Siebel Financial Services Connector for IAA-XML Guide

Siebel Innovation Pack 2014
November 2014

ORACLE®

Contents

Contents

1 What's New in This Release

What's New in Siebel Financial Services Connector for IAA-XML Guide, Siebel Innovation Pack 2014

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

NOTE: Siebel Innovation Pack 2014 is a continuation of the Siebel 8.1/8.2 release.

2 Overview of Siebel Connector for IAA-XML

Oracle's Siebel Connector for IAA-XML provides integration between Oracle's Siebel Business Applications and other IAA (Insurance Application Architecture) based business models such as Customer Information Integration Solutions (CIIS) or other applications which support IAA-XML standard. The Siebel Connector for IAA-XML is a configurable set of components that you can use to exchange data between Siebel Business Applications and external IAA based applications and databases.

IAA are set of business data models built to support insurance industry. IAA-XML is IBM's messaging architecture standard for insurance industry. It allows different applications having different data models and running on different technologies to communicate. IAA-XML has 3 key benefits that address the problems in inter application communications:

  Lack of shared semantic and content structure

  Lack of shared syntax and

  Technology Dependency

IAA-XML provides a messaging standard for inter-application communications for the insurance and financial services industries. For example, when two companies merge, IAA-XML can be used as the standard for the design of messages in a hub or a spoke architecture, providing the mechanism for integrating systems built on different platforms. It can also be used as a way of opening systems to business partners. For example, communication between a financial services company and its external network of intermediaries.

The Siebel Connector for IAA-XML is responsible for receiving, parsing and processing the business operations specified in the XML message that conforms to the specification detailed out in the IAA-XML specification document. This integration offers powerful capabilities designed to meet all the IAA message specification requirements. This solution allows you to effectively harness the synergies between Siebel front office applications and IAA based applications. The Siebel Connector for IAA-XML extends Siebel applications to integrate with back office data and business processes.

The Siebel Connector for IAA-XML supports both synchronous and asynchronous transactions across application boundaries. The resulting consistency and sharing of data allows efficient coordination between front and back office operations. For example, sales and service professionals can enter policy information in Oracle's Siebel Financial Services applications and receive real-time response with a quote for the policy entered. The sales or service professional can then complete entering the policy details through the Siebel application interface by completing the requirements and issuing the policy to the customer.

Using the Siebel Connector for IAA-XML

To use the Siebel Connector for IAA-XML you need to

  Install all required components. See "Required Components" on page 8.

Create integration objects. See "To Create Integration Object" on page 41.

Define mappings for data between the Siebel application and the IAA-XML based external application. See "To configure the DTE map" on page 53.

Create integration workflows based on the mapped objects. See "Sample Workflow IAA Add Party Outbound Workflow" on page 56 and "Sample Workflow IAA Server Party Package Workflow" on page 60.

Required Components

In order to implement messages between Siebel Business Applications and IAA compliant applications the Siebel Connector for IAA-XML requires the following components:

Siebel Financial Services Business Applications

Licensed to use the Siebel Connector for IAA-XML.

NOTE: To obtain license for Siebel Connector for IAA-XML from Siebel Manufacturing Operation contact your Oracle sales representative.

Licensed to use IAA (Insurance Application Architecture) models. Additional information about licensing these models can be obtained from your IBM Sales Representative or from www.ibm.com.

Optional Components

In addition to required components, there are optional component you may want to install to help with your integration process. These components are:

License to use MQSFSE (MQSeries Financial Services Edition). For details, see "The MQSeries Financial Services Edition" on page 33.

NOTE: For more information, contact your IBM Sales Representative or visit www.ibm.com.

License to use Oracle's Siebel Event Manager to instantiate a Siebel Workflow Manager or you can write an eScript to instantiate your Siebel Workflow Manager. Siebel Workflow is delivered as a part of Siebel Financial Services Business Applications.

Architectural Overview

Siebel Financial Services EAI architecture is built on top of Oracle's Siebel Enterprise Application Integration (EAI) architecture. Oracle's Siebel Financial Services EAI framework has been built to support XML messaging based communication infrastructure. Customers of Financial Services have the need to integrate with many different applications through messaging mechanisms. To fulfill this requirement, many connectors have to be built to support various industry standards.

Siebel Financial Services allows you to build and deploy multiple connectors. To demonstrate such flexibility, Siebel Financial Services Application has built three connectors—Siebel Connector for IAA-XML, Siebel Connector for ACORD XML, and Siebel Connector for IFX XML—all based on the Siebel Financial Services EAI framework.

Figure 1 illustrates the high-level architecture of the Siebel Financial Services EAI and the standard connectors.

NOTE: For details, see *Siebel Financial Services Enterprise Application Integration Guide*.
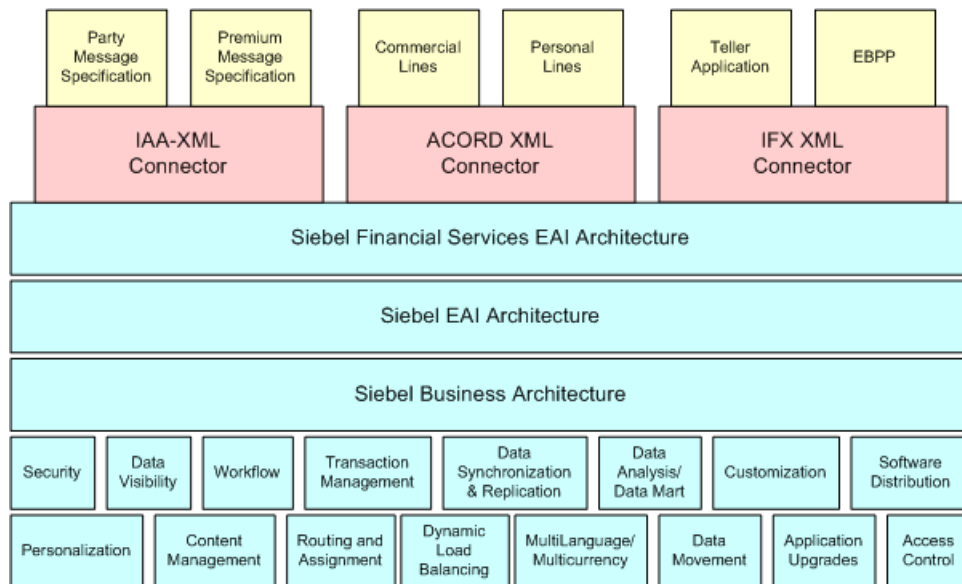
Figure 1.   High-level Architecture of Siebel Financial Services EAI

Siebel Connector for IAA-XML is built based on IAA-XML standards for insurance industry exchange and is designed to address the real-time requirement by defining transactions that include both a request and a response message. The Siebel Connector for IAA-XML provides functions such as:

Handling message header

Handling heterogeneous commands in the body section of an XML message

Data type formatting and conversions

Data model mapping through various modules

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

Business Data Flow

Each standard integration or custom integration is based on the creation of business data flows. A business data flow controls the whole transformation of an IAA based data object to a Siebel data object and a Siebel data object to an IAA based data object. , illustrates inbound and outbound business data flows. Some of the business flows might constitute messages published by IAA such as <AddPartyRequest>, <DeletePartyRequest>, <FindPartyRequest>, <ModifyPartyRequest>, <SubmitApplicationRequest>, <InquiryFinancialServicesAgreementRequest>, <InquiryFinancialServiceAgreementResponse> and so on. Some of these messages are included in the sample database as reference implementations.

There are two types of business data flows:

Outbound to External IAA-based application (Send)

Inbound from an external IAA-based application (Receive)

The processing flow for each of these business data flow is largely contained within a Siebel workflow process. The workflow process is then initiated by the Siebel Event Manager, or by a call from Oracle's Siebel eScript based on an action performed within Siebel Business Applications to process the data in an outbound scenario. In an inbound scenario, the workflow will be instantiated by the Business Integration Manager.
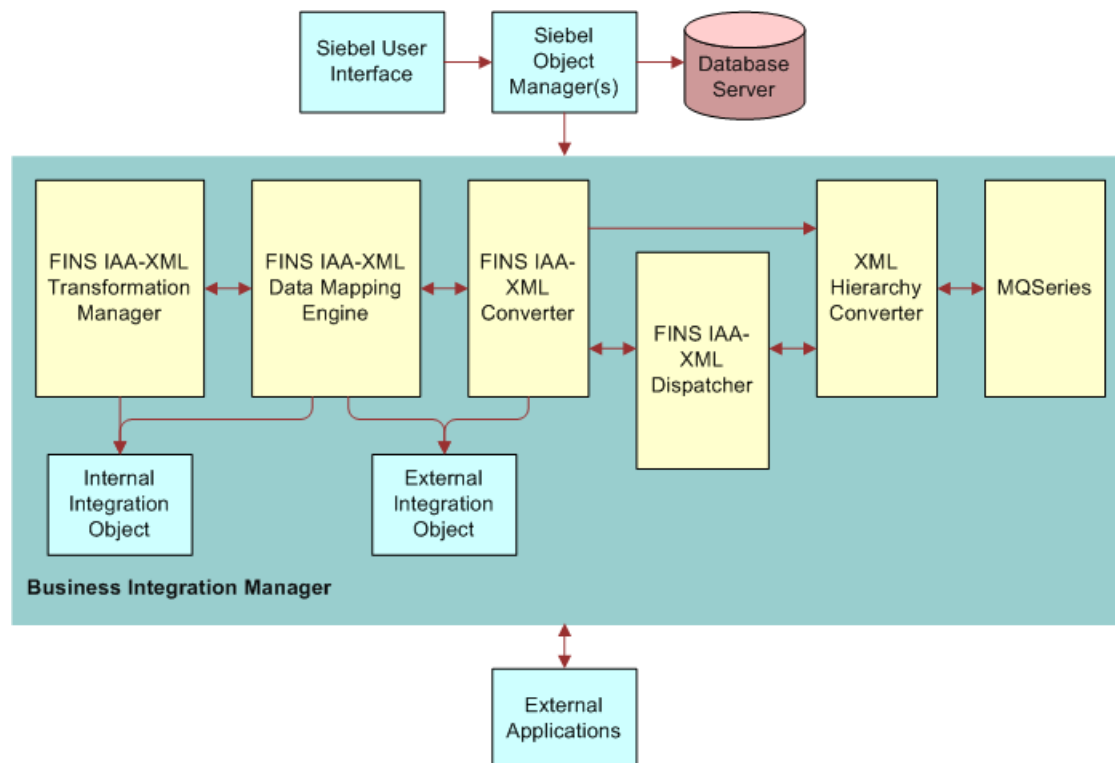
Figure 2.   Bi-directional Business Data Flow of Siebel Connector for IAA-XML

