

Siebel integration objects allow you to represent integration metadata between a Siebel business object and an external XML standard, using the FINS ACORD XML Data Transformation Engine (DTE). The integration object represents a common structure that the EAI infrastructure can understand.

Because these integration objects adhere to a set of structural conventions, they can be traversed and transformed programmatically, using Oracle's Siebel eScript objects, methods, and functions, or transformed declaratively using Siebel Data Mapper.

To use Siebel Connector for ACORD XML to integrate data, you need to build three different integration objects:

ACORD Envelope Integration Object. An envelope integration object provides envelope and header information for an ACORD XML document.

User properties in an ACORD envelope provide flexibility to the connector. For example, when a user sends an initial ACORD request, the ACORD document uses a <SignonRq> header that is different from subsequent <SignonRq> headers. Two different integration component user properties, initsignon and sessionsignon, can be used to construct different headers under the same envelope. See *Configuration Roadmap* for an example of creating an envelope integration object.

ACORD Internal Integration Object. An internal integration object represents the Siebel business object hierarchy for a particular Siebel business object. See *Configuration Roadmap* for an example of creating an internal integration object.

ACORD External Integration Object. An external integration object represents the ACORD XML hierarchy for a particular ACORD message. See *Configuration Roadmap* for an example of creating an external integration object.

FINS ACORD XML Dispatcher Map

The dispatcher map is used by the FINS ACORD XML Dispatcher. The dispatcher map is an integration object that provides a rule set for handling incoming ACORD XML messages. The dispatcher map is created and updated by the FINS ACORD Wizard during the process of creating external and internal integration objects.

The map contains information that associates message instances with the appropriate internal and external integration objects for incoming and outgoing messages. It associates each incoming or outgoing message with all the Siebel Connector for ACORD XML elements that are necessary to translate it into Siebel data.

The map contains DTE map names, the internal integration object names, the external integration object names, and transaction manager operations. These elements make up the translation scheme for the message instance. The dispatcher map allows the dispatcher to associate the proper translation scheme with each message instance.

All the mapping information is stored in the user property part of the dispatcher map integration object.

FINS ACORD XML Transaction Manager

The FINS ACORD XML Transaction Manager is responsible for data transactions with a Siebel database. It may invoke the Siebel adapter or another business service configured in its user properties. It is an adapter that resides logically between the Siebel object manager and the rest of the connector. It executes operations specified in an XML message instance as Siebel database transactions.

The transaction manager translates XML command elements into Siebel Adapter operations. The transaction manager either carries out the operation or finds another business service to carry out the operation.

The transaction manager combines return results as a single property set. A property set is an intermediate data store that can be used in subsequent operations within the connector.

For inbound processing, the transaction manager accepts an ACORD XML property set, which may contain multiple integration object instances for multiple transactions. It pairs each individual transaction request with an integration object instance and invokes methods in Siebel EAI Adapter.

For outbound processing, the transaction manager pairs a transaction request with an integration object instance and sends an ACORD XML property set to the FINS ACORD XML DTE.

Transaction Manager User Properties

The following table describes the user properties for the FINS ACORD XML Transaction Manager.

Name	Value	Description
IgnoreSvcMethodArgs	true, false	To enable the runtime input arguments, use false.
SaveInFileForRollback	<filename>	The file name in which to save the current record for future rollback.
SaveInMemForRollback	<session key>	The session key to set or look up in memory.
XXX (Operation) such as SAUpsert, SAQuery, and so forth.	ServiceName/MethodName/Argument, such as “EAI Siebel Adapter/Delete/ RollbackOnSame;”	Indicates the operation to be executed. Format is ServiceName/MethodName/Arguments.

Transaction Manager Methods and Arguments

The ACORD XML transaction manager methods and arguments are explained in the following tables. The following table describes the transaction manager methods.

Method	Display Name	Function
Execute	Execute Transaction	Can be used for inbound or outbound messages when the integration object instance is provided. When only Row_Id is available, use the Execute Outbound method.
ExecuteOutbound	Execute Outbound	Executes an outbound operation specified in input arguments.
ExecuteSave	Execute and Save	Executes an outbound operation specified in input arguments and saves the transaction result in memory or in a file.

The Method Arguments for the following table describes the arguments for the FINS ACORD XML Transaction Manager.

Argument	Value	Description
OnlyIOI	true, false	For an inbound message, the integration object instance for request may contain header, body, and envelope portions. When the transaction manager takes the proper operation against the Siebel database, the integration object instance for response is generated as well. If this value is set to “true,” all non-message information from the request message is dropped so the converter and the DTE do not need to deal with, for example, header information. If this value is set to “false,” all request information is carried over.
XMLHierarchy	< Input/Output XML Property Set>	The property set holder for input and output hierarchies.
RollbackInError	true, false	Indicates whether the operation will be rolled back for recovery if an error condition exists.
IXMLMapPath	<Entry key for the dispatcher map>	Stores the key value for the dispatcher map. The transaction manager uses it to look up the value for the integration object instance.
PrimaryRowId	<RowID of a Siebel object>	The primary row ID of the Siebel object that is used by the Siebel adapter to execute a query operation on the Siebel database.
SiebelFINS OperationOut	<Operation name from user property>	The operation to be used by the transaction manager, which is predefined in the user properties of the transaction manager.
SearchSpec	<Search spec>	The search specification for a query that is used by the Siebel adapter to execute a query operation on the Siebel database.
ReportErrorInMsg	true, false	The default value is “false.” If it is set to “true,” the transaction manager generates an error object within the output hierarchy.

The following table provides specifications for the Execute method arguments.

Argument	Display Name	Data Type	Type	Optional
OnlyIOI	Produce only an integration object instance	String	Input	Yes
XMLHierarchy	XML Property Set	Hierarchy	Input or Output	No
RollbackInError	Rollback In Error	String	Input	Yes
ReportErrInMsg	Report Error In Message	String	Input	Yes

The following table provides specifications for the Execute Outbound method arguments.

Name	Display Name	Data Type	Type	Optional
IXMLMapPath	IXML Map Path	String	Input	No
PrimaryRowId	Primary Row Id	String	Input	Yes
SiebelFINS OperationOut	Outbound Operation	String	Input	No
SearchSpec	Search Specification	String	Input	Yes
XMLHierarchy	XML Property Set	Hierarchy	Output	No

The following table provides specifications for the Execute Save method arguments.

Name	Display Name	Data Type	Type	Optional
IXMLMapPath	IXML Map Path	String	Input	No
PrimaryRowId	Primary Row Id	String	Input	Yes
SiebelFINS OperationOut	Outbound Operation	String	Input	No
SearchSpec	Search Specification	String	Input	Yes
RollbackInError	Is Rollback in Error	String	Input	Yes
PlaceToSave	Place To Save	String	Input	No
XMLHierarchy	XML Property Set	Hierarchy	Output	No

FINS ACORD XML Data Transformation Engine (DTE)

The FINS ACORD XML DTE transforms property sets in a Siebel internal integration hierarchy to an external integration object hierarchy, and vice versa. This function allows the FINS ACORD XML Converter to exchange data between two systems with different data models. The transformation map is defined at run time from Siebel Administration views.

For inbound processing, the DTE accepts a property set from the FINS ACORD XML Converter and transforms it into a property set to be used by the FINS ACORD Transaction Manager. The incoming property set is made up of one or more external integration object instances. If there are multiple instances, the DTE parses them into individual instances

and transforms them. The DTE then packages the returned transformed instances as an output property set as internal integration object instances.

For outbound processing, the DTE accepts a property set from the transaction manager and transforms it into a property set to be used by the converter. The outgoing property set is made up of one or more internal integration object instances. The DTE then packages the returned transformed instances as an output property set as external integration object instances.

DTE Methods and Arguments

The FINS ACORD XML DTE methods and arguments are described in the following tables. The following table describes the DTE methods.

Method	Display Name	Function
ToExternal	Transform To External Hierarchy	Transforms a Siebel hierarchy into an external hierarchy.
ToInternal	Transform To Siebel Hierarchy	Transforms an external hierarchy into a Siebel hierarchy.

The following table describes the arguments for the DTE methods.

Argument	Value	Description
XMLHierarchy	XML Property Set	For ToExternal, takes as input the output of the Execute outbound method of the transaction manager. Sends an output hierarchy that contains the XML document in Siebel external integration object format. For ToInternal, takes as input the output of the XMLPropetySetToPropertySet method of the converter. Sends an output hierarchy that contains the ACORD document in Siebel internal integration object format.
MapArgs>		Run-time input arguments that can be used by DTE maps when a map is called from a workflow. See the explanation in the following section.

The following table provides specifications for the ToExternal method argument.

Name	Display Name	Data Type	Type	Optional
XMLHierarchy	XML Property Set	Hierarchy	Input or Output	No

The following table provides specifications for the ToInternal method argument.

Name	Display Name	Data Type	Type	Optional
XMLHierarchy	XML Property Set	Hierarchy	Output	No

Name	Display Name	Data Type	Type	Optional

Using <MapArgs>

<MapArgs> is a runtime input argument used by the DTE map to match an integration map argument of an integration object map. The FINS ACORD XML DTE can take as many <MapArgs> as needed as long as each name is unique among all the <MapArgs> that are passed to the FINS ACORD XML DTE at the same time.

For example, suppose that the output integration object instance has some fields mapping to a workflow process property, such as an ID field.

1. Using the Data Map view, select the integration map to edit in the Integration Object Map applet.
2. In the Integration Map Argument applet, create the map and set the following values:
    a. Name = CompId
    b. Data Type = “DTYPE_TEXT”
    c. Display Name = Component ID
3. In the Integration Field Map applet, set the following values:
    a. Target Field Name = [Id]
    b. Source Expression = [&CompId]
4. In the workflow, set the data transformation engine input argument as follows:
    a. Input Argument = CompId
    b. Type = Process Property
    c. Property Name = Object Id

At runtime, the DTE replaces [&CompId] with the value of the Object ID.

For some mappings, if the DTE cannot find the source field value, the DTE creates empty tags by default. To remove the empty tags, add IgnoreEmptyTag as the map argument.

For complete information, see *Business Processes and Rules: Siebel Enterprise Application Integration* .

FINS ACORD XML Converter

The purpose of the FINS ACORD XML Converter is to generate and process ACORD-specific elements, such as the <SignonRq> aggregate and the <SignonRs> aggregate.

The FINS ACORD XML Converter receives hierarchy output and converts it into a property set or an XML string.

Converter User Properties

The following table describes the FINS ACORD XML Converter user properties.

Name	Value	Description
PI_Parameter:PI_Name	<PI_Value>	PI_Parameter: is a constant prefix. PI_Name=PI_Value would be a PI name-value pair included in ACORD PI. Zero or more pairs can be defined. Examples: • PI_Parameter:newfileuid • PI_Parameter:oldfileuid • PI_Parameter:version
PI_Type	ACORD	Process instruction type
XMLEnvIntObjectName	<Integration object name>	Integration object name that defines the ACORD envelope.

Converter Methods and Arguments

The FINS ACORD XML Converter methods and arguments are described in the following tables. The following table describes the FINS ACORD XML Converter methods.

Method	Display Name	Description
PropSetToXML	PropSetToXML	Generate the XML message to be sent.
PropSetToXMLPropSet	PropSetToXMLPropSet	Prepare the DOM structure of the XML message to be sent.
XMLPropSetToPropSet	XMLPropSetToPropSet	Convert the XML message received into hierarchical property sets.
XMLToPropSet	XMLToPropSet	Prepare the hierarchical property sets from the DOM structure of the XML message received.
ErrorHandler	ErrorHandler	Generate a response ACORD message with detailed error information for an operation error.

The following table describes the arguments common to FINS ACORD XML Converter methods.

Argument Name	Value	Comments
ClientApplicationOrganization	<Client application organization name>	The value for <Org> in <ClientApp>
ClientApplicationName	<Client application name>	The value for <Name> in <ClientApp>

Argument Name	Value	Comments
ClientApplicationVersion	<Client application version>	The value for <Version> in <ClientApp>
IsClient	true, false	“True” if Siebel FINS is behaving as an ACORD client. “False” if Siebel FINS is behaving as an ACORD server.
IsSignoff	true, false	If “true,” the converter generates a SignoffRq aggregate for the current ACORD document. If “false,” the converter does not generate a SignoffRq aggregate.
ServiceProviderName	<ACORD service provider name>	The value will be used as the value of <SPName> in <SignonRq> aggregate.
XMLEnvIntObjectName	<Name of the envelope integration object>	Name of the integration object that defines ACORD envelope.
SignonRsEcho	Property Set storing echo values from <SignonRq>	Those echo values are used by SignonRs header.
initsignon	<Integration component user property of envelope integration object>	Determine which user property in envelope integration object will be used to construct initial SignonRq header.
sessionsignon	<Integration component user property of envelope integration object>	Determine which user property in envelope integration object will be used to construct subsequent SignonRq header.
signonRs	<Integration component user property of envelope integration object>	Determine which user property in envelope integration object will be used to construct SignonRs header.
signoffRq	<Integration component user property of envelope integration object>	Determine which user property in envelope integration object will be used to construct SignoffRq header.
signoffRs	<Integration component user property of envelope integration object>	Determine which user property in envelope integration object will be used to construct SignoffRs header.
ErrorCode	<Error code value>	The error code during operation to be set in the ACORD Status header.
ErrorMessageText	<Error message text>	The actual error message during operation to be set in the ACORD Status header.

Argument Name	Value	Comments
GeneralErrorMessageText	<Error message text>	The generic error text to be pre-appended to the actual error text. For example, “The error happens before the transaction manager.”

The following table provides specifications for the PropSetToXML method arguments.

Name	Display Name	Data Type	Type	Optional
ClientApplicationOrganization	Client Application Organization	String	Input	Yes
ClientApplicationName	Client Application Name	String	Input	Yes
ClientApplicationVersion	Client Application Version	String	Input	Yes
IsClient	Is Client	String	Input	Yes
IsSignoff	Is Signoff	String	Input	Yes
ServiceProviderName	Service Provider Name	String	Input	Yes
XMLHierarchy	XML Property Set	Hierarchy	Input	No
<Value>	XML Document	String	Output	No
XMLEnvIntObjectName	XMLEnvIntObjectName	String	Input	Yes
SignonRsEcho	SignonRsEcho	Hierarchy	Input	Yes
initsignon	initsignon	String	Input	Yes
sessionsignon	sessionsignon	String	Input	Yes
signoffRq	signoffRq	String	Input	Yes
signonRs	signonRs	String	Input	Yes
signoffRs	signoffRs	String	Input	Yes

The following table provides specifications for the PropSetToXMLPropSet method arguments.