Outbound Data Flow

As shown in Figure 2 on page 10, during an outbound data flow, followings are the order of process that taking place:

1   The workflow is instantiated, the FINS IAA-XML Transaction Manager takes all the ROW_IDs of the objects and the operation name as input.

2   The FINS IAA-XML Transaction Manager then extracts data from the Siebel database and then returns all the instances retrieved as property sets.

This data is then used to populate the internal integration objects based on the Siebel business objects.

NOTE: Property set is the representation of data in the internal format in memory and is used widely by the business services.

3   The internal integration object instances are then passed to the FINS IAA-XML Data Transformation Engine (DTE) which transform the internal integration object instances into external integration object instances. It also adds all necessary IAA specific command layer attributes into the instances.

4   Next, the FINS IAA-XML Converter converts all external integration object instances into proper XML integration object instance and adds the envelope, header, and other sections to the newly converted instance.

5   Then XML Hierarchy Converter converts this XML integration object instance from a property set format into a text format.

6   As a last step, the message is then sent to external systems using any transport mechanism supported by Siebel EAI such as, HTTP or MSMQ Receiver supported by Oracle's Siebel EAI.

Inbound Data Flow

As illustrated in Figure 2 on page 10, an inbound business data flow starts with a Receiver Server Component such as the MQSeries, HTTP, or MSMQ Receiver. The Receiver runs in the background continuously waiting for message to arrive from external IAA-XML based applications. Upon receiving an IAA-XML message, it then invokes the workflow process configured to handle and process the data further. The workflow dictates the business logic behind the Siebel Connector for IAA-XML as follows:

1   The raw XML text string is passed through XML Hierarchy Converter to be converted into an XML integration object instance.

2   The FINS IAA-XML Dispatcher then traverses the XML instance and identifies the messages received according to the rule sets of the IAA Dispatcher Map and then identifies the envelope, header and body sections among the hierarchy nodes and sends it to the FINS IAA-XML Converter.

NOTE: The Dispatcher Map is created by the FINS IAA Wizard. For details, see "FINS IAA Wizard" on page 19

3   The FINS IAA-XML Converter then takes the XML instance, and process individual sections of the instance while converting each sub-tree into external integration object instances before sending it to the FINS IAA-XML DTE.

4 The FINS IAA-XML DTE transforms the external integration object instances into internal integration object instances and passes it to the FINS IAA-XML Transaction Manager.

5 The FINS IAA-XML Transaction Manger then performs the operation specified in the instance by invoking the services configured in its user properties.

Workflow Integration

Siebel Workflow is the center of the business data flow. Workflow processes control the flow and transformation of data into and out of the Siebel applications. You create the workflow processes using a graphical user interface provided within the Siebel applications called the Siebel Workflow Designer.

NOTE: For details on 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 Builder provided in Oracle's Siebel Tools. The Integration Object Builder can create Siebel Integration Objects from Siebel Business Objects. For IAA integration process you need to select the FINS IAA Wizard as a wizard to build your integration objects. This wizard reads an IAA-XML Document Type Definition (DTD) and creates the envelope and header integration objects, the required external integration objects as well as the required internal integration objects. It then associates all of these in the rule-based dispatcher map. For details, see Chapter 4, "Configuration Roadmap."

NOTE: For more information on Integration Objects, see *Overview: Siebel Enterprise Application Integration* and *Siebel Financial Services Enterprise Application Integration Guide.*

Business Services

Business services execute predefined or custom actions in a workflow process. Examples of business services include the FINS IAA-XML Transaction Manager, the Siebel EAI Adapter, the FINS IAA-XML Converter, and so on. These business services act on property sets passed to them. They perform business logic operations such as interfacing with database, interfacing to IAA-XML based systems, or transforming one integration object into another. Oracle provides many business services but you can create your own. Although you can use business services to perform many different functions, they all have a standard interface. Business services have object-like qualities, such as methods, method arguments, and user properties. These elements define how a business service can be used. Business services are defined in Siebel Tools. This guide describes those business services used to interface to IAA-XML based systems.

NOTE: For more information on business services in general, see *Integration Platform Technologies: Siebel Enterprise Application Integration.*

IAA-XML Standard

As for the DTD which can be used, IAA has an interface design model maintained in Rational Rose which allows you to use a generation script in Visual Basic to generate IAA-XML messages and the DTD. This script uses not only the UML model but also additional generation directives to get to the precision level required by running code. You need to license the IAA models from IAA to get access to these design models and scripts. You can view and process the generated DTD by standard DTD editors and XML tools including the FINS IAA Wizard.

NOTE: For information on required version, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network and for more information about IAA, see www.IBM.com.

3 Siebel Connector for IAA-XML

This chapter discusses IAA-XML connector syntax, including methods and arguments for the transformation engine, converter, and dispatcher.

IAA-XML Syntax and Rules

IAA-XML is the IBM's messaging architecture standard for the Insurance and Financial Services industry. IAA-XML allows different applications having different data models and running on different technologies to inter-operate. The IAA-XML addresses the deficiency of shared semantic and content structure, lack of shared syntax, and technology dependency in today's inter-application communications. IAA-XML addresses to the need for content definition for the XML messages. These XML messages can support both business-to-customer and business-to-business message models.

The Siebel Connector for IAA-XML performs both inbound and outbound integration. The IAA-XML architecture defines the architectural elements that can appear in IAA-XML messages, as well as the way in which they relate to each other. These architectural elements describe the organization of data and commands into messages, and include the concepts of message, command, aggregate, aggregate relationship and property. The following example illustrates the layers of an IAA-XML message:

```
<Message id='Message1' version='1.4' bodyType='IAA-XML'

timeStampCreated='2001-12-14T11:55:04'

sourceLogicalId='Siebel'

destinationLogicalId='CIIS'

authenticationId='SysAdmin'

crfCmdMode='alwaysRespond'>

<Default>

    <DefaultTime zoneOffset='00:00'/>

</Default>

<CrfActionGroup bodyCategory='AddAPerson'>

    <CommandReference refid='Req1'/>

    <KeyGroup id='R1' keyGroupType='Person'>

        <UUID>1514</UUID>

    </KeyGroup>

    <KeyGroup id='R2' keyGroupType='MailingAddress'>
```

```
        <UUID>1652</UUID>

    </KeyGroup>

    <KeyGroup id='R3' keyGroupType='WorkPhoneNumber' >

        <UUID>1665</UUID>

    </KeyGroup>

</CrfActionGroup>

<COMMAND>

    <AddPartyRequest id='Req1' cmdType='request' cmdMode='always Respond'
    defaultState='added' >

<Person>

    <KeyGroup refid='R1' />

    <birthDate>1961-08-04</birthDate>

    <deathDate>2001-12-20</deathDate>

    <gender>Male</gender>

    <maritalStatus>Married</maritalStatus>

    <grossIncome>

        <currencyAmount>15000</CurrencyAmount>

        <currencyCode>USD</CurrencyCode>

    </grossIncome>

    <PersonName>

        <usage>Legal</use>

        <firstName>Chris</firstName>

        <lastName>Conway</lastName>

    </PersonName>

    </Person>

    </AddPartyRequest>

</COMMAND>

</Message>
```

The IAA-XML standard is structured in layers to allow reusing elements and facilitating the processing of the messages. Before defining any content for the messages, it is important to identify the overall structure of the messages and the technical elements that are needed in order to process these messages. Starting from the Interface Design Model (maintained in Rational Rose), it is possible to use a generation script (in Visual Basic) to generate IAA-XML Messages. This script uses not only the UML model but also additional generation directives to get to the precision level required by running code. After the DTD is generated, it can be viewed and processed by standard DTD editors and XML tools.

NOTE: Siebel Connector for IAA-XML can generate IAA-XML even without generating the IAA-XML header. The header is only required to work with IBM MQSFSE product Suite.