Name	Display Name	Data Type	Type	Optional
ClientApplicationOrganization	Client Application Organization	String	Input	Yes
ClientApplicationName	Client Application Name	String	Input	Yes
ClientApplicationVersion	Client Application Version	String	Input	Yes
IsClient	Is Client	String	Input	Yes
IsSignoff	Is Signoff	String	Input	Yes
ServiceProviderName	Service Provider Name	String	Input	Yes
XMLHierarchy	XML Property Set	Hierarchy	Input/output	No
XMLEnvIntObjectName	XMLEnvIntObject Name	String	Input	Yes
SignonRsEcho	SignonRsEcho	Hierarchy	Input	Yes
initsignon	initsignon	String	Input	Yes
sessionsignon	sessionsignon	String	Input	Yes
signoffRq	signoffRq	String	Input	Yes
signonRs	signonRs	String	Input	Yes
signoffRs	signoffRs	String	Input	Yes

The following table provides specifications for the XMLPropSetToPropSet method argument.

Name	Display Name	Data Type	Type	Optional
XMLHierarchy	XML Property Set	Hierarchy	Input/Output	No

The following table provides specifications for the XMLToPropSet method arguments.

Name	Display Name	Data Type	Type	Optional
<Value>	XML Document	String	Input	No
XMLHierarchy	XML Property Set	Hierarchy	Input/Output	No

Name	Display Name	Data Type	Type	Optional

The following table provides specifications for the ErrorHandler method arguments.

Name	Display Name	Data Type	Type	Optional
ErrorCode	ErrorCode	String	Input	No
ErrorMessageText	ErrorMessageText	String	Input	No
GeneralErrorMessageText	GeneralErrorMessageText	String	Input	Yes
XML Hierarchy	ErrXMLHierarchy	Hierarchy	Input/Output	Yes

FINS ACORD XML Dispatcher

The FINS ACORD XML Dispatcher handles inbound XML hierarchy instances. It provides the necessary information for subsequent modules to perform their operations, such as the integration objects to be used.

The dispatcher identifies incoming messages and parses them into header and envelope sections. It also analyzes incoming message body sections, walking through each command. Using the dispatcher map, the dispatcher associates each message with the appropriate external integration object so that the FINS ACORD XML Converter can use it. It also associates the message with the DTE map so that the FINS ACORD XML DTE can use it.

Dispatcher User Properties

The following table shows the user properties for the dispatcher.

Name	Value	Comments
DispatcherMapName	<Integration Object name>	Name of an integration object that details the dispatching rules and syntax for the ACORD XML standard. This map is usually created along with all the other integration objects needed by the wizard. The default map name is ACORDDispMap.
XMLEnvIntObjectName	<Integration Object name>	Name of an integration object that defines the content and hierarchy for the envelope and header sections of the ACORD message.
XMLFaultObject_X, such as XMLFautlObject_0, XMLFaultObject_1, and so forth.	<path to fault section or element>	Location for fault object. For example, //IOI/MsgStatus/MsgErrorCd. This allows the dispatcher to identify a fault section through the path. Extra fault

Name	Value	Comments
		objects can be added by incrementing the name with _1, _2, and so on.

Dispatcher Methods and Arguments

The FINS ACORD XML Converter methods and arguments are described in the following tables. The following table describes the FINS ACORD XML Dispatcher method.

Method	Display Name	Description
DispatchMessage	DispatchMessage	Validates the incoming XML message. If the message conforms to the dispatching rules, the integration object names and other necessary information will be attached to the message. It also checks for the envelope, header, and fault sections of the message and identifies them.

The following table describes the argument for the DispatchMessage method.

Argument	Value	Description
XMLHierarchy	Hierarchy name.	Property set in external integration object XML hierarchy.

The following table provides specifications for the DispatchMessage method argument.

Name	Display Name	Data Type	Type	Optional
XMLHierarchy	XML Hierarchy	Hierarchy	Input/Output	No

Transport Adapter

The transport adapter is a Siebel business service that provides the interface between the outside data source and the Siebel connector. The connector can use any of the following standard transport mechanisms:

- MQSeries
- MQSeries AMI
- HTTP
- MSMQ

For details about the transport adapter, see *Transports and Interfaces: Siebel Enterprise Application Integration* .

4 Configuration Roadmap

Configuration Roadmap

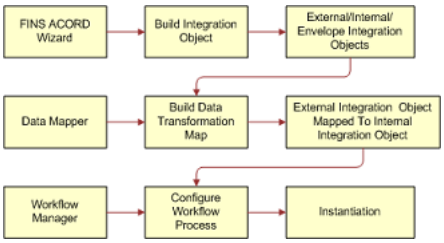
This chapter provides an illustrative example of configuring the ACORD XML connector.

The Siebel Connector for ACORD XML is made up of four pre-built business services:

- FINS ACORD XML Transaction Manager
- FINS ACORD XML Data Transformation Engine
- FINS ACORD XML Converter
- FINS ACORD XML Dispatcher

The Siebel Connector for ACORD XML can be configured to support several types of IFX Business Object Model packages. The following illustrates the main steps in configuring the Siebel Connector for ACORD XML, which are as follows:.

- **FINS ACORD Wizard.** Use FINS IAA Wizard to build an integration object, then external/internal/envelope integration objects, then build a data transformation map, then map external integration object to an internal integration object, then configure the workflow process, then instantiation.
- **Data Mapper.** Use to build a data transformation map, then map external integration object to an internal integration object, then configure the workflow process, then instantiation.
- **Workflow Manager.** Use to configure the workflow process, then instantiation.



This chapter presents the scenario of adding a new Auto Policy through a Siebel front-end application for outbound communication and receiving a request to add an auto policy during inbound communication. This operation corresponds to <PersAutoPolicyAddRq> and <PersAutoPolicyAddRs> commands in ACORD XML. The example uses the MQSeries Server Transport mechanism, though there is no specific transport mechanism required for Siebel Connector for ACORD XML.

The following checklist shows the high-level procedure for configuring your system to use the Siebel Connector for ACORD XML.

Checklist	
•	Create the integration objects in Siebel Tools. For details, see *Creating Integration Objects in Siebel Tools*.
•	Configure the ACORD XML business services in Siebel Tools.

Checklist	
	For details, see *Configuring the Connector Components*.
•	Configure the transformation maps in Siebel Client. For details, see *Configuring the Data Transformation Maps*.
•	For outbound communication, configure the outbound Siebel Connector for ACORD XML for sending an ACORD XML message. For details, see *Configuring an Outbound Siebel Connector for ACORD XML*.
•	For inbound communication, configure the inbound Siebel Connector for ACORD XML for receiving an ACORD XML message. For details, see *Configuring an Inbound Siebel Connector for ACORD XML*.
•	For outbound communication, configure events to trigger the workflow process in real time according to user input using Runtime Event Manager. For details, see *Configuring Runtime Events*.
•	For inbound communication, configure server tasks to dispatch the message to the workflow. For details, see *Configuring Server Tasks*.

Note: When generating integration objects, be certain to use the appropriate version of the ACORD DTD, version 1.0.0. It is available from www.acord.com.

Creating Integration Objects in Siebel Tools

Integration objects define the intermediate format of the data so that it can be used by the connector components to translate between Siebel data formats and ACORD XML data formats.

You use the FINS ACORD Wizard to create the envelope, internal, and external integration objects, as well as the dispatcher map.

The following table shows the pre-setup user properties for the FINS ACORD Wizard.

Name	Value	Comments
DispatcherMapName	ACORDDispMap	The Dispatcher Map Name. The wizard will use this map to update the key and value.
HasUIControl	Y	Internal use.

Name	Value	Comments
Integration Object Wizard	Y	Internal use.
Integration Object Base Object Type	Siebel Business Object	Internal use.
Operation KeyWord Match:0	Add/SAUpsert	Internal use. This means that when the wizard generates an external integration object for an Add message, it defines the operation in the transaction manager as SAUpsert. The operation name will be recorded in the dispatcher map.
Operation KeyWord Match:1	Inq/SAQuery	Internal use. This means that when the wizard generates an external integration object for an Inq message, it defines the operation in the transaction manager as SAUpsert. The operation name will be recorded in the dispatcher map.
Default Envelope Tag	ACORD	Value for Envelope Tag.

Note: You can define a new Operation KeyWord Match:X if you need to. For example, if the ACORD DTD in the future supports the delete operation, you can define Operation KeyWord Match:2 as Delete/SADelete.

Locking the Project and Selecting the DTD File

Whenever you create integration objects, you must lock the project you are working with and associate the appropriate DTD file with the object.

To lock the project and select the DTD file

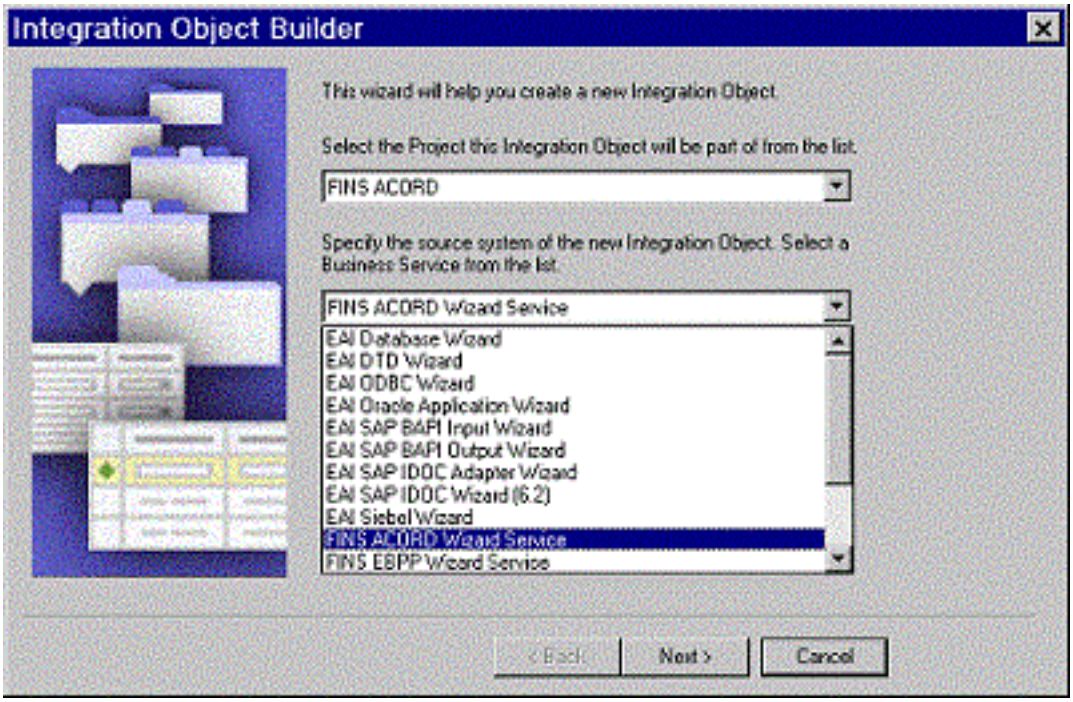
1. Start Siebel Tools.
2. Lock the appropriate project.

   The FINS ACORD XML Wizard requires a locked project.
3. From the application-level menu, choose File, and then New Object.

   The New Object Wizards dialog box appears.
4. Select the EAI tab and double-click Integration Object.

   The Integration Object Builder wizard appears.

5. Fill in the two fields.
   a. Select the project you have locked.
   b. Select FINS ACORD Wizard Service from the Business Service list.



6. Click Next.

   You may receive a warning about a missing dispatcher map.

   This warning offers to write to a temporary dispatcher map, which you can merge with the connector dispatcher map later on.
7. Choose the DTD file you want to use, and then click Next.

   If you are using the file received from the ACORD Web site, the filename is ACORD_PC_XML_V1_0_0.dtd. This filename indicates that this is version 1.00 of ACORD DTD.

   It takes some time for the wizard to parse the DTD file and to display the next page.

Creating an Envelope Integration Object

An envelope integration object provides the envelope and header information needed by the ACORD XML hierarchy.

The following procedure shows how to create an envelope integration object that works with external and internal integration objects. Note that the wizard provides the option of creating an envelope integration object that stands alone, which can be used for authentication.

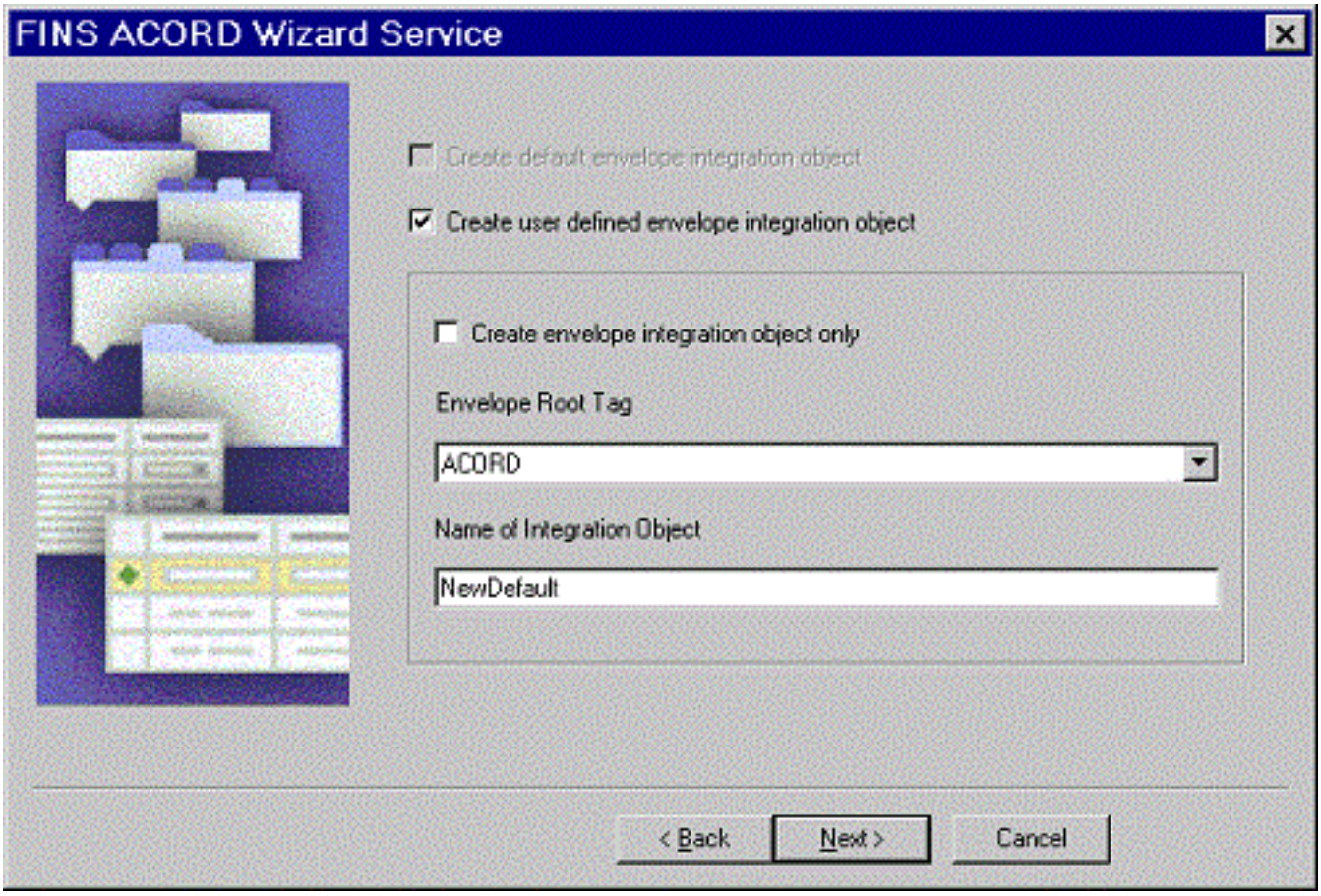
The example provides the steps for creating a user-defined envelope, which allows you to create a customized authentication mechanism. You can also create a default envelope from the wizard. The default envelope provides the user name, password, and session key authentication mechanism that conforms to the ACORD standard.