Envelope Section

The Siebel Connector for IAA-XML implements the envelope section of an IAA-XML message through the use of envelope integration object generated by the FINS IAA Wizard. The envelope section contains header information for message management, and authentication. The header also defines default defines default routing and message processing (acknowledgement and publication). The header also contains an action group for each IAA-XML command tied together through an XML reference to a COMMAND element wrapping each IAA-XML command. This allows the MQFSE message hub to deal with each IAA-XML command separately for routing and message response purposes. The envelope section is referred as Message Layer in the IAA-XML Specification. The following is an example of message layer.

```
<Message id='21FGE620108:13:16' version='1.4' bodyType='IAA-XML'

timeStampCreated='2001-08-22-T23:59:00'

timeStampExpired='2001-10-24-T23:59:00'

sourceLogicalId='FrontEnd'

authenticationId='SysAdmin'

crfPublish='false'>

    <Default>

    ....

    <Default>

    <CrfActionGroup bodyCategory='AddAPerson'>

    ...

    <CrfActionGroup>

</Message>
```

Body Section

The Siebel Connector for IAA-XML implements the body section of an IAA-XML message through the use of external integration objects generated by the FINS IAA Wizard. The external integration object, which contains the information about an action and the subject of that action, refers to the Command Layer within the Message Layer of the IAA-XML Message. A body section may contain multiple external integration object instances, which corresponds to heterogeneous commands in a message. Each integration object instance provides enough information about the command for the receiving software to process it accurately. The following is an example of a body section in IAA-XML format.

```
<COMMAND>

    <ModifyPartyRequest id='123' cmdType='request'>

    <SystemInfo/>…</SystemInfo>

    <Person/>…</Person>

    </ModifyPartyRequest >

</COMMAND>
```

The external integration components refers to the Aggregate Layer in an IAA-XML message. It provides a convenient way to address groups of properties that are often used together. They also may serve as limiting, functional views of the subjects of the commands. External integration components can be referred to through their key, it can also be defined to represent the return of certain operations. The following is an example of external integration component in IAA-XML format.

```
<PersonName id='456'>

    <KeyGroup refid='R3'/>

    <usage>Legal</usage>

    <firstName>Chris</firstName>

    <lastName>Conway</lastName>

</PersonName>
```

The external integration component fields refers to the Property Layer which defines the atomic elements of the IAA-XML Message specification. An external integration component field defines a single piece of information that is passed between applications. The format of the information is specified in the integration component field's data type. The definition provided with the property describes more precisely the purpose of this information. External integration component fields can only appear inside an external integration component. The following is an example of external integration component field in IAA-XML format.

```
<Person>

    <birthDate>1962-11-11</birthDate>

</Person>
```

NOTE: For details on IAA-XML Standard and specification, see IBM documentation.

FINS IAA Wizard

Siebel Financial Services application provides different wizards to guide you through building various integration objects for integrating data with external applications. You can access these wizards using Siebel Tools. You should use the Integration Object Builder in Siebel Tools to create new Siebel integration objects. For example, you should use the FINS IAA Wizard within Integration Object Builder to build integration objects for the Siebel Connector for IAA-XML. The Integration Object Builder guides you through the process of building the necessary integration object for your integration project. The Integration Object Builder builds a list of valid components from which you can choose the components to include in your integration object. Table 1 lists all the user properties for FINS IAA Wizard.

Table 1.   FINS IAA Wizard User Properties

Name	Value	Description
DispatcherMapName	IXMLDispMap	The Dispatch Map Name, wizard will use this map to update the key and value.
HasUIControl	Y or N	Internal Use
Integration Object Wizard	Y or N	Internal Use
Integration Object Base Object Type	Siebel Business Object	Internal Use.
Operation KeyWord Match:X	Key/value	Using this user property wizard can find out the possible transaction operation. For example, if query operation always start with <inquiry***> then this user property is set as follows: Name: Operation KeyWord Match:0 Value: inquiry/IXMLOperation_Query where IXMLOperation_Query is defined in Transaction manager user property
Default Command Tag	COMMAND	Used by the Wizard only. Do not change.
Default Message Tag	Message	Used by the Wizard only. Do not change.

Integration Objects

Siebel integration objects allow you to represent integration metadata for Siebel business objects XML as common structures 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 are transformed declaratively using Siebel Data Mapper. To use the Siebel Connector for IAA-XML to integrate data you need to build three different integration objects.

IAA Envelope Integration Object. The Integration Object that defines the envelope and the header portions of the IAA-XML Message. For details, see "To Create IAA envelope Integration Objects" on page 42

IAA External Integration Objects. The integration object that defines the IAA-XML data hierarchy. For details, see "To create IAA external integration objects" on page 44.

IAA Internal Integration Objects. The integration object that defines the Siebel data hierarchy. For details, see "Creating IAA Internal Integration Objects" on page 46.

IAA Dispatcher map

The IAA dispatcher map is created automatically by the FINS IAA Wizard. The IAA dispatcher map serves as rule sets for FINS IAA-XML Dispatcher to identify and associate the IAA-XML message to the external integration objects created. It also indicates the Data Transformation Maps to be used for transforming the messages received. For details, see "IAA Dispatcher Map" on page 48.

FINS IAA-XML Transaction Manager

The FINS IAA-XML Transaction Manager executes operations specified in IAA-XML message instance as Siebel database transactions. It walks through heterogeneous commands and execute transactions and invokes Siebel EAI Adapter or other business services, configured in its user properties, multiple times. The FINS IAA-XML Transaction Manager also translates IAA-XML command elements to Siebel Adapter actions and combines return results as a single property set. Table 2 lists all the user properties for FINS IAA-XML Transaction Manager.

Table 2.   FINS IAA-XML Transaction Manager User Properties

Name	Value	Description
DispatcherMapName	IAADispMap	Transaction manager uses this map to tag the Body information for other component. This value can be set as a runtime input argument as well.
IgnoreSvcMethodArgs	true or false	This parameter allows runtime input arguments.
SaveInFileForRollback	<filename>	The filename to save current record for future rollback when rollback operation is desired.
SaveInMemForRollback	<filename>	The session key to set or look up the record cached in memory if rollback operation is desired.
XXX (Operation)	ServiceName/MethodName/ Argument	Type of operation to use. For details, see "Configuring the IAA-XML Business Service" on page 50.

The FINS IAA-XML Transaction Manager uses the pre-built methods described in Table 3 to process inbound or outbound.

Table 3.   FINS IAA-XML Transaction Manager Methods

Name	Display Name	Description
Execute	Execute Transaction	This method can be used for inbound or outbound purpose, as long as the integration object instance is provided. Note that you should use the Execute Outbound method when Row Id is the only available input.
ExecuteOutbound	Execute Outbound	This method can only be used for outbound purpose to execute operation specified in input arguments. See Table 6 on page 22.
ExecuteSave	Execute and Save	This method executes outbound operation specified in input arguments and also saves the transaction into memory or file. See Table 7 on page 22.

FINS IAA-XML Transaction Manager business service uses combination of user property, method, and method arguments to achieve different tasks. Table 4 lists all the method arguments available to use with FINS IAA-XML Transaction Manager business service.

Table 4.   FINS IAA-XML Transaction Manager Method Arguments

Argument	Default Value	Description
OnlyIOI	false	This is only used for Inbound integration. The inbound message may contain header, body, and envelope portion. When Transaction Manager takes the proper operation against the Siebel application, the integration object instances for response is generated as well. All information from the request message is dropped if this value were set to true. Therefore, the FINS IAA-XML Converter and FINS IAA-XML Data Transformation Engine do not need to deal with the overheads. If this is not set to true then request information will still be carried over.
XMLHierarchy	N/A	Property set in internal integration object hierarchy.
IXMLMapPath	N/A	Store the key value for the dispatch map. FINS IAA-XML Transaction Manager will use it to lookup the value, and attach necessary value into integration object instance.
PrimaryRowId	N/A	The PrimaryRowId of the integration object.
SiebelFINSOperationOut	N/A	The outbound operation FINS IAA-XML Transaction Manager takes through its user property.
SearchSpec	N/A	The Search Specification for query.

Table 4.   FINS IAA-XML Transaction Manager Method Arguments

Argument	Default Value	Description
PlaceToSave	mem	PlaceToSave indicates whether the rollback instance is getting from file or mem. Valid value are "file" or "mem."
RollbackInError	false	Indicates whether the transaction should be rolled back. Valid values are "true" or "false."
ReportErrInMsg	false	Indicates to report error to screen. Valid values are "true" or "false."

Although these arguments are available to be used by FINS IAA-XML Transaction Manager but not all can be used with each method. Table 5 through Table 7 list all the method arguments for each method.

Table 5.   Method Arguments for Execute Method

Argument	Display Name	Data Type	Type	Optional
OnlyIOI	Only produce Integration Object Instance	String	Input	Yes
XMLHierarchy	XML Property Set	Hierarchy	Input/Output	Yes

Table 6.   Method Arguments for ExecuteOutbound Method

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

Table 7.   Method Arguments for ExecuteSave Method

Argument	Display Name	Data Type	Type	Optional
IXMLMapPath	IXML Map Path	String	Input	No
PrimaryRowId	Primary Row Id	String	Input	Yes
SiebelFINSOperationOut	Outbound Operation	String	Input	No
SearchSpec	Search Specification	String	Input	Yes

Table 7.   Method Arguments for ExecuteSave Method

Argument	Display Name	Data Type	Type	Optional
XMLHierarchy	XML Property Set	Hierarchy	Output	No
PlaceToSave	Place To Save	String	Input	Yes

FINS IAA-XML Data Transformation Engine

The Siebel Data Mapper provides you with a declarative interface to specify maps for both inbound and outbound data transformation. The maps you set up using the Siebel Data Mapper call the FINS IAA-XML Data Transformation Engine to complete the data transformation. The FINS IAA-XML Data Transformation Engine both transforms Siebel internal object hierarchy to external integration object hierarchy, defined in IAA-XML DTD, and external integration object hierarchy to Siebel internal integration object hierarchy. The FINS IAA-XML Data Transformation Engine walks through and transform elements in heterogeneous commands and with proper input integration objects and maps it combines results as a single property set. The FINS IAA-XML DTE also removes the empty-valued integration component fields when required. Using the Siebel Data Mapper one can often reduce or even eliminate the number of scripts you need to write.