To create an envelope integration object

1. Lock the project and select the DTD file for the object.

   As described in the previous section, you select the ACORD_PC_XML_V1_0_0.dtd file.

   After the wizard parses the DTD file, it displays the next page so that you can create an envelope integration object.
2. For this example, select the Create user defined envelope check box, fill in the name of the integration object, and click Next.



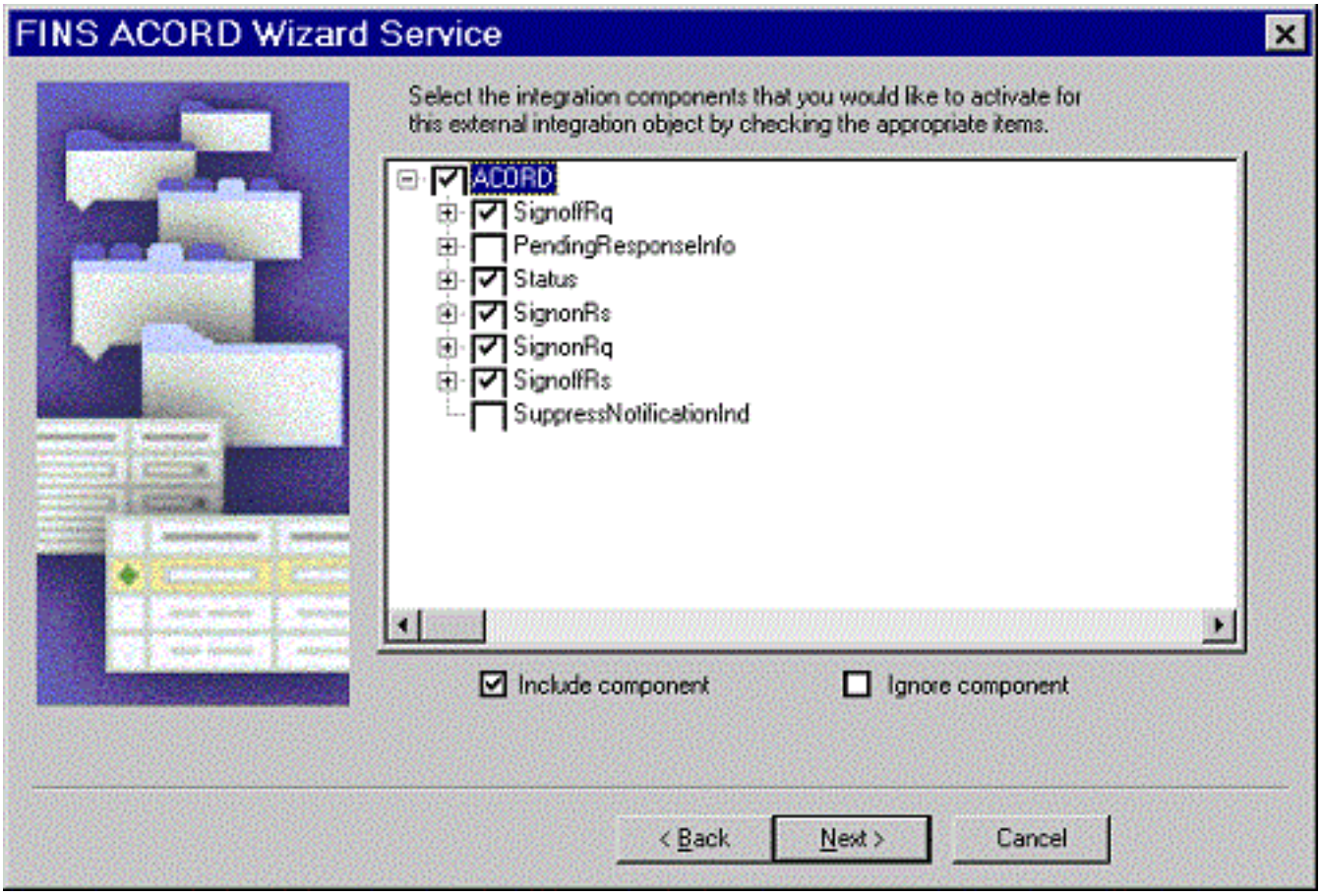
If you wanted to create a standalone envelope, you would select the Create envelope integration object only check box. A standalone envelope integration object can be used by any integration objects later on.

3. Click Next.

   The wizard displays a visual selection hierarchy so that you can select the elements you want to include in the header.

   These elements include <Signon> and <Signoff> elements for both request and response.

   For the example, deselect PendingResponseInfo and SuppressNotification.



4. Verify that the required check boxes are selected, and click Next.

   The wizard displays the first screen in the sequence to create an external integration object.

Creating External Integration Objects

An external integration object establishes the hierarchy for an ACORD XML message. Each type of message under a service needs its own integration object that defines the body portion of the XML document.

When you create an external integration object, you create a pair of such objects, one for the request portion of the cycle and one for the response portion of the cycle.

Each external integration object is paired with an internal integration object when you configure the DTE map.

To create an external integration object

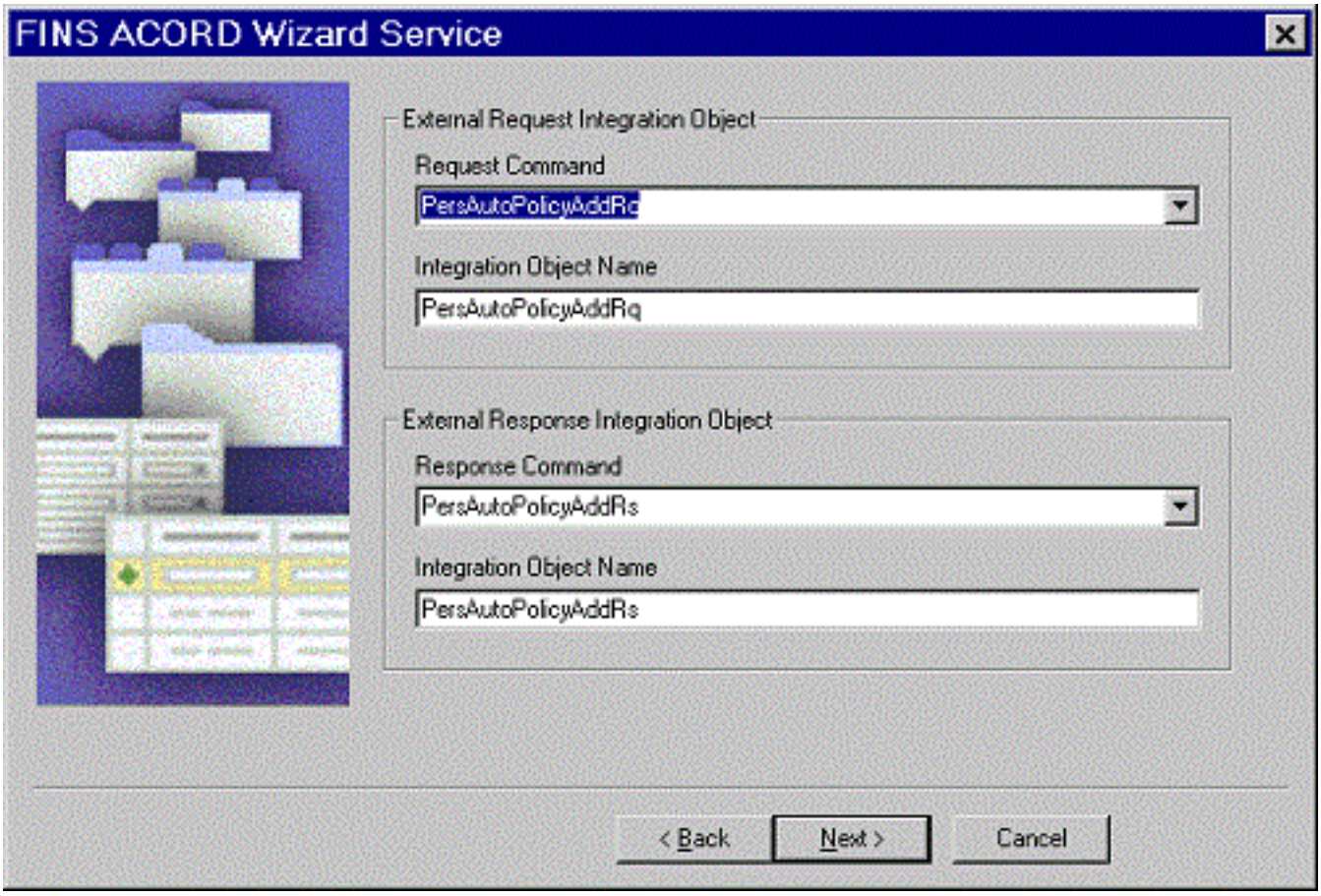
1. Lock the project and select the DTD file for the object.

   You select the ACORD_PC_XML_V1_0_0.dtd file.
2. Create a default or user-defined envelope integration object.

   The steps are described in the preceding section.
3. Choose a Request Service and Response Service pair.

For an external integration object, you need to specify a request service and a response service, for example <InsuranceSvcRq> and <InsuranceSvcRs>. These are the service aggregate elements.

4. Click Next, and then choose the Request Command and Response Command you want to use.



This screen uses Request Command to identify the ACORD request message. The request message you select is automatically paired with an appropriate response message. For this example, you choose <PersAutoPolicyAddRq>; it is automatically paired with <PersAutoPolicyAddRs>.

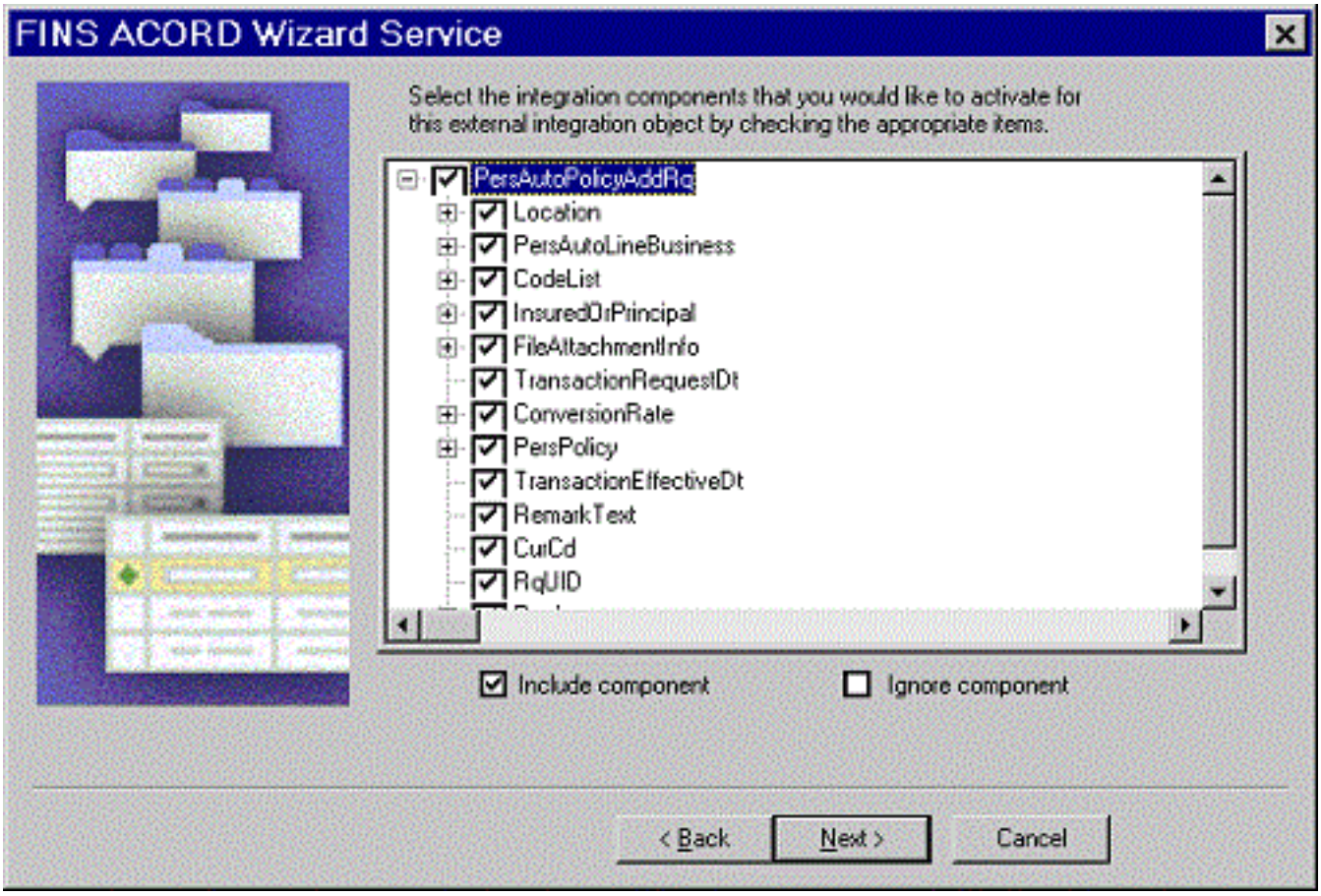
> **Note:** Make a note of the integration object names. You will need to know the names when you configure the DTE map.

You can change the Integration Object Name for the request and response integration objects for administrative convenience. You should consider establishing a set of naming conventions to make groups of objects easy to recognize. This example uses the default names.

5. Click Next to display the integration components screen in which you select the message elements to include.

You will select message elements for the request integration object in this screen, and you will select the message elements for the response integration object in the next screen.

6. Click the plus symbol (+) to display the message elements.



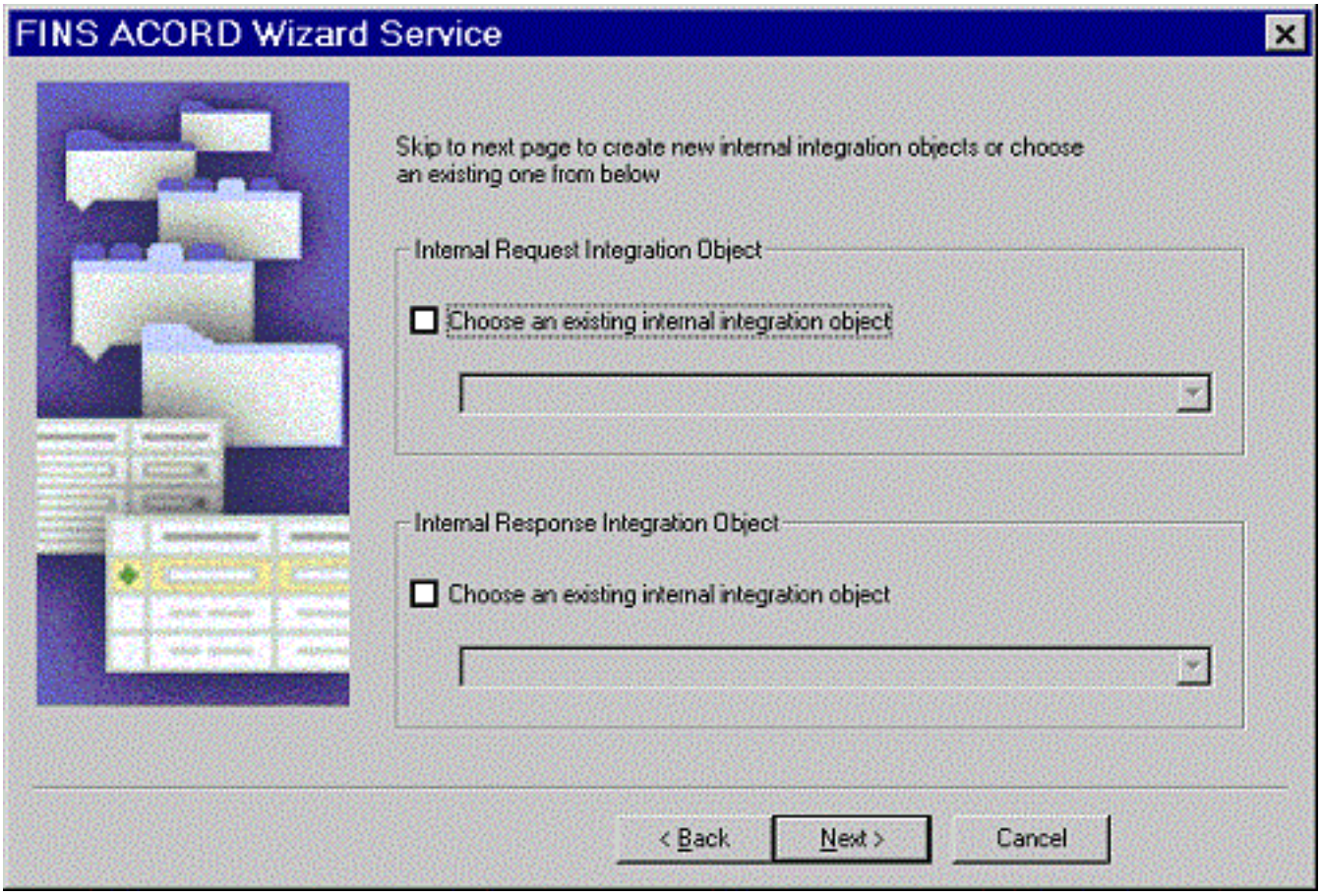
This screen displays a visual hierarchy of the message structure. It provides all the available aggregates and elements for the message. The screen starts with all of these selected (included).

7. Choose the ACORD message components that you want to activate for this integration object.

   Click to select each item. Notice that if you deselect the parent, all the child items are deselected. Reselecting the parent does not reselect the child items, so you can select just a subset of child items.

8. Click Next to display the second integration components screen, and choose the ACORD message components you want to activate for the response integration object.

9. Click Next to display the Select Internal Integration Object screen.



Creating Internal Integration Objects

An internal integration object creates a structure that matches the data structure of a Siebel business object.

You can choose an already-created internal integration object, if one has been created. Siebel integration objects are interfaces for outside systems to interact with internal Siebel data.

This example uses the FINS ACORD Wizard to create a new internal integration object, as shown in the following procedure.

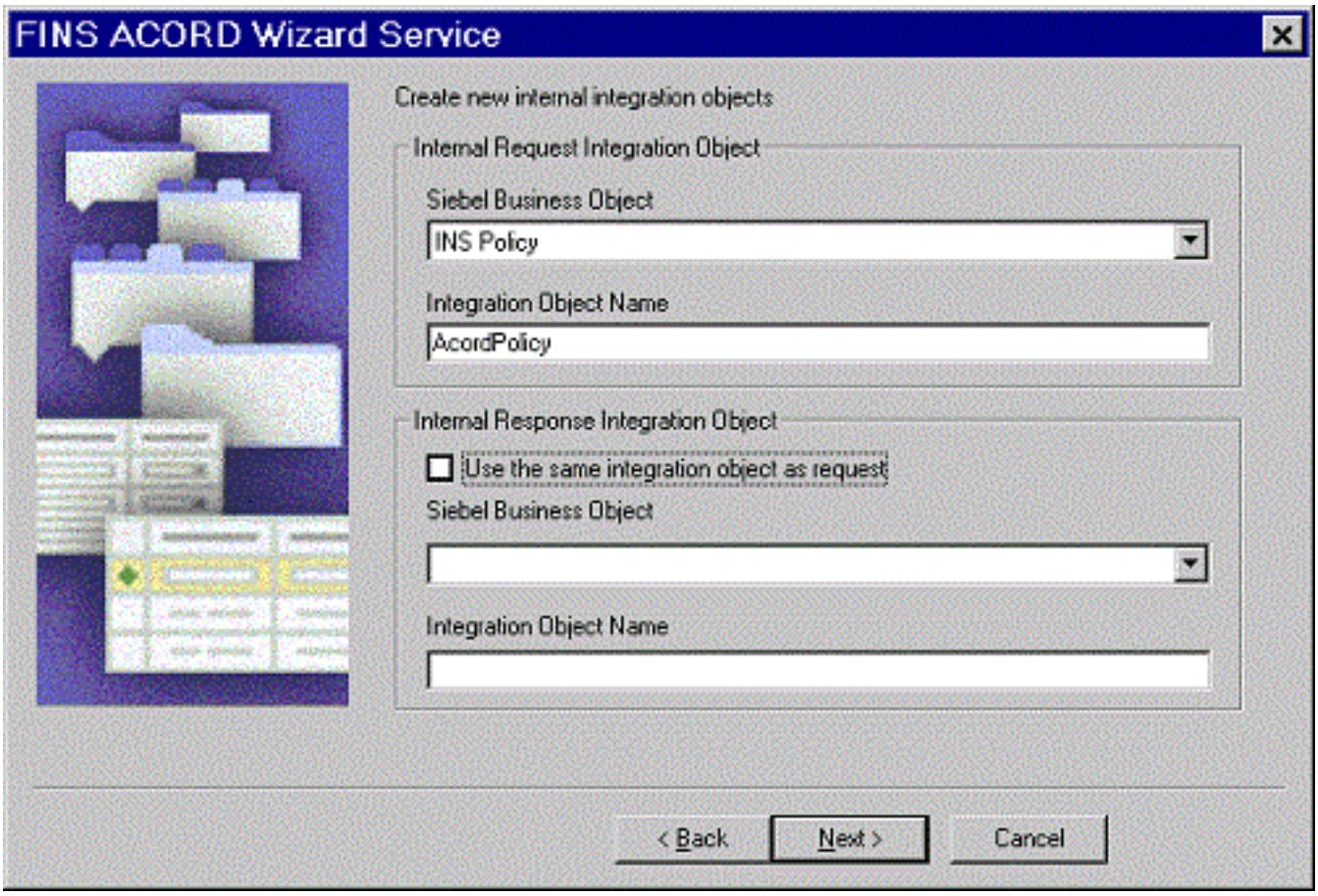
To create an internal integration object

1. Lock the project and select the DTD file for the object.

   You select the ACORD_PC_XML_V1_0_0.dtd file.
2. Create an envelope integration object and a pair of external integration objects.

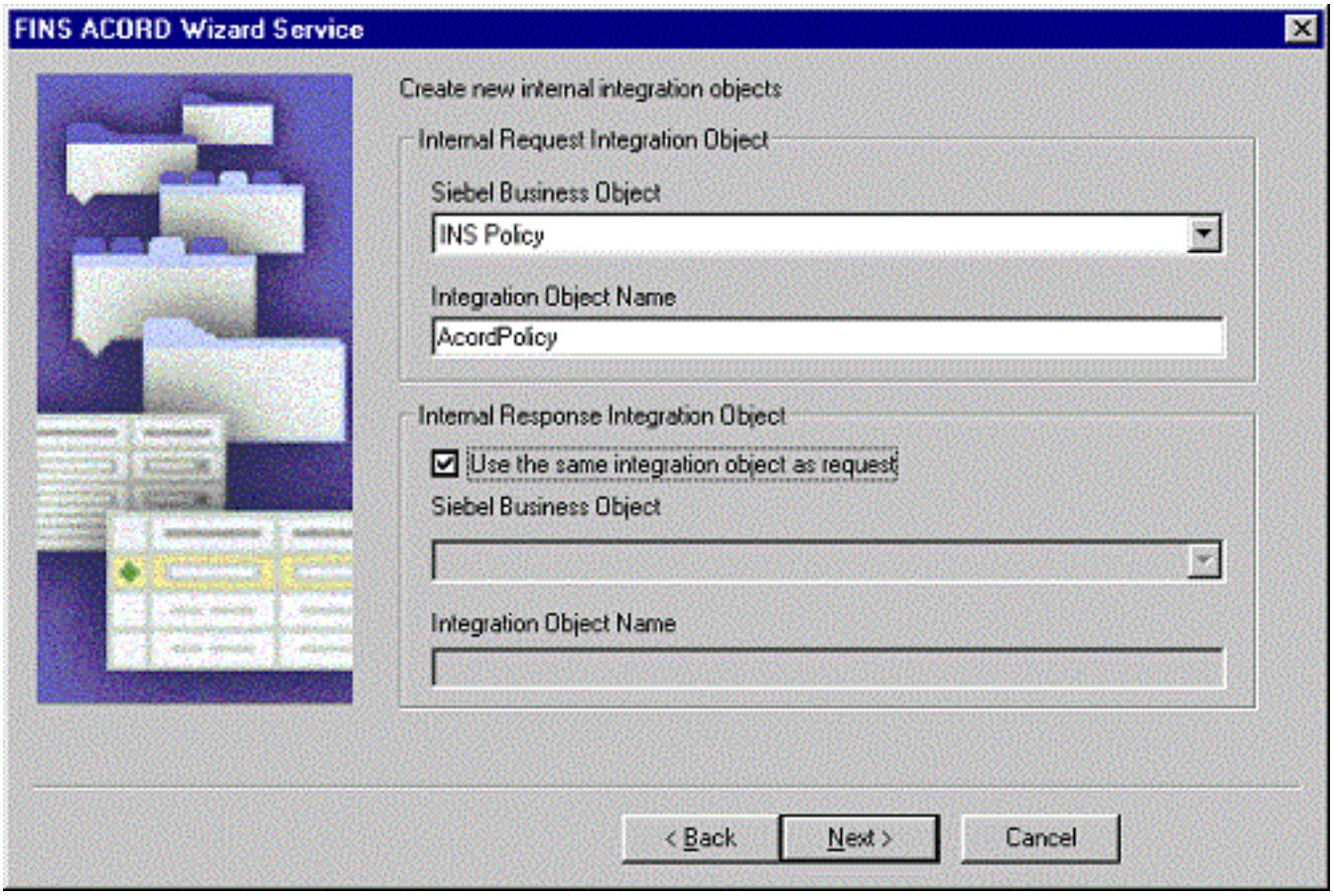
   The steps are described in the preceding sections.

3. In the Select Internal Integration Object screen, do not select either check box, and click Next to display the New Integration Object screen.
4. In the Internal Request Integration Object area, choose the business object that contains the information that the connector will exchange, and enter the name of the integration object.