NOTE: For details on Siebel Data Mapper and EAI Data Mapping Engine, the base components for FINS IAA-XML Data Transformation Engine, refer to *Business Processes and Rules: Siebel Enterprise Application Integration*.

Like any other Siebel business service, FINS IAA-XML Data Transformation Engine has specific methods as described in Table 8.

Table 8.   FINS IAA-XML Data Transformation Engine Methods

Name	Display Name	Description
ToExternal	Transform To External Hierarchy	Transforms Siebel internal integration object hierarchy into the IAA-XML external hierarchy.
ToInternal	Transform To Siebel Hierarchy	Transform IAA-XML external hierarchy into the Siebel internal integration object hierarchy.

FINS IAA-XML Data Transformation Engine business service uses combination of method, and method arguments to achieve different tasks. Table 9 lists all the method arguments available to use with FINS IAA-XML Data Transformation Engine business service.

Table 9.   FINS IAA-XML Data Transformation Engine Method Arguments

Argument	Default Value	Description
XMLHierarchy	N/A	Property set in internal or external integration object hierarchy.
<MapArgs>	N/A	See *Business Processes and Rules: Siebel Enterprise Application Integration*.

Each method defined for the FINS IAA-XML Data Transformation Engine business service takes some input arguments and produces some output arguments as listed in Table 10 and Table 11.

Table 10.   Method Arguments for ToExternal Method

Name	Display Name	Data Type	Type	Optional
XMLHierrarchy	XML Property Set	Hierarchy	Input/Output	No
<MapArgs>	-	String	Input	Yes

Table 11.   Method Arguments for ToInternal Method

Name	Display Name	Data Type	Type	Optional
XMLHierrarchy	XML Property Set	Hierarchy	Input/Output	No
<MapArgs>	-	String	Input	Yes

FINS IAA-XML Converter

The FINS IAA-XML Converter provides a way to convert data from the IAA property set to IAA-XML messages for outbound communication, and converts IAA-XML messages received to IAA property set for inbound communication. It iterates through the requests and responses in the message to construct the IBM EID Hub aggregated in the header section. It also constructs the envelope section of the message and data type formatting for fields. Table 12 lists all the user property for this converter.

Table 12.   FINS IAA-XML Converter User Properties

Name	Value	Description
EscapeNames	true or false	If set to true, the converter replaces illegal XML name characters with escape characters. Otherwise, it will ignore illegal XML name characters. The default value is true.
HierarchyFormat	XML Hierarchy	Format of the property set input or output. For Internal Use.
XMLEnvIntObjectName	<integration object name>	Name of an integration object that defines the content and hierarchy for the envelope and header section of IAA-XML. The default envelope integration object is "IAA Envelope."

This service provides several ready-to-use-methods as described in Table 13.

Table 13.   FINS IAA-XML Converter Methods

Methods	Display Name	Description
PropSetToXML	PropSetToXML	Converts an integration object hierarchy to XML string.
PropSetToXMLPropSet	PropSetToXMLPropSet	Converts an integration object hierarchy to XML hierarchy.
XMLToPropSet	XMLToPropSet	Converts an XML string to an integration object hierarchy.
XMLPropSetToPropSet	XMLPropSetToPropSet	Converts an XML hierarchy to an integration object hierarchy.
ErrorHandler	ErrorHandler	Constructs the XML hierarchy to respond to the external systems when undesired operation or configuration errors occurred.

For each method defined the FINS IAA-XML Converter business service takes input arguments and produces output arguments. These arguments are described in Table 14.

Table 14.　FINS IAA-XML Converter Method Arguments

Argument	Default	Description
XML Document	N/A	XML document streams.
Escape Names	true	Escape characters, invalid XML characters, to be removed or not.
Ignore Character Set Conversion Errors	false	If some characters cannot be represented in destination character set, for example, local codepage, the errors can be ignored.
XML Character Encoding	N/A	XML character encoding to use in the output XML document. If encoding is blank or not supported, errors will be produced.
XML Header Text	N/A	Text to pre-append to the XML document.
XML Hierarchy	N/A	Property Set in external integration object or XML hierarchy.
External Entity Directory	N/A	Location of an external entity files, such as a DTD file.
Validate External Entity	false	If true, the XML parser will validate the document according to the DTD specified in the <DocType> element.
Truncate Field Values	true	truncate field values.
Contains Inline Attachments	true	Indicates the message contains attachment documents.
GereralErrorMessage Text	N/A	The general text to be pre-append to the <errorMessageText> element in IAA fault section.
ErrorMessageText	N/A	The actual error text to be set to the <errorMessageText> element in IAA fault section.
ErrorCode	N/A	The actual error code to be set to the <errorCode> element in IAA fault section.
ErrXMLHierarchy	N/A	Input hierarchy will contain the request XML hierarchy to be associated with when generating the respond XML hierarchy. This argument can be ignored. Output hierarchy will contain the XML hierarchy with <ErrorInfo> aggregate generated.

Each FINS IAA-XML Converter method uses group of these method arguments to convert the data for the next service in the integration process. Table 15 through Table 19 display the method arguments for each method.

Table 15.   Method Arguments for PropSetToXML Method

Name	Default Display Name	Data Type	Type	Optional
<Value>	XML Document	String	Output	Yes
EscapeName	Escape Name	String	Input	Yes
IgnoreCharSetConvErrors	Ignore Character String Set Conversion Errors	String	Input	Yes
XMLCharEncoding	XML Character Encoding	String	Input	Yes
XMLHeaderText	XML Header Text	String	Input	Yes
XMLHierarchy	XML Hierarchy	Hierarchy	Input	No

Table 16.   Method Arguments for PropSetToXMLPropSet Method

Name	Default Display Name	Data Type	Type	Optional
XMLHierarchy	<IgnoreEmptyTag>	Hierarchy	Input/Output	No

Table 17.   Method Arguments for XMLToPropSet Method

Name	Default Display Name	Data Type	Type	Optional
<Value>	XML Document	String	Input	No
EscapeName	Escape Name	String	Input	Yes
ExternalEntityDirectory	External Entity Directory	String	Input	Yes
IgnoreCharSetConvErrors	Ignore Character String Set Conversion Errors	String	Input	Yes
ValidateExternalEntity	Validate External Entity	String	Input	Yes
XMLCharEncoding	XML Character Encoding	String	Output	Yes
XMLHierarchy	XML Hierarchy	Hierarchy	Output	Yes

Table 18.   Method Arguments for XMLPropSetToPropSet Method

Name	Default Display Name	Data Type	Type	Optional
ContainsInlineAttachments	Contains Inline Attachments	String	Input	Yes
TruncateFieldValues	Truncate Field Values	String	Input	Yes
XMLHierarchy	XML Hierarchy	Hierarchy	Input/Output	No

Table 19.   Method Arguments for ErrorHandler Method

Name	Default Display Name	Data Type	Type	Optional
GeneralErrorMessageText	General Error Message Text	String	Input	Yes
ErrorMessageText	Error Message Text	String	Input	No
ErrorCode	Error Code	String	Input	No
XMLHierarchy	Err XML Hierarchy	Hierarchy	Input/Output	No

FINS IAA-XML Dispatcher

The FINS IAA-XML Dispatcher is responsible for receiving and dispatching inbound messages. It receives the inbound message and scans the message for any commands specified in its rule sets dispatcher map and associate the integration objects for the connector components. It then parses the envelope of the message and converts it to the header property set and passes both, the integration objects for the specified action, the envelope layer property set and the XML message to the connector components for further processing. Table 20 lists all the user properties for FINS IAA-XML Dispatcher.

Table 20.   FINS IAA-XML Dispatcher User Properties

Name	Value	Description
DispatcherMapName	<integration object name>	Name of an integration object that details the dispatching rules and syntax for the IAA-XML standard. This map is usually created along with all the other integration objects by the wizard. The default map name is "IAADispMap".

Table 20.   FINS IAA-XML Dispatcher User Properties

Name	Value	Description
XMLEnvIntObjectName	<integration object name>	Name of an integration object that defines the content and hierarchy for the envelope and header section of IAA-XML. The default envelope integration object is "IAA Envelope".
XMLFaultObject_O	<path to fault section or element>	This allows dispatcher to identify a fault section with the first token and further confirm it with the value of second token if applicable. Extra fault objects can be added by incrementing the name with _1, _2, and so on. An example for the value for this user property name is XMLFaultObject_1 and value is //IOI/@cmdstatus;fail.

The IAA-XML Dispatcher business service provides ready-to-use methods and method arguments described in Table 21 and Table 22.

Table 21.   FINS IAA-XML Dispatcher Methods

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

For each method defined the FINS IAA-XML Dispatcher business service takes input arguments and produces output arguments. These arguments are described in Table 22 and Table 23.

Table 22.   FINS IAA-XML Dispatcher Method Arguments

Argument	Default	Description
XML Hierarchy	N/A	Property Set in XML hierarchy.

Table 23.   Method Arguments for DispatchMessage Method

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

Transport Adapter

Transport Adapter is a pre-built business service providing interface between Siebel application and external applications. Transports allow Siebel Financial Services applications to exchange data with external applications using standard technologies for both synchronous and asynchronous communication protocols.

Transports provide connectivity to virtually any communication protocol that can represent data as text or binary messages, including MQSeries from IBM, MSMQ from Microsoft, and HTTP.

NOTE: For details on transport adapter, refer to *Transports and Interfaces: Siebel Enterprise Application Integration*.

4 Configuration Roadmap

This chapter describes the types of business object models that the Siebel Connector for IAA-XML can be configured to support.

Siebel Connector for IAA-XML Configuration Overview

The Siebel Connector for IAA-XML comprises of four pre-built business services:

FINS IAA-XML Transaction Manager

FINS IAA-XML Data Transformation Engine

FINS IAA-XML Converter

FINS IAA-XML Dispatcher

The Siebel Connector for IAA-XML can be configured to support several types of IAA Business Object Model packages. Figure 3 illustrates the main steps in configuring the Siebel Connector for IAA-XML.
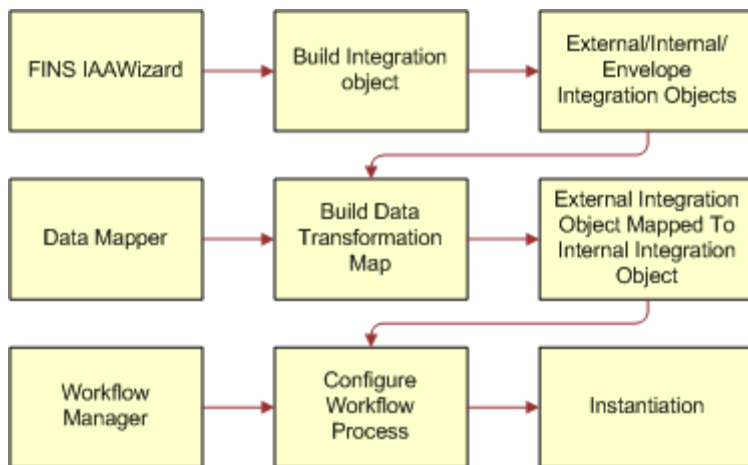

Figure 3.   Main Steps to Configure the Siebel Connector for IAA-XML

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

Check list	
	Install the MQSeries Financial Services Edition. *For details, see* "The MQSeries Financial Services Edition" on page 33*.*
	Creating the integration objects in Siebel Tools. *For details, see* "Creating Integration Objects in Siebel Tools" on page 40*.*
	Configuring the IAA-XML business service in Siebel Tools. *For details, see* "Configuring the IAA-XML Business Service" on page 50
	Configuring the transformation maps in Siebel Client. *For details, see* "Configuring the IAA-XML Business Service" on page 50
	Configuring events to trigger the workflow in real time using Runtime Event Manager (Inbound). *For details, see* "Configuring the IAA-XML Business Service" on page 50
	Configuring server tasks to dispatch the message to workflow (Outbound). *For details, see* "Configuring the IAA-XML Business Service" on page 50
	Configuring the Siebel Connector for IAA-XML for sending an IAA-XML message. *For details, see* "Configuring the IAA-XML Business Service" on page 50
	Configuring the Siebel Connector for IAA-XML for receiving an IAA-XML message. *For details, see* "Sample Workflow IAA Server Party Package Workflow" on page 60*.*
	Design your workflow. *For details, see* "Configuring the Data Transformation Maps" on page 52

To illustrate this procedure consider a scenario where Siebel Financial Services application sends a newly created contact to an external application for adding it to its database and also receives a newly generated contact record from an external applications to add to Siebel database. This illustration corresponds to the <AddPartyRequest> and <AddPartyResponse> commands in IAA-XML. We use the MQ Server Transport in this configuration roadmap but you can use other transports when applicable.

The MQSeries Financial Services Edition

IAA-XML is generic framework to support XML messaging in the financial services industry. It also supports the IBM MQSeries Financial Services Edition (MQSFSE) as shown in Figure 4.

NOTE: The MQSFSE is used as the default implementation through out this chapter. For details on MQSFSE, see IBM documentation.

Figure 4.   Integration Between Siebel Applications and MQSFSE

You can use the MQSFSE product to integrate Siebel applications with various legacy applications as shown in Figure 4. The adapters and connectors translate the application-specific information into the IAA-XML format understood by MQSFSE hub. The MQSFSE hub provides the routing, data manipulation, application cross-reference, publish or subscribe, logging, and security services required to make sure successful delivery of the appropriate IAA-XML message to the appropriate destinations. This IAA-XML message is then translated by the adapters, between the MQSFSE hub and the destination, into the application-specific data requirements.

In order to support the key feature of Cross Reference function, the Siebel Connector for IAA-XML is designed to generate the IAA-XML Message Header. An example of the header is shown below:

```
<Message id='1FEG6200108:13:16' version='1.4' bodyType='IAA-XML'

        timeStampCreated='2001-08-08 20:13:16'

        timeStampExpired='2001-11-08 20:13:16'

        sourceLogicalId='FrontEnd'

        authenticationId='SysAdmin'

        crfPublish='false'>

    <Default>

        <DefaultTime zoneOffset='00:00'/>

        <DefaultCurrency currencyCode='USD'/>

        <DefaultUnit measuredConcept='weight'>kg</DefaultUnit

    </Default>

    <CrfActionGroup bodyCategory='AddPartyRequest'>
```

```
        <CommandReference refid='cmd0'/>

        <KeyGroup id='R3' keyGroupType='PersonName'>

         <AlternateId value='12-F12V'

            sourceLogicalId='SIEBEL'

            attributeString = id number'

            state=''

        </KeyGroup>

        …

    </CrfActionGroup>

    <COMMAND>

        <AddPartyRequest id='xy' cmdType='request'>

         <Person>

            <KeyGroup refid = 'R3'/>

         …

         </Person>

        </AddPartyRequest>

    </COMMAND>

    …

</Message>
```

Certain values in the header are automatically generated by the connector, but you can specify the others in the envelope integration object and external integration objects. Table 24 describes all the elements and how they are configured. For system generated attributes, there is no configuration available.

NOTE: For details on IAA-XML, see IBM documentation.

Table 24. Elements of IAA-XML Message

Attribute	Description	Configuration
Message element structure in top level		
A container for Commands that can also contain elements such as Defaults, and CrfActionGroup. Valid elements are described below.		
id	Unique Id of the message. Every instance of every message has a unique Id.	System generated

Table 24. Elements of IAA-XML Message

Attribute	Description	Configuration
SessionId	Multiple Messages may be sent on the same topic OR a one message may start chain of messages. In both case, the sessionId associates these messages together. Associated messages have the same sessionId value. For example, the sessionId for the messages containing a Command and the one containing its Response are the same.	System generated
version	Version of IAA-XML architecture or message structure. This attribute applies only to IAA-XML if bodyType='IAA-XML'.	IAA Envelope Integration Object->Message Integration Component->Message_version Integration Component Field, XML Literal Value. The default is 1.4.
bodyType	Describes the architecture of the message content. Defined values: IAA-XML, EID.	IAA Envelope Integration Object->Message Integration Component->Message_version Integration Component Field, XML Literal Value. The default is IAA-XML.
timeStampCreated	The time the message was created in the source system. Format: yyyy-mm- ddThh: mm:ss.ffffff timestamp.	System generated
timeStampExpired	The time when the message expires.	IAA Envelope Integration Object->Message Integration Component->Message_timeStampExpired Integration Component Field, XML Literal Value.
sourceLogicalId	Logical Id of the system that issued the message.	IAA Envelope Integration Object->Message Integration Component->Message_sourceLogicalId Integration Component Field, XML Literal Value. The default is Siebel.
destinationLogicalId	Logical Id of the system that should receive the message. Typically this attribute is used for notification when the publish-subscribe mechanism is not used.	IAA Envelope Integration Object->Message Integration Component->Message_destinationLogicalId Integration Component Field, XML Literal Value. The default is BackEnd.

Table 24.   Elements of IAA-XML Message

Attribute	Description	Configuration
authenticationId	Id of the user that issued the message.	IAA Envelope Integration Object -> Message Integration Component -> Message_authenticationId Integration Component Field, XML Literal Value.
crfPublish	Indicates whether to publish the hub activity to subscribers and defines the default value for the crfPublish attribute within CrfActionGroup. Valid values are true or false.	IAA Envelope Integration Object->Message Integration Component->Message_crfPublish Integration Component Field, XML Literal Value. The default is false.
crfCmdMode	Defines response type that will be generated by the hub and defines the default value for the crfCmdMode attribute within CrfActionGroup. Valid values are alwaysRespond, neverRespond, onlyRespondInError.	IAA Envelope Integration Object->Message Integration Component->Message_crfCmdMode Integration Component Field, XML Literal Value The default is alwaysRespond.
txnScope	Valid values are all or any. 'All' means all commands in the Message are part of a transaction. 'Any' means that Commands are transaction independent.	IAA Envelope Integration Object->Message Integration Component->Message_txnScope Integration Component Field, XML Literal Value.
Default element structure in Message		
XML attributes listed below can appear many times in a message with the same value. Using the Defaults section allows these attributes to be predefined which reduces message size.		
DefaultTime element structure in Default		
Applies to type Timestamp.		
zoneOffset	Default time zone offset from UTC. Values must be between −720 and +720. Value is typically an exact number of hours (a multiple of 60) but the offset may also include half and quarter hours.	IAA Envelope Integration Object->DefaultTime Integration Component-> DefaultTime_zoneOffset Integration Component Field, XML Literal Value
DefaultCurrency element structure in Default		
Applies to type CurrencyAmount.		