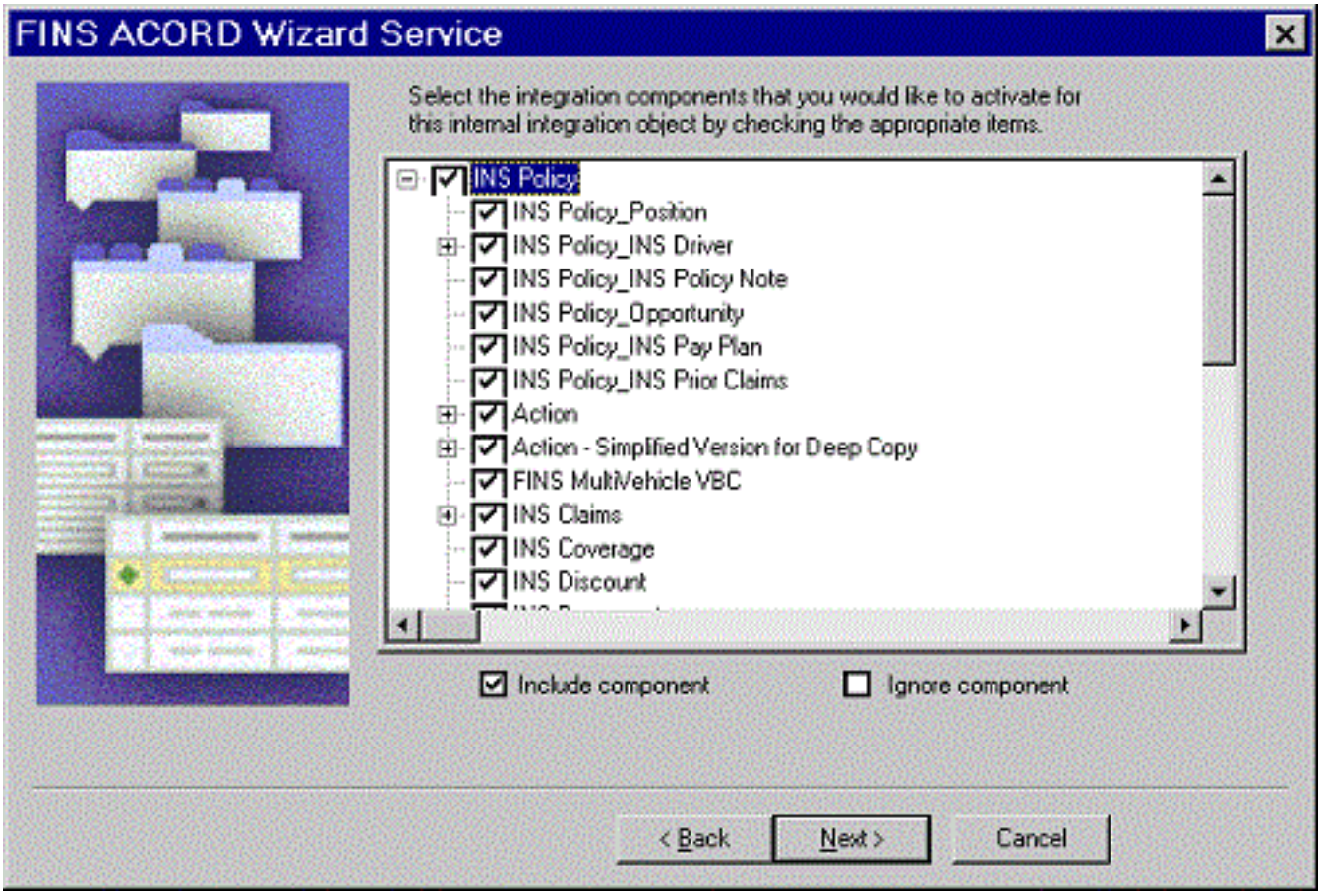


5. For the Internal Response Integration Object, select the Use the same integration object as request check box.

   The entry boxes are not available for selection.



6. Click Next to display the integration components available from the business object you selected on the previous screen.



This screen displays a visual hierarchy of the business object structure. It provides all the available aggregates and elements for the message.

7. Deselect the elements you do not want to include, and then click Next.

The screen displays a warning telling you that it may take some time to create the integration objects.

8. Click Yes to create the integration objects.

After the wizard creates the integration objects, it shows the objects that have been created.

9. Click Finish.

The wizard guides you to the Integration Objects view, which displays a list of integration objects that includes the ones you have just created.

Viewing the Dispatcher Map

When it creates the paired external and internal integration objects, the FINS ACORD Wizard creates a pair of new or updated entries in the dispatcher map.

The FINS ACORD XML dispatcher map is an integration object that contains the rule sets used by the FINS ACORD XML Dispatcher. The default ACORD dispatcher map is ACORDDispMap. If you have the correct project locked, the wizard updates the user properties of the default ACORD dispatcher map. Otherwise, the wizard creates a new ACORD dispatcher map with the following name format and updates its user properties:

ACORDDispMap_<currently locked project name>

To view the Dispatcher Map user properties

1. From Siebel Tools, choose Object Explorer, and then Integration Object.
2. Query for the dispatcher map name, for example ACORDDispMap.
3. Navigate to the user properties of the dispatcher map to see its user properties.

   The following table shows the rule sets created by the wizard for the Add Policy scenario.

Name	Value
`ACORD/InsuranceSvcRq/PersAutoPolicyAddRq`	`ACORD/InsuranceSvcRq/PersAutoPolicyAddRq;PersAutoPolicyAddRq_ERqIRqMapIn;PersAutoPolicyAddRq_IRsERsMapOut;PersAutoPolicyAddRq;AcordPolicy;SAUpsert`
`ACORD/InsuranceSvcRs/PersAutoPolicyAddRs`	`ACORD/InsuranceSvcRs/PersAutoPolicyAddRs;PersAutoPolicyAddRs_ERsIRsMapIn;PersAutoPolicyAddRs_IRqERqMapOut;PersAutoPolicyAddRs;AcordPolicy;SAUpsert`

   The name of the user property represents the rule the dispatcher tries to match and the value represents the value the dispatcher needs to insert. For example, the name:

   `ACORD/InsuranceSvcRq/PersAutoPolicyAddRq`

   is the path the dispatcher uses to locate the message received, and if it finds the match then it uses the information in the value column,

   `ACORD/InsuranceSvcRq/PersAutoPolicyAddRq;PersAutoPolicyAddRq_ERqIRqMapIn;PersAutoPolicyAddRq_IRsERsMapOut;PersAutoPolicyAddRq;AcordPolicy;SAUpsert`

   to determine the action it needs to take.

   The following is a description of the meaning of each of the parts of the information in the value column.

   Each value is made up of six tokens that are separated by semicolons (;), and each token represents specific information.

   - The first token is the location to insert the remaining five tokens at runtime. For example, `ACORD/InsuranceSvcRq/PersAutoPolicyAddRq`.
   - The second token is the name of the data transformation map for mapping the external request integration object indicated by ERq to the internal request integration object indicated by IRq. For example, `PersAutoPolicyAddRq_ERqIRqMapIn`.

- The third token is the name of the data transformation map for mapping the internal response integration object IRs to the external response integration object ERs. For example, `PersAutoPolicyAddRq_IRsERsMapOut`.
- The fourth token is the external request integration object. For example, `PersAutoPolicyAddRq`.
- The fifth token is the internal response integration object. For example, `AcordPolicy`.
- The sixth token is the operation corresponding to the <PersAutoPolicyAddRq> message, for example `SAUpsert`. This token will be used as the key to operation for FINS ACORD XML Transaction Manager.

The data transformation map names must be used when configuring the transformation maps. For details, see *Configuring the Data Transformation Maps*. The map names have to be unique and you need to modify the dispatcher map entries to reflect the new name. The same principle applies to all the tokens.

Note: Deliver the changes to the integration branch.

Configuring the Connector Components

Siebel Connector for ACORD XML provides four prebuilt business services that you can configure for your specific use:

- FINS ACORD XML Transaction Manager
- FINS ACORD XML Data Transformation Engine
- FINS ACORD XML Converter
- FINS ACORD XML Dispatcher

Each business service has its own user properties. The values of these user properties are decided by configuration time. However, you can also override those values in the workflow by entering a run-time value. The meanings of the user properties are described in *Siebel Connector for ACORD XML* This section shows the configuration of the user properties for each of these business services.

FINS ACORD XML Transaction Manager

Several prebuilt operations have been defined in the transaction manager. These operations are sufficient to support most needs in Oracle's Siebel Connector for ACORD XML. It is recommended that you not change these values unless you want to add new operations.

Operation Name	Value
SAQuery	EAI Siebel Adapter/Query/
SARowIdQuery	EAI Siebel Adapter/Query/PrimaryRowId;!SiebelMessage;
SASynchronize	EAI Siebel Adapter/Synchronize/
SAUpsert	EAI Siebel Adapter/Upsert/
SAUpsert_ROLL_BACK	EAI Siebel Adapter/Delete/RollbackOnSame;

Operation Name	Value
SAOperation_FIND	FINS Industry BC Facility- Service/HierarchySearchSpec/! SiebelMessage;IntObjectName=>SiebelFINSRespIntObjName;

The following are examples of how the values in the table are interpreted:

- SAQuery means the operation will execute the EAI Siebel Adapter's Query method.
- SAUpsert means the operation will execute the EAI Siebel Adapter's Upsert method.

The basic format for the value entry is as follows:

- Service/Method/Argument;Argument;
- /Method/Argument;Argument;
- Service, method, and argument are separated by a slash (/).
- Each argument ends with a semicolon (;).
- The default Service name is EAI Siebel Adapter.
- The default argument name is SiebelMessage.

FINS ACORD XML Data Transformation Engine

You do not have to provide any new values in the pre-built business service.

FINS ACORD XML Converter

Please set the user property values according to the following table. These values will appear in the pre-header section of an ACORD message.

Name	Value
XMLEnvIntObjectName	NewDefaultEnv (Just created by the Wizard)
EscapeNames	TRUE
PI_Parameter:version	v1.0.0 (ACORD version)
PI_Parameter:newfileuid	(default empty)
PI_Parameter:oldfileuid	(default empty)
PI_Type	ACORD

FINS ACORD XML Dispatcher

Two user properties need to be set to new values for your specific case, as listed in the following table. In the dispatcher user properties, fill in the names of the dispatcher map and envelope integration objects that are created by FINS ACORD Wizard.

Name	Value
DispatcherMapName	ACORDDispMap
XMLEnvIntObjectName	NewDefaultEnv
XMLFaultObject_0	//IOI/MsgStatus/MsgErrorCd

Configuring the Data Transformation Maps

Configuring the integration objects associates the fields in an internal integration object with the message elements in an external integration object. The result is the creation of the DTE map that will be used by the data transformation engine.

All entries created by the wizard are stored in the Integration Object User Properties of the Dispatcher Map.

In the example, there are four maps that need to be configured to have a complete outbound/inbound transaction route available. Each one can be found in the user properties entry in the ACORDDispMap dispatcher map integration object.

The integration object for the server entry is ACORD/InsuranceSvcRq/PersAutoPolicyAddRq, and it has two maps, as follows:

- PersAutoPolicyAddRq_ErqIRqMapIn (server receiving an inbound request)
- PersAutoPolicyAddRq_IrsERsMapOut (server sending an inbound response)

The integration object for the client is ACORD/InsuranceSvcRs/PersAutoPolicyAddRs, and it has two maps, as follows:

- PersAutoPolicyAddRs_IRqERqMapOut (client sending an outbound request)
- PersAutoPolicyAddRs_ErsIRsMapIn (client receiving an outbound response)

If you wish, you can change the map name in the Dispatcher Map list, then use the new name for the DTE map.

To configure the DTE map

1. Start Siebel Financial Services.
2. Navigate to the Administration - Integration screen, then Data Map Editor, and then Integration Object Map view.

3. In the Integrated Object Map applet, create a new map.
   - **Name.** This name must be the same as the DTE map name created by the wizard and stored in the Dispatcher Map list.
4. Select the Internal Integration Object and the External Integration Object.
   These objects have been created during the process of creating the integration objects with the wizard. Keep the following definitions in mind:
   - **Source Object.** For an message that will be sent out, the source object is the internal integration object; for a message that will be received, the source object is the external integration object.
   - **Target Object.** For a message that will be sent out, the target object is the external integration object; for a message that will be received, the target object is the internal integration object.
5. Map the source components and the target components.
6. Map fields to fields.
   For detailed information, see the following Siebel EAI documents: the chapter on creating and using dispatch rules in *Siebel Financial Services Enterprise Application Integration Guide* , *and the chapter on* data mapping and the data mapper in *Business Processes and Rules: Siebel Enterprise Application Integration* .

Configuring the Workflow Process

The example in this section shows how to create an outbound workflow and an inbound workflow that handle an ACORD XML message.

Configuring an Outbound Siebel Connector for ACORD XML

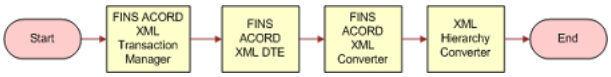
The following procedure shows how to construct a workflow to handle an outbound message.

To create an outbound connector

1. Start Siebel Financial Services.
2. Navigate to the Administration - Business Process screen, then Workflow Processes, and then Workflow Processes view.
3. Create a new record in the Workflow Processes view.
   The keyboard shortcut is CTRL+N.
4. Give the workflow a name.
   The name must be unique within the project. The other values are optional, so for this example, they remain blank.
   In the example, the name is ACORD Add Auto Policy Outbound Workflow. The complete workflow is included in Sample Workflows.
5. Click the Process Designer tab.
   The Process Designer provides a blank working space onto which you will move the step symbols and connectors that create the workflow.
   For complete details about using this working space, and information about workflows in general, see *Siebel Business Process Framework: Workflow Guide* .

6. Move Start, Stop, and Business Service steps onto the work area, and name them appropriately.
7. Use connector arrows to connect the Start and Business Service steps.

   Continue until you have created a workflow with the required components. The following image shows the structure of the finished workflow: FINS ACORD XML Transaction Manager, FINS ACORD XML DTE, FINS ACORD XML Converter, XML Hierarchy Converter.



Adding Process Property Values

You need to create Process Property values for the workflow, to be used in later configurations.

To create process properties

1. Click the Process Properties tab.
2. Enter the values shown in the following table.

Name	Data Type	Default String
ACORD Client Application Name	String	Siebel Financial Services
ACORD Client Application Operation	String	Oracle
ACORD Client Application Version	String	8.0
ACORD DOCTYPE	String	
ACORD DispMap Integration Object Name	String	ACORDDispMap
ACORD Message Full Name	String	ACORD/InsuranceSvcRs/ PersAutoPolicyAddRs
ACORD Service Provider Name	String	Siebel Partner
ACORD Signoff	String	FALSE
Error Code	String	
Error Message	String	
Object Id	String	
PropSet Converter Out	Hierarchy	
PropSet DTE Out	Hierarchy	

Name	Data Type	Default String
PropSet TransMgr Out	Hierarchy	
SiebelFINSOperation Out	String	SARowIdQuery
XML Document	String	
initsignon	String	initsignon
sessionsignon	String	sessionsignon
signoffRq	String	signoffRq

Configuring the FINS ACORD XML Transaction Manager

Now that the workflow structure exists, you can configure the components. The first outbound component is the transaction manager.

To configure the transaction manager

1. Name the transaction manager appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML Transaction Manager.
3. Choose the method.

   The example uses the Execute Outbound method because the data is being sent out from the Siebel data source.
4. Establish the input and output arguments.

   As for all Siebel business services, create a new record (CTRL+N) for a new argument. Select from the available input arguments. Add all required arguments first, then go on to any optional arguments. See *Siebel Connector for ACORD XML* for input and output specifications.

   The following are the input argument settings for the example transaction manager configuration.