Table 24.   Elements of IAA-XML Message

Attribute	Description	Configuration
currencyCode	Default currency code (for currency amounts not specifying a currency code) using the ISO-4217 values.	IAA Envelope Integration Object->DefaultCurrency Integration Component-> DefaultCurrency_currencyCode Integration Component Field, XML Literal Value
DefaultUnit element structure in Default Define default unit of measure. For example, kg for weight.		
measuredConcept	Measured concepts includes weight, distance, temperature, and so on.	IAA Envelope Integration Object->DefaultUnit Integration Component-> DefaultUnit_measuredConcept Integration Component Field, XML Literal Value
CrfActionGroup element structure in Message Grouping of hub activity related to a command section. This must be defined for each command contained in the message and is required if the hub processing the message.		
bodyCategory	Name of Workflow script that the adapter wants used to process a message.	any external integration object->Integration Object User Property->Name:CrfActionGroup_bodyCategory, Value:<specify the values here>.
crfPublish	Determines whether or not the messages need to be published specific to this action. Valid values are true or false.	IAA Envelope Integration Object->CrfActionGroup Integration Component-> CrfActionGroup _crfPublish Integration Component Field, XML Literal Value. The default is false.
crfCmdMode	Type of required response for action group. Valid values are alwaysRespond, neverRespond, or onlyRespondInError. This attribute is a specialized version of cmdType attribute defined for IAA-XML commands.	IAA Envelope Integration Object->CrfActionGroup Integration Component-> CrfActionGroup _crfCmdMode Integration Component Field, XML Literal Value The default is alwaysRespond.

Table 24.   Elements of IAA-XML Message

Attribute	Description	Configuration
destinationLogicalId	Logical Id of destination system receiving the message specific to this action group.	any external integration object->Integration Object User Property->Name:CrfActionGroup_destination LogicalId, Value:<specify the values here>.
CommandReference element structure in CrfActionGroup		
Refers to a (command) in the core (business content) section of the message.		
refid	Reference to a COMMAND element corresponding to this action group.	System generated

Table 24. Elements of IAA-XML Message

Attribute	Description	Configuration
KeyGroup element structure in CrfAction CrfActionGroup		
Defined for each aggregate contained in a particular command. For the MQSFSE hub, the message header contains the KeyGroup information instead of the aggregate. The aggregate refers to its KeyGroup using refid. When there are multiple keys, multiple alternate ids will show within KeyGroup. It assumes all keys are related by UUID.		
id	Identifier for referencing key blocks within a message.	System generated
refid	Reference to key block (used at aggregate level to reference their keys in the header).	System generated
keyGroupType	Type of key. Corresponds to aggregate type.	System generated
UUID element structure in KeyGroup		
	Universal unique identifier for accessing all keys for all systems corresponding to this entry.	System generated
AlternateID element structure in KeyGroup		
Alternative or system specific key used by a system to identify an aggregate.		
value	Key value stored in a particular system (if different from UUID).	System generated
sourceLogicalId	Id of the system owning this key.	IAA Envelope Integration Object->AlternateId Integration Component-> AlternateId_sourceLogicalId Integration Component Field, XML Literal Value. The default is Siebel.
attributeString	Set of strings that some systems may require to be held in the hub cross reference file (CRF). Valid separator is comma.	IAA Envelope Integration Object->AlternateId Integration Component-> AlternateId_attributeString Integration Component Field, XML Literal Value.

Table 24.   Elements of IAA-XML Message

Attribute	Description	Configuration
state	Action on the key. **referenced.** Key is information sent from one system to another and that CRF should not have an entry for it. **add.** Key was created in a system and needs to be added to CRF, only for message into hub. **added.** Key was created in the originating system and in the CRF, only for message coming from hub. **exists.** Key is in the CRF & should be translated for another system. **modify.** Key was modified in the originating system and needs to be modified in the CRF, only for message into hub. **modified.** Key was modified in the originating system and in the CRF, only for message coming from hub. **delete.** The key needs to be removed from the CRF, only for message into hub	any external integration object->Integration Object User Property-> Name:AlternatedId_state, Value:<specify the values here>.
newValue	Value to which the key should be modified. If a newValue is present, then the atttributeString attribute (if present) contains the new attributeString.	Not Supported

Creating Integration Objects in Siebel Tools

In order to use the Siebel Connector for IAA-XML you need to build three different types of integration objects:

IAA Envelope Integration Object. For details, see "To Create Integration Object" on page 41.

IAA External Integration Objects. For details, see "To create IAA external integration objects" on page 44.

IAA Internal Integration Objects. For details, see "Creating IAA Internal Integration Objects" on page 46.

You use the Integration Object Builder in Siebel Tools to create these integration objects. The Builder first walks you through steps for selecting your project, business service, and .DTD file and then provides you with the option of creating all three types of integration objects or just the ones you need.

NOTE: All the integration objects used in the following procedures are included in the Sample database.

To Create Integration Object

1 Start Siebel Tools.

2 Lock the appropriate project.

3 Choose File > New Object… to display the New Object Wizards dialog.
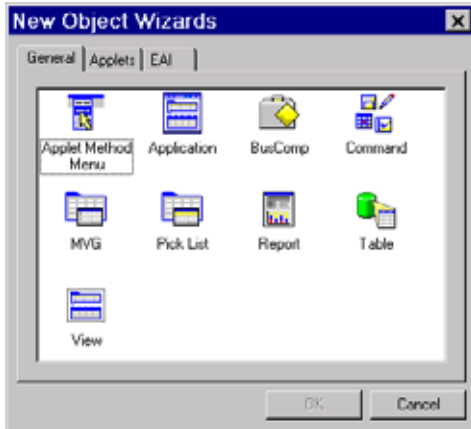
4 Select the EAI tab and double click on the Integration Object icon.

The Integration Object Builder appears.

5 Fill in the two fields in the dialog box:

   a Select the project you locked in Step 2 on page 41.

   b Select the FINS IAA Wizard Service as your source system of the integration object.

6 Click Next.

The FINS IAA Wizard Service dialog box appears.
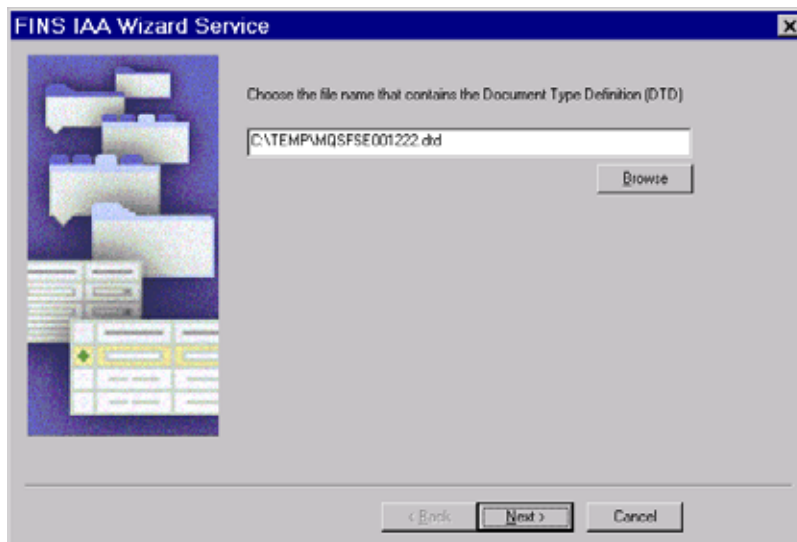
7 Select the .DTD file you want the wizard to use and click Next to build the envelope integration object. "To Create IAA envelope Integration Objects" on page 42.
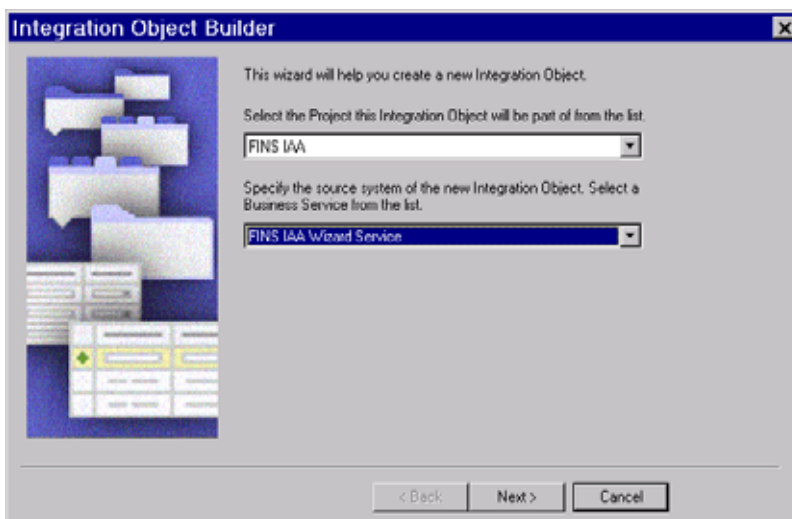
   NOTE: The .DTD file you use to create your integration objects has to comply with IAA-XML standard.

Creating IAA Envelope Integration Objects

The envelope integration object is not required for all integration processes. If your integration process does not require envelope integration object you can skip this section by clicking Next. Otherwise you can build the envelope integration object using the following procedure.

To Create IAA envelope Integration Objects

1 Determine whether or not you only need envelope integration object and select the appropriate option on the FINS IAA Wizard Service dialog box.

2 Provide a meaningful name for the envelope integration object and click Next.

NOTE: It is recommended to use Message as you Envelope Root Tag unless you have customized your .DTD file.
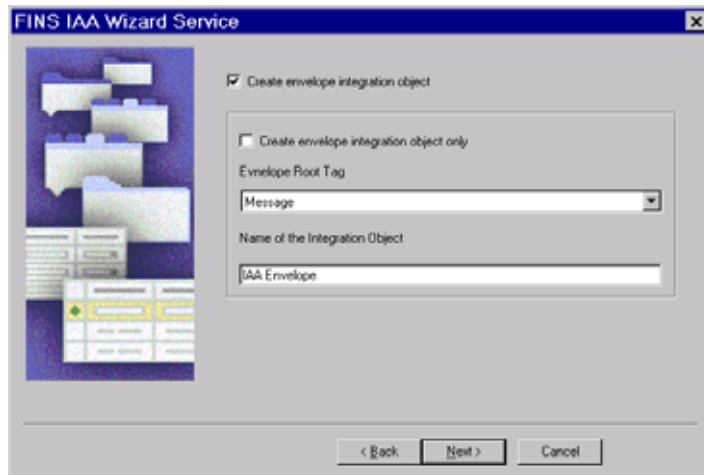
The next dialog box of the FINS IAA Wizard Service appears.
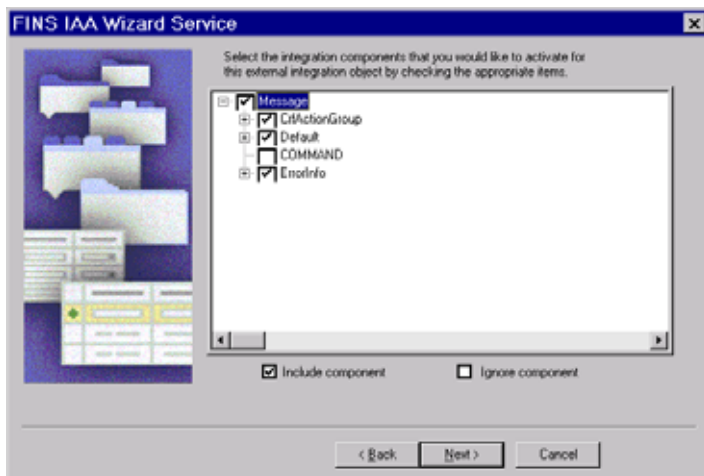
3 Select the components you would like to include in your integration object and click Next.

The next dialog box for building external integration object appears. For details, see "To create IAA external integration objects" on page 44.

NOTE: Any component that has a plus sign (+) next to it is a parent in a parent-child relationship with one or more child components. If you deselect the parent component, the children below that component are deselected as well. You cannot include a child component without also including the parent. The Integration Object Builder enforces this rule by automatically selecting the parent of any child you choose to include. You should only select the business components required for your business processes to avoid unnecessary performance deterioration.

Creating IAA External Integration Objects

The external integration object maps the internal integration object components and fields to IAA-XML aggregations and fields. The external integration object is used by both the FINS IAA-XML Data Transformation Engine and FINS IAA-XML Converter to construct the IAA-XML aggregations and format. To build the external integration object you use the Integration Object Builder.

To create IAA external integration objects

1 Create envelope integration object. For details, see "To Create Integration Object" on page 41.

   After you create you envelope integration object and click Next the following dialog box of FINS IAA Wizard Service appears.

2 Pick the appropriate Request and Respond Command from each picklist and give unique and meaningful names to your external request and respond integration objects.

   You may encounter an error similar to the following.
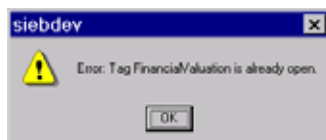
   This is a benign error, due to a recursive element references in your .DTD document, and can be ignored.

3   Click Next to move to the next dialog box to select the IAA aggregates for your external request integration object.
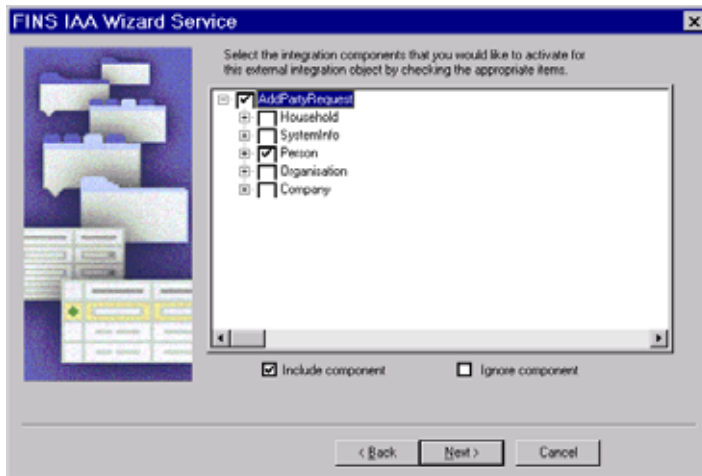
NOTE: Any component that has a plus sign (+) next to it is a parent in a parent-child relationship with one or more child components. If you deselect the parent component, the children below that component are deselected as well. You cannot include a child component without also including the parent. The Integration Object Builder enforces this rule by automatically selecting the parent of any child you choose to include. You should only select the business components required for your business processes to avoid unnecessary performance deterioration.
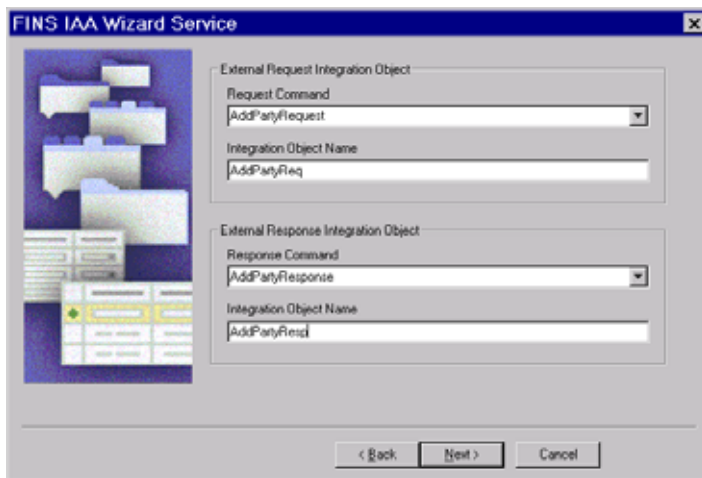
4   Click Next to move to next screen to select the IAA aggregates for your external response integration object.
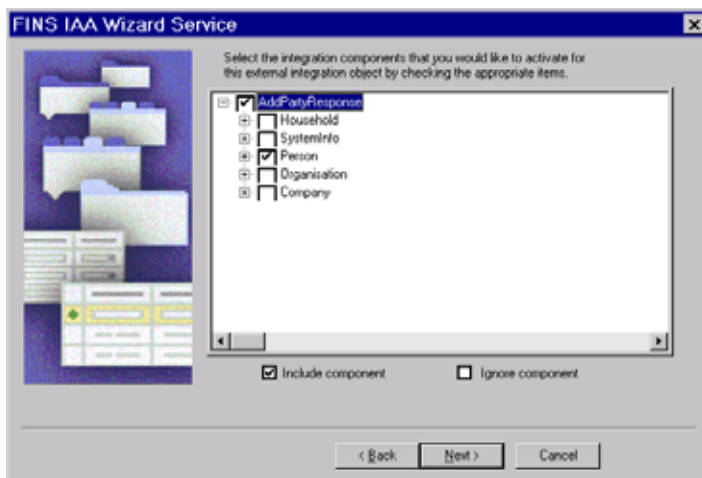
5   Click Next to move to the next screen to build your internal integration object as described in .

Creating IAA Internal Integration Objects

The internal integration object has the hierarchy of Siebel business object and business components as well as the fields. It maps to the objects in Siebel application and is created using Integration Object Builder in Siebel Tools. This internal integration object is required in order to facilitate the FINS IAA-XML Transaction Manager to package the data gathered. This service will gather the data needed for a particular message and package it into the hierarchy defined in the internal integration object.

To create IAA Internal Integration Objects

1 Create envelope and external integration objects. For details, see "To Create Integration Object" on page 41 and "To create IAA external integration objects" on page 44.

After you build these integration objects and click Next on the last dialog box, the dialog box for building internal integration object appears.
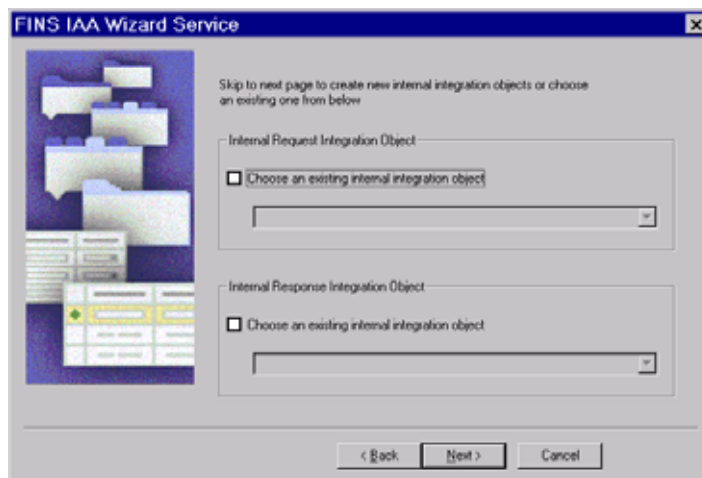
You can either select an existing integration object from the picklist on the FINS IAA Wizard Service or click Next to build a new internal integration object.