Input Arguments	Type	Value	Property Name	Property Data Type
DispatcherMapName	Process Property		ACORD DispMap Integration Object Name	String
IXMLMapPath	Process Property		ACORD Message Full Name	String

Input Arguments	Type	Value	Property Name	Property Data Type
PrimaryRowId	Process Property		Object Id	String
SiebelFINSOperationOut	Process Property		SiebelFINSOperationOut	String

The following are the output argument settings for the example transaction manager configuration.

Property Name	Type	Value	Output Argument
PropSet TransMgr Out	Output Argument		XML Hierarchy

Configuring the FINS ACORD XML Data Transformation Engine

The second component is the data transformation engine (DTE).

To configure the DTE

1. Name the DTE appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML DTE.
3. Choose the method.

   The example uses the Transform to External Hierarchy method because the data is moving from a Siebel internal system to an external system.
4. Set the input and output arguments.

   These arguments include the DTE map name, created during the configuration of the internal and external integration objects.

   The following are the input argument settings for the example DTE configuration.

Input Arguments	Type	Value	Property Name	Property Data Type
XML Property Set	Process Property		PropSet TransMgr Out	Hierarchy

Input Arguments	Type	Value	Property Name	Property Data Type

The following are the output argument settings for the example DTE configuration.

Property Name	Type	Value	Output Argument
PropSet DTE Out	Output Argument		XML Property Set

Configuring the FINS ACORD XML Converter

The third component is the converter.

To configure the converter

1. Name the converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.
   For this component, choose the FINS ACORD XML Converter.
3. Choose the method.
   The example uses PropSetToXMLPropSet because the converter is converting a property set from the DTE into a standard XML property set.
4. Set the input and output arguments.
   The following are the input argument settings for the example converter configuration.

Input Argument	Type	Value	Property Name	Property Data Type
Client Application Name	Process Property		ACORD Client Application Name	String
Client Application Operation	Process Property		ACORD Client Application Operation	String
Client Application Version	Process Property		ACORD Client Application Version	String
Is Signoff	Process Property		ACORD Signoff	String
Service Provider Name	Process Property		ACORD Service Provider Name	String
XML Property Set	Process Property		PropSet DTE Out	Hierarchy

Input Argument	Type	Value	Property Name	Property Data Type
initsignon	Process Property		initsignon	String
sessionsignon	Process Property		sessionsignon	String
signoffRq	Process Property		signoffRq	String

The following are the output argument settings for the example converter configuration.

Property Name	Type	Value	Output Argument
PropSet Converter Out	Output Argument		XML Property Set

Configuring an XML Hierarchy Converter

The fourth component is a hierarchy converter. For information about Siebel hierarchy converters, see *XML Reference: Siebel Enterprise Application Integration* .

To configure an XML hierarchy converter

1. Name the hierarchy converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the XML Hierarchy Converter.
3. Choose the method.

   This example uses the XML Hierarchy to XML Document method because, for an outgoing message, the final conversion is from an XML hierarchy to an XML document that can be accepted by any XML-compliant converter at the external location.
4. Set the input and output arguments, XML Hierarchy name, and XML document name.

   The following are the input argument settings for the example hierarchy converter configuration.

Input Argument	Type	Value	Property Name	Property Data Type
XML Header Text	Process Property		ACORD DOCTYPE	String
XML Hierarchy	Process Property		PropSet Converter Out	Hierarchy

The following are the output argument settings for the example hierarchy converter configuration.

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

Configuring a Transport Mechanism (Optional)

The fifth component is a transport mechanism. Although not shown on the diagram, this example uses the MQSeries transport. For information about transport mechanisms supported by Oracle, see *Transports and Interfaces: Siebel Enterprise Application Integration* .

To configure an MQSeries transport

1. Name the transport component appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the EAI MQSeries Server Transport.
3. Choose the method.

   This example uses the Send method.
4. Set the input and output arguments, including the Physical Queue Name, the Queue Manager Name, and the Message Text.

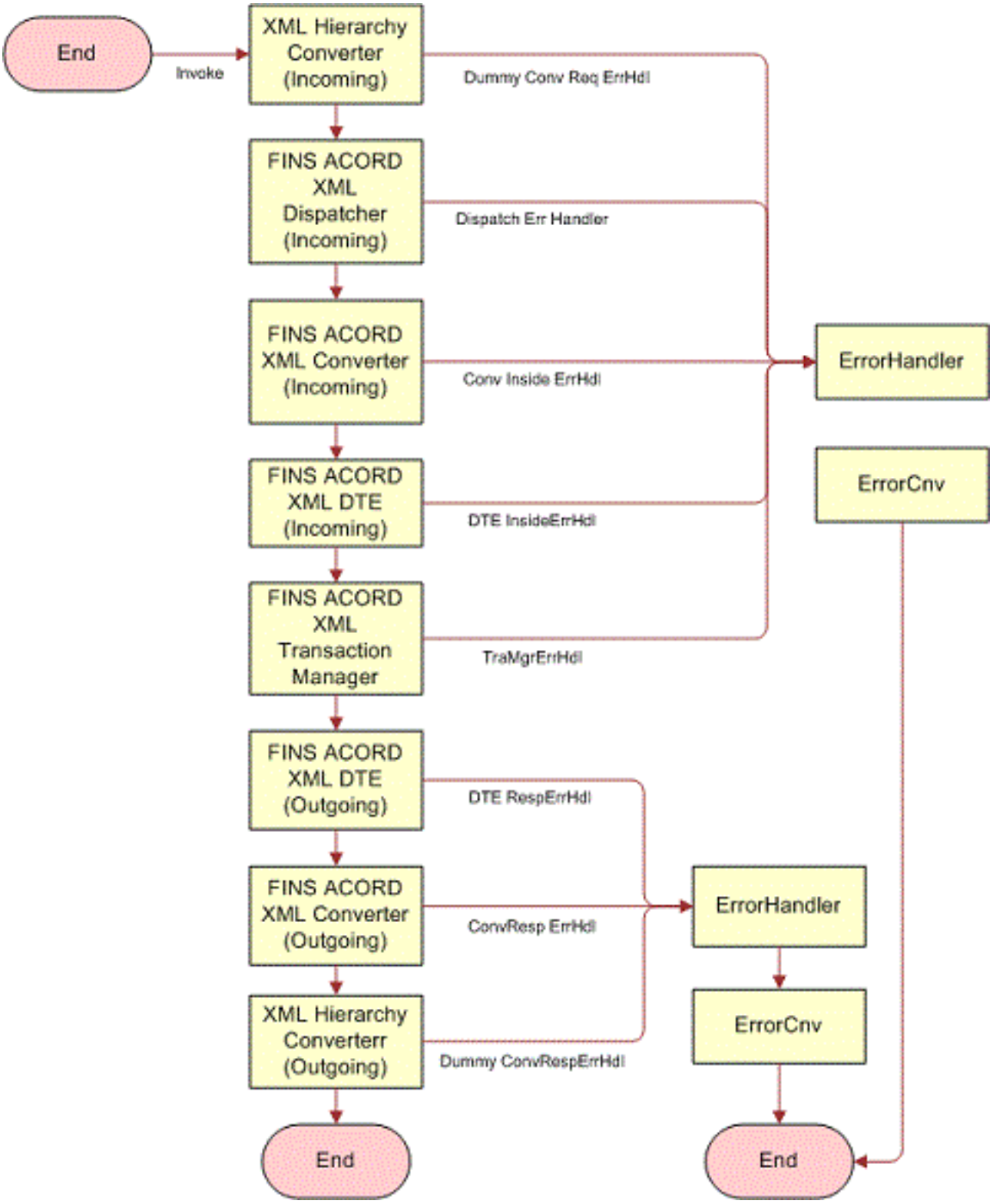
   The following are the input argument settings for the example MQSeries transport configuration.

Input Argument	Type	Value	Property Name	Property Data Type
Message Text	Process Property		XML Document	String
Physical Queue Name	Literal	TO_ACORD_QUEUE		
Queue Manager Name	Literal	SHAN		

   This step completes the outbound workflow. The outbound ACORD XML connector can be used as the basis for any workflow that is used to send an outbound request message.

Configuring an Inbound Siebel Connector for ACORD XML

For an inbound message, one that originates in an external application, the following shows the structure of a workflow that receives a request and sends back a response. The example workflow is the ACORD Server Policy Package Workflow included in the sample workflows.


End
Invoke
XML Hierarchy Converter (Incoming)
Dummy Conv Req ErrHdl
FINS ACORD XML Dispatcher (Incoming)
Dispatch Err Handler
FINS ACORD XML Converter (Incoming)
Conv Inside ErrHdl
ErrorHandler
FINS ACORD XML DTE (Incoming)
DTE InsideErrHdl
ErrorCnv
FINS ACORD XML Transaction Manager
TraMgrErrHdl
FINS ACORD XML DTE (Outgoing)
DTE RespErrHdl
FINS ACORD XML Converter (Outgoing)
ConvResp ErrHdl
ErrorHandler
XML Hierarchy Converterr (Outgoing)
Dummy ConvRespErrHdl
ErrorCnv
End
End

The steps in this workflow are as follows:

1. Convert XML Hierarchy (Dummy Conv Req ErrHdl)
    - FINS ACORD XML Dispatcher (Incoming). (Dispatch Err Handler)
    - FINS ACORD XML Converter (Incoming) (Conv Inside ErrHdl)
    - FINS ACORD XML DTE (Incoming) (DTE InsideErrHdl)
    - FINS ACORD XML Transaction Manager (TraMgrErrHdl)
2. FINS ACORD XML DTE (Outgoing) (DTE RespErrHdl)
    - FINS ACORD XML Converter (Outgoing) (ConvResp ErrHdl)
    - XML Hierarchy Converter (Outgoing) Dummy ConvRespErrHdl
3. ErrorHandler
4. ErrorCnv

Note: The error handling conditions and business services (for example ErrorHandler and ErrorCnv) on the side of the diagram are necessary to handle any rare processing errors. The sample workflow provides information about how they are configured.

Adding Process Property Values

You need to create Process Property values for the workflow, to be used in later configurations.

To create process properties

1. Click the Process Properties tab.
2. Enter the values shown in the following image.

Name	Data Type	Default String
<Value>	String	<Value>
ACORD Client Application Name	String	Siebel Financial Services
ACORD Client Application Operation	String	Oracle
ACORD Client Application Version	String	8.0
ACORD DOCTYPE	String	
ACORD DispMap Integration Object Name	String	ACORDDispMap
ACORD Envelope Integration Object Name	String	Newdefault
ACORD Service Provider Name	String	Siebel Partner

Name	Data Type	Default String
ACORD Signoff	String	FALSE
Error Code	String	
Error Message	String	
IsClient	String	FALSE
Object Id	String	
PropSet Converter Out (Incoming)	Hierarchy	
PropSet Converter Out (Outgoing)	Hierarchy	
PropSet DTE Out (Incoming)	Hierarchy	
PropSet DTE Out (Outgoing)	Hierarchy	
PropSet Dispatcher Out (Incoming)	Hierarchy	
PropSet TransMgr Out (Outgoing)	Hierarchy	
PropSet XML Converter Out (Incoming)	Hierarchy	
SiebelFINSOperation	String	
SignonRsEcho	Hierarchy	
XML Document Response	String	

Configuring an XML Hierarchy Converter (Incoming)

The first connector component in the inbound sequence is a hierarchy converter. For information about Siebel hierarchy converters, see *XML Reference: Siebel Enterprise Application Integration* .

To configure an XML hierarchy converter

1. Name the hierarchy converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the XML Hierarchy Converter.
3. Choose the method.

   This example uses the XML Document to XML Hierarchy method because, for an incoming message, the conversion is from an XML document to an XML hierarchy that can be used by the FINS ACORD XML Dispatcher.
4. Set the input and output arguments, XML Hierarchy name, and XML document name.

   The following are the input argument settings for the example hierarchy converter incoming configuration.

Input Argument	Type	Value	Property Name	Property Data Type
XML Document	Process Property		<Value>	String

   The following are the output argument settings for the example hierarchy converter incoming configuration.

Property Name	Type	Value	Output Argument
PropSet XML Converter Out (Incoming)	Output Argument		XML Hierarchy

Configuring the FINS ACORD XML Dispatcher (Incoming)

The second connector component is the dispatcher.

To configure the dispatcher

1. Name the dispatcher appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML Dispatcher.
3. Choose the method.

   The example uses the Dispatch Message method.
4. Set the input and output arguments.

   The following are the input argument settings for the example dispatcher incoming configuration.

Input Argument	Type	Value	Property Name	Property Data Type
DispatcherMapName	Process Property		ACORD DispMap Integration Object Name	String

Input Argument	Type	Value	Property Name	Property Data Type
XMLEnvIntObjectName	Process Property		ACORD Envelope Integration Object Name	String
XML Property Set	Process Property		PropSet XML converter Out (Incoming)	Hierarchy

The following are the output argument settings for the example dispatcher incoming configuration.

Property Name	Type	Value	Output Argument
PropSet Dispatcher Out (Incoming)	Output Argument		XML Property Set

Configuring the FINS ACORD XML Converter (Incoming)

The third connector component is the converter.

To configure the converter

1. Name the converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML Converter.
3. Choose the method.

   The example uses XMLPropSetToPropSet, because the converter is converting an XML property set from the dispatcher into a Siebel-hierarchy-based property set.
4. Set the input and output arguments.

   The following are the input argument settings for the example converter incoming configuration.

Input Argument	Type	Value	Property Name	Property Data Type
IsClient	Process Property		IsClient	String
XMLEnvIntObjectName	Process Property		ACORD Envelope Integration Object name	String
XML Property Set	Process Property		PropSet Dispatcher Out (Incoming)	Hierarchy

Input Argument	Type	Value	Property Name	Property Data Type

The following are the output argument settings for the example converter incoming configuration.

Property Name	Type	Value	Output Argument
PropSet Converter Out (Incoming)	Output Argument		XML Property Set
SignonRsEcho	Output Argument		SignonRsEcho

Configuring the FINS ACORD XML Data Transformation Engine (Incoming)

The fourth connector component is the data transformation engine (DTE).

To configure the DTE

1. Name the DTE appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML DTE.
3. Choose the method.

   The example uses the Transform to Siebel Hierarchy method because the data is moving from an external system to a Siebel internal system.
4. Set the input and output arguments.

   The following are the input argument settings for the example DTE incoming configuration.

Input Argument	Type	Value	Property Name	Property Data Type
XML Property Set	Process Property		PropSet Converter Out (Incoming)	Hierarchy

The following are the output argument settings for the example DTE incoming configuration.

Property Name	Type	Value	Output Argument
PropSet DTE Out (Incoming)	Output Argument		XML Property Set

Configuring the FINS ACORD Transaction Manager

The fifth connector component is the transaction manager.

To configure the transaction manager

1. Name the transaction manager appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML Transaction Manager.
3. Choose the method.

   The example uses the Execute Transaction method because the data is being received from the external data source and must be delivered to the Siebel application.
4. Establish the input and output arguments.

   The following are the input argument settings for the example transaction manager configuration. Notice that the data flow within the connector changes direction at this point from incoming to outgoing.

Input Argument	Type	Value	Property Name	Property Data Type
Only product Integration Object Instance	Literal	true		
Report Error in Message	Literal	true		
Rollback In Error	Literal	false		
SiebelFINSOperationOut	Process Property		Siebel FINS Operation	String
Status Object	Literal	true		
XML Property Set	Process Property		PropSet DTE Out (Incoming)	Hierarchy

   The following are the output argument settings for the example transaction manager configuration.

Property Name	Type	Value	Output Argument
PropSet TransMgr Out (Outgoing)	Output Argument		XML Property Set

Configuring the FINS ACORD XML Data Transformation Engine (Outgoing)

The sixth connector component is the data transformation engine (DTE).

To configure the DTE

1. Name the DTE appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML DTE.
3. Choose the method.

   The example uses the Transform to External Hierarchy method because the data is moving from the Siebel internal system to an external system.
4. Set the input and output arguments.

   The following are the input argument settings for the example DTE outgoing configuration.

Input Argument	Type	Value	Property Name	Property Data Type
XML Property Set	Process Property		PropSet TransMgr Out (Outgoing)	Hierarchy

   The following are the output argument settings for the example DTE outgoing configuration.

Property Name	Type	Value	Output Argument
PropSet DTE Out (Outgoing)	Output Argument		XML Property Set

Configuring the FINS ACORD XML Converter (Outgoing)

The seventh connector component is the converter.

To configure the converter

1. Name the converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the FINS ACORD XML Converter.
3. Choose the method.

   The example uses PropSetToXMLPropSet because the converter is converting a property set from the DTE into a standard XML property set.

4. Set the input and output arguments.

The following are the input argument settings for the example converter outgoing configuration.

Input Argument	Type	Value	Property Name	Property Data Type
Client Application Name	Process Property		ACORD Client Application Name	String
Client Application Organization	Process Property		ACORD Client Application Organization	String
Client Application Version	Process Property		ACORD Client Application Version	String
Is Client	Process Property		IsClient	String
Is Signoff	Process Property		ACORD Signoff	String
Service Provider Name	Process Property		ACORD Service Provider Name	String
SignonRsEcho	Process Property		SignonRsEcho	Hierarchy
XMLEnvIntObjectName	Process Property		ACORD Envelope Integration Object	String
XML Property Set	Process Property		PropSet DTE Out (Outgoing)	Hierarchy

The following are the output argument settings for the example converter outgoing configuration.

Property Name	Type	Value	Output Argument
PropSet Converter Out (Outgoing)	Output Argument		XML Property Set

Configuring an XML Hierarchy Converter (Outgoing)

The eighth connector component is a hierarchy converter. For information about Siebel hierarchy converters, see *XML Reference: Siebel Enterprise Application Integration* .

To configure an XML hierarchy converter

1. Name the hierarchy converter appropriately (it must be unique within the workflow).
2. Choose the type of business service from the multi-value group (MVG) window.

   For this component, choose the XML Hierarchy Converter.
3. Choose the method.

   This example uses the XML Hierarchy to XML Document method because, for an outgoing message, the final conversion is from an XML hierarchy to an XML document that can be accepted by any XML-compliant converter at the external location.
4. Set the input and output arguments, XML Hierarchy name, and XML document name.

   The following are the input argument settings for the example hierarchy converter outgoing configuration.

Input Argument	Type	Value	Property Name	Property Data Type
XML Header Text	Process Property		ACORD DOCTYPE	String
XML Hierarchy	Process Property		PropSet Converter Out (Outgoing)	Hierarchy

   The following are the output argument settings for the example hierarchy converter outgoing configuration.

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

   This step completes the inbound workflow. The inbound ACORD XML connector can be used as the basis for any workflow that is used to receive an inbound request message.

Configuring Runtime Events

Oracle supports triggering workflows based on runtime events. Using a runtime event allows you to incorporate configured workflow functions into actual applications. For complete information about runtime events, see *Siebel Events Management Guide* .

The example in this section describes a runtime event that adds an auto policy when the Sub Status field changes in an INS Policy business component record. In the following procedure, a runtime event is defined and linked to an action that starts the example outbound workflow. It sends the active record as an ACORD XML message in the example connector.

To create a runtime event

1. Start the Siebel client.
2. Navigate to the Administration - Runtime Events screen, then Events, and then Events view.
3. Create a new event record, and enter the following values:
    - Action Set Name = ACORDDemo
    - Object Type = BusComp
    - Object Name = INS Policy
    - Event = SetFieldValue
    - Subevent = Sub Status
4. Enter the following condition expression:

    ([Sub Status] = LookupValue ('INS_POLICY_SUBSTATUS', 'Submitted'))

    This expression instructs the Siebel application to initiate the action set ACORDDemo when the Sub Status field value changes to Submitted. Typically, this will occur when the user selects Submitted from the value list.
5. Create a new record for Action Sets/Actions, with the following parameters:
    - Actions Type = BusService
    - Business Service Name = Workflow Process Manager
    - Business Service Method = RunProcess
    - Business Service Context = "ProcessName," ACORD Add Auto Policy Outbound Workflow (the name of the configured workflow that handles outbound messages).

Configuring Server Tasks

A server task is an instantiation of a server component. To run a server task, you need to run a component request, which will request that one or more server tasks run.

The following example describes how to configure a server task for the EAI MQSeries Server Transport. The Siebel EAI MQSeries Server Transport is designed to provide a messaging solution to help you integrate data between Oracle's Siebel Business Applications and external applications that can interface with the IBM MQSeries. The IBM MQSeries Server Transport transports messages to and from IBM MQSeries queues.

In order to run this server task successfully you need to first configure two named subsystems.

To configure named subsystems

1. Start the Siebel client.
2. Navigate to the Administration - Server Configuration screen, then Profile Configuration, and then Profile Configuration view.

3. Create a new record in the Component Profiles list and provide the required information.
    - **Name.** Name of the named subsystem, for example ACORDMQConnSubsys
    - **Type.** Type of the named subsystem, MQSeriesServerSubsys

    Note: The subsystem type that you select should have a checkmark in the Is Named Enabled field.
4. Save the record.
5. In the Enterprise Profile Configuration list, modify parameters.
    - **MqPhysicalQueueName.** <Queue name from which to receive inbound request message>
    - **MqQueueManagerName.** <Name of the queue manager who owns the queues>
    - **MqRespPhysicalQueueName.** <Queue name to which to send the response message>
    - **MqSleepTime.** 100 <or longer if needed>
6. Save the record.
7. Create another Named Subsystem with the following name and parameters:
    - **Name.** Name of the named subsystem, for example ACORDMQDataSubsys
    - **Type.** EAITransportDataHandlingSubsys
8. Save the record.
9. In the Enterprise Profile Configuration list, modify the parameters.
    - **DispatchWorkflowProcess.** ACORD Server Party Package Workflow

After creating and configuring the named subsystems, you need to configure MQSeries Receiver.

To configure MQSeries Receiver

1. Start the Siebel client.
2. Navigate to the Administration - Server Configuration screen, then Enterprises, and then Component Definitions view.
3. Query for MQSeries Server Receiver and set the following:
    - **Receiver Connection Subsystem.** ACORDMQConnSubsys <the name from step 4 in the previous procedure>
    - **Receiver Data Handling Subsystem.** ACORDMQDataSubsys <the name from step 8 in the previous procedure>
    - **Receiver Method Name.** ReceiveDispatch <or ReceiveDispatchSend>
    - **Default Tasks.** 1 <or number of tasks desired>
4. Restart the Siebel server and make sure that the MQSeries Server Receiver server component is running.

Note: For details on creating and configuring server tasks, see *Siebel System Administration Guide* and for details on configuring MQSeries, see *Transports and Interfaces: Siebel Enterprise Application Integration* .

Sample Workflows

This chapter has provided an example configuration roadmap. To demonstrate the entire process, the ACORD Policy Package is included in the sample database for Oracle's Siebel Financial Services. The package contains all the workflows, integration objects and DTE Maps needed. The package supports the following functions:

- Sending out standard ACORD request messages from a Siebel client application to add (Add) or query (Inquiry) ACORD-compatible insurance policy records.
- The messages can be sent from the Auto Policies view, the Property Policies view, and the PUL Policies view.
- For a Siebel server receiving an ACORD request message (from step 1), executing the proper database transaction and sending back a standard ACORD response message.
- For a Siebel client application receiving an ACORD response message (from step 2), executing the proper database transaction.

The package includes the following integration objects:

- Data Transformation Maps (DTE maps)
- Including 24 Maps for Siebel ACORD Policy Messages

The package includes several workflows to minimize your custom configuration tasks. These workflows can be used as templates to customize or configure to become your own workflows.

Workflows are categorized as follows:

Clients: Ready for use

- ACORD Client Auto Policy Package Workflow (comprehensive)
  - ACORD Add Auto Policy Outbound Workflow
  - ACORD Inquiry Auto Policy Outbound Workflow
- ACORD Client Home Policy Package Workflow (comprehensive)
  - ACORD Add Home Policy Outbound Workflow
  - ACORD Inquiry Home Policy Outbound Workflow
- ACORD Client PUL Policy Package Workflow (comprehensive)
  - ACORD Add PUL Policy Outbound Workflow
  - ACORD Inquiry PUL Policy Outbound Workflow
- ACORD Policy Outbound Response Workflow No Rollback
- ACORD Policy Outbound Response Workflow With Rollback

Server: Ready for use

- ACORD Server Policy Package Workflow

5 Data Types

Data Types

This chapter describes the ACORD XML data types.

ACORD XML Data Types

The following data types are used to represent all data passed between clients and servers using the messages defined in the ACORD specification. All information elements are based on these data types. Supported data types are discussed in the sections that follow.

Character

Character indicates an element that allows character data up to a maximum number of characters, regardless of the number of bytes required to represent each character. The number after the hyphen specifies the maximum number of characters. For example, C-12 specifies an element of characters with maximum length 12 characters. C-Infinite indicates an element with no maximum length.

> **Note:** Depending on the character encoding, each character may be represented by one or more bytes. For example, UTF-8 uses two bytes to encode characters represented in ISO Latin-1 as single bytes in the range 128 through 255 decimal. In addition, characters may be encoded as multi-byte *character entities,* and XML encodes the ampersand, less-than symbol, and greater-than symbol as “&” “<” and “>” when they appear as document content. Therefore, an element of type C-40 may be represented by more than 40 bytes in a UTF-8 encoded XML stream. For example, the string “AT&T” encodes “AT&T”.

Narrow Character

Elements of type *Narrow Character* are elements of *character* data type with the additional restriction that the only allowable characters are those contained within the ISO Latin-1 character set.

Boolean

The *Boolean* data type has two states, true and false. True is represented by the literal character 1 (one), while false is represented by the literal character 0 (zero). Unless otherwise indicated in this specification, an optional element of type Boolean is implied to be not answered if it is absent.

Date and Time Formats

This specification uses the date and time format specified in the ISO 8601 standard. The specification includes five time-related compound data types: YrMon, Date, Time, DateTime, and Timestamp. In all types that describe Date information, the specification uses the Gregorian calendar. Data types including time information refer to a 24-hour clock.

There is one format for representing dates, times, and time zones. The complete form is:

YYYY-MM-DDTHH:mm:ss.ffffff±HH:mm

where all punctuation and the *T* are literal characters; “YYYY” represents a four-digit year; “MM” represents a two-digit month; “DD” represents a two-digit date; the first “HH” represents a two-digit, 24-hour format hour; the first “mm” represents a two-digit minute; “ss” represents a two-digit second; and “ffffff” represents fractional seconds, and may be of any length. The second “HH” and “mm” describe the time zone offset from coordinated universal time (UTC), in hours and minutes, respectively. The “±” can be either a “+” or a “-” depending on whether the time zone offset is positive or negative.

All date and time types include (with the largest units given first) year, month, day, hour, minute, second, and fractions of a second. Any particular type may include a subset of these possible values. Types including time information (hour, minute, and so on) may also include an offset from Coordinated Universal Time (UTC).

> **Note:** In a DateTime element, specifying a date without a time or time zone will result in a time of midnight, UTC. This will result in the previous date appearing for all time zones in the Western Hemisphere. For example, October 5, 2002, without a specified time zone offset, will appear to be October 5, 2002, for the Eastern Hemisphere, but October 4, 2002, for the Western Hemisphere. Therefore, for DateTime elements where a single date is desired worldwide, the time must be included, and it must result in noon, UTC (for example, “12:00:00” or “09:00:00-03:00”).

As a general rule for date and time compound data types, values may be entered that omit the smallest logical elements. In every case, the value is taken to mean the same thing as if the minimum values (such as zeros) were included. (The default is always the start of an otherwise ambiguous range for types other than YrMon.) For example, a DateTime value omitting the time portion means the start of the day (12:00 midnight). Note that time zone qualifiers (in time and DateTime values) are exceptions to this rule, as they may be included even if times are not specified to the millisecond.

The logical elements appearing in each of these compound data types are summarized in the following table. “Required” means that the element must occur in all instances of the data type. “Recommended” means that the element should be included in all instances of the data type. “Optional” elements may be omitted from an instance of the data type. Optional elements must be included if smaller elements are to be included. For example, month must not be omitted from a *date* value if day is included.

	Contains	YrMon	Date	Time	DateTime	Timestamp
Year	YYYY 0000-9999	Required	Required	N/A	Required	Required
Month	MM 01-12	Required	Optional	N/A	Required	Required

	Contains	YrMon	Date	Time	DateTime	Timestamp
Day	DD 01-31	N/A	Optional	N/A	Required	Required
Hours	HH 0-23	N/A	N/A	Required	Optional	Required
Minutes	MM 0-59	N/A	N/A	Optional	Optional	Required
Seconds	SS 0-60	N/A	N/A	Optional	Optional	Required
Fractional Seconds	XXX (minimum) Precision is determined by the implementation	N/A	N/A	Optional	Optional	Optional
UTC offset (time zone indication)	Hours/Minutes -12:59 to +12:59	N/A	N/A	Recommended	Recommended	Recommended

YrMon

Elements of data type YrMon contain an indication of a particular month. This data type describes a unique period of time (not a repeating portion of every year

Tags specified as type YrMon accept years and months in the YYYY-MM format.

Date

Elements of data type Date contain an indication of a particular day. This data type describes a unique period of time, normally 24 hours (not a repeating portion of every year).

Tags specified as type Date accept dates in the YYYY-MM-DD format.

Time

Elements of data type Time contain an indication of a particular time during a date. This data type describes a repeating portion of a day. That is, each time described (ignoring leap seconds) occurs once per calendar date. In the specification, it is required that a time data type be able to represent a specific period with indefinite precision. Milliseconds are the minimum required precision of the time data type.

Tags specified as type Time accept times in the following format:

hh:mm:ss.ffffff±HH:mm

A time represented using this data type must not be ambiguous with respect to morning and afternoon. That is, the time must occur once and only once each 24-hour period.

In addition, the Time data type must not be ambiguous with respect to location at which the time occurs. If unspecified, the time zone defaults to Coordinated Universal Time (UTC). Generally, use of a specific time zone in the representation is preferred. The time zone should always be specified to avoid ambiguous communication between clients and servers.

DateTime

Tags specified as type DateTime accept a fully formatted date/time/time zone string. For example, "1996-10-05T13:22:00.124-5:00" represents October 5, 1996, at 1:22 and 124 milliseconds p.m., in Eastern Standard Time. This is the same as 6:22 p.m. Coordinated Universal Time (UTC).

Several portions of a DateTime element are optional. The following table describes the optional components and the meaning if they are absent.

Component	Meaning If Absent
±HH:mm (time zone offset)	+00:00 (UTC)
THH:mm:ss.ffffff±HH:mm (time component)	T00:00:00+00:00 (midnight, UTC)
:ss.ffffff (seconds and fractional seconds)	:00.000000 (zero seconds)
.ffffff (fractional seconds)	.000000 (zero fractional seconds)

Note: Time zones are specified by an offset, which defines the time zone. Valid offset values are in the range from -12:59 to +12:59, and the sign is required.

Timestamp

Elements of data type Timestamp contain the same information as DateTime values. Unlike that data type, Timestamp information is not intended to have meaning at the other end of the communication. In addition, microseconds are the minimum required precision of the time portion of this data type.

The intent here is to describe a type identical to DateTime but without semantic meaning between two machines. The general DateTime data type has meaning on both ends of the protocol (even though time synchronization is not required by this specification). Timestamp indicates an exact point in time with respect to the generating application.

For example, a Timestamp value may be generated at a server when creating an audit response. The client application may return that value to the server in later requests, but the client software should not interpret the information.

Phone Number

Phone Number indicates a string of up to 32 narrow characters in length (NC-32). It must begin with a plus sign (+) followed by country code, a hyphen, city/area code, another hyphen, and then the local phone number. If a PBX

extension is to be included, it must appear at the end of the field, separated from the rest of the telephone number by a plus sign.

For example, “+1-800-5551212+739” indicates PBX extension 739 at phone number 5551212 within area code 800 of North America (country code 1).

Decimal

Decimal indicates a numeric value that meets the following rules:

- The value is up to 15 digits in length, excluding any punctuation (sign, decimal, currency symbol, and so on).
- The value is not restricted to integer values and has a decimal point that may be placed anywhere from the start to the second last digit in the value, but not after the last digit (for example, +.12345678901234 is acceptable while 12345678901234567 is not).
    - The sign is always optional. If it is absent, the value is assumed to be positive.
    - Absence of a decimal point implies one after the last digit (that is, an integer).
    - The Decimal data type is always expressed as a Base-10, ASCII-character-set string.

      For example: +1234567890.12345 is acceptable, while 12345678901234567 is not.

Long

The Long data type is an Integer expressed as a Base-10, ASCII-character-set string representation of a 32-bit signed integer in the range -2147483648 to +2147483647. Elements of type Long do not permit a decimal point.

Enum

Enum is a Narrow Character type that has a limited number of specified valid values, each of which is represented by a tag of up to 80 characters. The Enum data type is either a Closed Enum or an Open Enum. Adding a value to a Closed Enum requires a spec update, while adding a value to an Open Enum only requires out-of-band agreement by the end points. Open Enums may also be extended using SPX.

Wherever it is appropriate to reference a non-ACORD code list, a reference to the <CodeList> aggregate can be created as listed in the following table. (See the Common Aggregates section of the Business Message Specification.)

Tag	Type	Usage	Description
@CodeInfoRef	Identifier Reference	Optional	A Reference to the Identifier of the <CodeList>

Closed Enum

A Closed Enum is an element where a number of valid values are defined within this specification. All other values should be rejected as invalid.

Open Enum

An Open Enum is an element where a number of valid values are defined within this specification, but other values should not be rejected as invalid by any system other than the final message destination. Open Enums provide a mechanism for a client and final destination server to communicate with values that may be known to both endpoints but not to all intermediate servers that route the message. Open Enums are typically used for elements related to system message processing and have been defined as open to support extensibility and customization of the specification.

Identifiers

This specification provides three different types of identifiers:

- Assigned Identifiers
- Transient Unique Identifiers
- Universally Unique Identifiers

Assigned Identifiers

An assigned identifier is created by an organization, carrier, agent, state, or other body. These include policy numbers, social security numbers, passport IDs, driver's license numbers, and so on.

Object identifiers in the specification are of the data type "Assigned Identifier". This is a Character data type with a maximum length of 36.

Transient Unique Identifiers

An ACORD document provides a unique identifier with the XML stream that is used for referencing information within the document. This is a transient identifier, as listed in the following table, that is only used to link information within a document stream. As the word *transient* implies, the identifiers are not meant for use once the message has been processed by the receiving system.

Transient identifiers in the specification are of the data type "Identifier." This is Character data that matches the XML rules for ID attribute data type values.

Tag	Type	Usage	Description
@id	Identifier	Optional	A document unique identifier used when an object (element) needs to be referenced elsewhere in the document. An ID should only be present on an element when it is being referenced within the stream.

Transient identifiers are not used on framework tags. These identifiers are used on all elements and aggregates that appear after the business message level except:

- <ActionCd>
- <PreviousValue>
- <ChangeDesc>
- <RqUID>
- <SystemID>

The transient identifier is optional, except when used with the following tags, when it is required:

- <SPFieldEditDefinition>
- <SPRelationalEditDefinition>

Universally Unique Identifier (UUID)

UUID elements are Narrow Character with a maximum length of 36. Applications can often obtain conforming UUIDs by calls to the operating system or the run-time environment.

The following information on UUID is based on Internet-Draft <leach-uuids-guids-01.txt>:

A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. To be precise, the UUID consists of a finite bit space. Thus, the time value used for constructing a UUID is limited and will roll over in the future (approximately at A.D. 3400, based on the specified algorithm). A UUID may be used for multiple purposes, from tagging objects with an extremely short lifetime to reliably identifying very persistent objects across a network.

The generation of UUIDs does not require that a registration authority be contacted for each identifier. Instead, it requires a unique value over space for each UUID generator. This spatially unique value is specified as an IEEE 802 address, which is usually already available to network-connected systems. This 48-bit address may be assigned based on an address block obtained through the IEEE registration authority. This section of the UUID specification assumes the availability of an IEEE 802 address to a system desiring to generate a UUID, but if one is not available, section 4 specifies a way to generate a probabilistically unique one that cannot conflict with any properly assigned IEEE 802 address.

In its most general form, all that may be said of the UUID format is that a UUID is 16 octets, and that some bits of octet 8 of the UUID called the variant field (specified in the next section) determine finer structure.

For use in human-readable text, a UUID string representation is specified as a sequence of fields, some of which are separated by single dashes. Each field is treated as an integer and has its value printed as a zero-filled hexadecimal digit string with the most significant digit first. The hexadecimal values a to f inclusive are output as lowercase characters, and are not case sensitive on input. The sequence is the same as the UUID constructed type. The formal definition of the UUID string representation is provided by the following extended BNF:

String	Value
UUID	<time_low> “-” <time_mid> “-” <time_high_and_version> “-” <clock_seq_and_reserved> <clock_seq_low> “-” <node>
time_low	4*<hexOctet>
time_mid	2*<hexOctet>

String	Value
time_high_and_version	2*<hexOctet>
clock_seq_and_reserved	<hexOctet>
clock_seq_low	<hexOctet>
node	6*<hexOctet
hexOctet	<hexDigit> <hexDigit>
hexDigit	zero | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”| “a” | “b” | “c” | “d” | “e” | “f” | “A” | “B” | “C” | “D” | “E” | “F”

The following is an example of the string representation of a UUID:

f81d4fae-7dec-11d0-a765-00a0c91e6bf6

Identifier References

An Identifier Reference is a technique for referencing an identifier on an aggregate or element that is elsewhere in the stream. This specification provides two different types of identifier references:

- Identifier Reference
- Multiple Identifier References

Identifier Reference

Related to the transient identifier is its matching reference. Identifier References in the specification are of the data type “Identifier Reference.” This is Character data that matches the XML rules for ID attribute data type values and it must match a value in the current data stream. These values are typically shown as @xxxRef, where xxx is replaced with a value that describes the type of object or tag that the item references.

There is a special Identifier Reference, called CodeListRef, used on all tags of type Open and Closed Enum. Its usage is always optional. When used, it should reference the ID of a CodeList aggregate that identifies (among other things) the owner of the code list. Although it is not shown in the rest of this document, it is defined in the next section.

Multiple Identifier References

When an aggregate or element references more than one item in the stream, a “Multiple Identifier References” data type is defined and is typically shown as @xxxRefs, where xxx is replaced with a value that describes the type of object that the item references.

URL

A Uniform Resource Locator (URL) is of the Narrow Character data type with a length of up to 1024 characters (NC-1024). URLs are defined in RFC 1738, which is a subset of the Uniform Resource Identifier (URI) specification (RFC 2396). URLs contain only the printable US-ASCII characters 32 through 126 decimal.

An element of the Uniform Resource Locator URL data type specifies the URL where a customer may access information. A URL is of the Narrow Character data type with a length of 1024 Characters (NC-1024). The format of a URL begins with a string that identifies which protocol is to be used to access the information, such as “http://”.

6 Troubleshooting

Troubleshooting

This chapter describes run-time setup problems in workflows, integration object setup problems, a rollback problem, and an envelope problem.

Run-Time Event Setup Problems in Workflows

The following sections describe run-time event setup problems in workflows.

Workflow is double-triggered

Problem

After configuring a runtime event to trigger a workflow process, the system freezes without returning an error, or it returns the error message, “Try to read or write to the invalid memory address.”

Reason

A workflow may trigger the same runtime event that initially launched the workflow. Therefore, the same workflow that is triggered is triggered again. The result is an infinite loop.

Solution

Use an applet as the event object type instead of using a business component. Doing so restricts the event to a single view instead of multiple views. Business components are reused in many different ways. If it is certain that the selected business component will not be changed during the processing of the workflow, users will not encounter this problem.

Workflow is not triggered

Problem

The workflow is not triggered although the event has been configured.

Solution

Verify that the following settings are correct:

- The branch type within the start step of the workflow is correct.
  If you have defined a runtime event outside the workflow, the branch type within the start step should be "Default." If you have defined a runtime event inside the workflow (from Workflow Designer), the start step should be "Condition."
- The process name in the Runtime Event Administration view is correct.

For example, a workflow process name is "ACORD Add Auto Policy Outbound workflow," as listed in the following table.

Field Name	Field Value
Business Service Name	Workflow Process Manager
Business Service Method	RunProcess
Business Service Context	"ProcessName,""ACORD Add Auto Policy Outbound workflow"

Note: A blank space is required between "ProcessName" and "ACORD Add Auto Policy Outbound workflow."

Object Id is not passed to the workflow

Problem

The Object Id is not passed to a workflow correctly when using Runtime Event Management to trigger the workflow.

Solution

The business object name of the workflow is not set up correctly. Please select the business object name desired.

Integration Object Setup Problems

The following sections describe integration object setup problems.

Errors in some component fields

Problem

FINS ACORD XML Transaction Manager returns an error message indicating that an error occurs in some component fields.

Solution

Make inactive all unused integration component fields for internal integration objects created by the FINS ACORD Wizard. In some cases, two or more integration component fields are based on the same table column, which causes an SQL error in the object manager.

Transaction manager cannot create or update a record

Problem

The <MsgStatus> information in the response ACORD messages indicates that the FINS ACORD XML Transaction Manager cannot create or update a record.

Solution

Check to make sure the current User Keys of the integration components are properly configured. User Keys created by the FINS ACORD Wizard may not fully meet a customized situation. Customize the user key combination as necessary to allow the transaction manager to uniquely identify a business component record.

All required fields, not including system fields, need to have an initial value when a new record of a business component is created. Please make sure those fields are properly initialized.

Some required fields are empty in an XML string

Problem

Some required fields are empty in an XML string generated by the FINS ACORD XML Converter.

Solution

Initialize all required fields in the integration component “XML Literal Value” column. Some values are specified in the integration object/component/component field user property. Be sure to customize them to suit your integration needs.

Rollback Problem

The following section describes a rollback problem.

Rollback operation fails

Problem

A needed rollback does not happen. For example, after running an ACORD outbound workflow to add a policy record, if the returning response message contains errors indicating that the operation is unsuccessful, the workflow ACORD Add Policy Outbound Response Workflow With Rollback should delete (roll back) the original record from the database.

Solution

The probable reason is that the rollback operation is improperly defined in the FINS ACORD XML Transaction Manager. Use Oracle's Siebel Tools to make sure that the rollback operation is defined in the business service user property of the FINS ACORD XML Transaction Manager. The operation name is "SAUpsert_ROLL_BACK", and the value is "EAI Siebel Adapter/Delete/RollbackOnSame;". Do not forget to deliver the changes to the integration branch.

Envelope Problem

The following section describes an envelope problem.

Message does not show user-selected information

Problem

A user chooses header elements while generating an envelope integration object, but the resulting message for initial request does not show the chosen header information.

Solution

When a user creates a user-defined envelope integration object, all the user-chosen tags are memorized in the userdefined user property of each integration component of the envelope integration object. The value of this user property of each integration component indicates the names of its child integration components.

In the workflow, find the business service corresponding to the "PropsetToXMLPropSet" method of the FINS ACORD XML Converter. Change the value of the input argument from initsignon to userdefined. Doing so causes the user property userdefined to be used to construct the initial SignonRq header instead of the default value initsignon. The SignonRq header will reflect the user-chosen elements.