2   Pick the appropriate Business Object to build your internal request integration object and give a unique and meaningful name to your internal integration object.
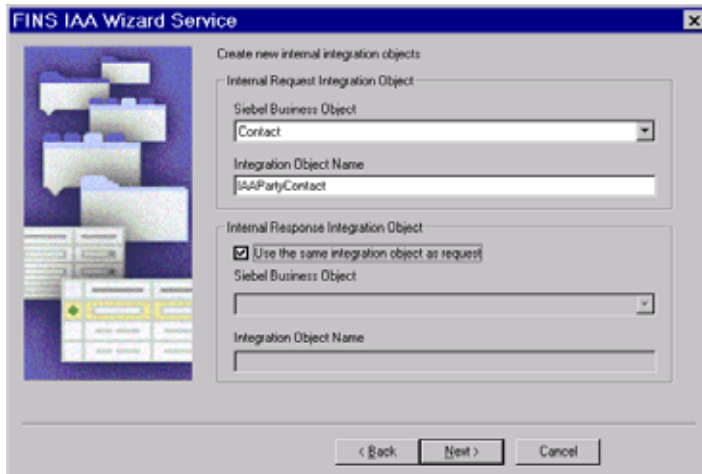
3   For the internal response integration object you can either use the same internal request integration object or have the wizard build you a separate integration object by choosing a business object from the picklist and providing a meaningful name for you internal response integration object. In this illustration the same internal integration object is used.

4   Click Next to move to the next dialog box to select the components for your integration object.
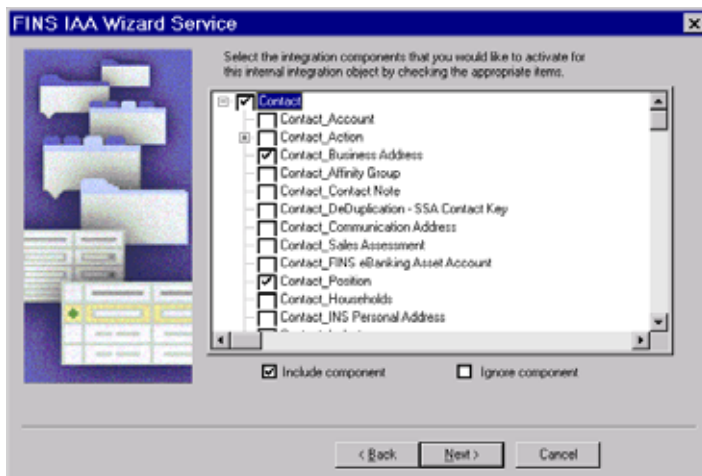
5 Deselect any of the component listed on this dialog box to ignore that component. If you do not include a component, you will be unable to integrate data for that component between the Siebel Financial Services applications and another system.

NOTE: Any component that has a plus sign (+) next to it is a parent in a parent-child relationship with one or more child components. If you deselect the parent component, the children below that component are deselected as well. You cannot include a child component without also including the parent. The Integration Object Builder enforces this rule by automatically selecting the parent of any child you choose to include.

6 Click Next for the wizard to build the integration object.

7 Click Yes on the message your receive to confirm your desire to build this integration object.

The wizard start building the integration object and then displays error or warning messages generated during the process. Review the messages and take the appropriate action to address them.

8 Click Finish to complete the process of creating a new Siebel integration object.

9 Your new integration objects appear in the list of integration objects in Siebel Tools.

IAA Dispatcher Map

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

IAADispMap_<currently locked project name>

To view the Dispatcher Map user properties

1 From Siebel Tools, choose Object Explorer > Integration Object.

2 Query for the dispatcher map name. For example. IAADispMap.

3 Navigate to the user properties of the dispatcher map to see its user properties.

The first two rows in the User Properties applet are the rule sets created by the wizard for AddParty scenario.

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, Message/COMMAND/ AddPartyRequest/Person, is the path the dispatcher try to locate in the message received and if it finds the match then it uses the information in the value column, Message/COMMAND/ AddPartyRequest;AddPartyRequest_ERqIRqMapIn;AddPartyRequest_IRsERsMapOut;AddPartyRe q;IAAPartyContact;IXMLOperation_ADD, to determine the values it needs to insert as follow:

Each value is made up of six tokens that are separated by "; " and each token represents a specific information.

- First token is the location to insert the remaining 5 tokens at runtime. For example, Message/ COMMAND/AddPartyRequest.

- Second token is the name of data transformation map for mapping external request integration object indicated by ERq to internal request integration object indicated by IRq. For example, AddPartyRequest_ERqIRqMapIn.

- Third token is the name of data transformation map for mapping internal response integration object IRs to external response integration object ERs. For example, AddPartyRequest_IRsERsMapOut.

- Forth token is the external request integration object. For example, AddPartyReq.

- Fifth token is the internal response integration object. For example, IAAPartyContact.

- Sixth token is the operation corresponding to <AddPartyRequest> command. For example, IXMLOperation_ADD Which is configured in FINS IAA Wizard user properties. This token will be used as key to operation for FINS IAA-XML Transaction Manager.

The data transformation map names must be used when configuring the transformation maps. For details, see "Configuring the Data Transformation Maps" on page 52. 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: Compile all the integration objects and the dispatcher map created by the wizard into your .srf. Make sure you migrate your new integration objects to the same database used by your client. You also need to copy your newly compiled .srf to the correct object directory of the same server used by your clients.

Configuring the IAA-XML Business Service

Siebel Financial Services provides an object called a business service, which you can reuse in multiple applications. The Siebel Connector for IAA-XML provides the following pre-built business services which you can configure to meet your business requirements:

FINS IAA-XML Transaction Manager

FINS IAA-XML Data Transformation Engine

FINS IAA-XML Converter

FINS IAA-XML Dispatcher

You can configure business services by manipulating their user properties or you can create your own business service in Siebel Tools.

NOTE: Once you have configured your business services to accomplish the tasks required for your business scenario you need to compile the business service to include the new information in your .srf file and copy the .srf file to the directory where your clients can access.

FINS IAA-XML Transaction Manager

You can extend this business service by manipulating its Operation user property as follow:

The basic format for value entry for each operation is as follow:

IAAOperation_Query
Service/Method/Argument;Argument; or /Method/Argument;Argument;

Service, Method, and Argument are separated by "/".

Each Argument ends with ";".

IAAOperation_XMLQuery
EAI Siebel Adapter/Query/#XMLHierarchy;

The default Service name is "EAI Siebel Adapter" and the default argument name is "SiebelMessage."

!SiebelMessage means turning off the SiebelMessage.

#XMLHierarchy means replacing SiebelMessage with XMLHierarchy.

IAAOperation_ADD_ROLL_BACK
EAI Siebel Adapter/Delete/RollbackOnSame;

RollbackOnSame means that the current operation is used for rollback operation, and the rollback operation is performed based on the incoming instance. The default value is the instance, which was stored in memory before the request message was sent.

IAAOperation_GetValue

FINs Industry/BC Facility Service/HierarchySearchSpec/!SiebelMessage;A=>B;

A=>B means getting argument value of "A" from argument value of "B" where argument "B" is Connector Integration Object Instance's arguments.

Table 25 lists all the user properties examples for the FINS IAA-XML Transaction Manager.

Table 25.   User Properties Examples for the FINS IAA-XML Transaction Manager

Name	Value
IXMLOperation_ADD	EAI Siebel Adapter/Upsert/
IXMLOperation_ADD_ROLL_BACK	EAI Siebel Adapter/Delete/RollbackOnSame;
IXMLOperation_DEL	EAI Siebel Adapter/Delete/
IXMLOperation_FIND	FINS Industry BC Facility Service/HierarchySearchSpec/!SiebelMessage;IntObjectName=>SiebelFINSRespIntObjName;
IXMLOperation_MOD	EAI Siebel Adapter/Synchronize/
IXMLOperation_MOD_ROLL_BACK	EAI Siebel Adapter/Synchronize/
IXMLOperation_QUERY	EAI Siebel Adapter/Query/
IXMLOperation_QUERY_BY_ID	EAI Siebel Adapter/Query/PrimaryRowId;!SiebelMessage;

FINS IAA-XML Data Transformation Engine

You can configure this business service using the IgnoreEmptyTag integration map argument. You can access this argument when configuring the data transformation maps in the Data Maps screen. When this argument is available, the FINS IAA-XML Data Transformation Engine can remove the empty-value integration component fields in the integration object received.

FINS IAA-XML Converter

Table 26 displays the only user property you can configure for this business service. This value will appear in the pre-header section of your IAA-XML message.

Table 26.   User Properties for Converter

Name	Value
XMLEnvIntObjectName	Name of the envelope integration object created by the wizard.

Besides, the FINS IAA-XML converter uses the IAA-XML hierarchy represented in the external integration object to guide the message through the converting process. If there are elements in the external integration object instance received that do not have a definition defined in the integration object definition, the converter errors out. If you expect such a situation, you can set the Ignore Undefined XML Tag parameter on the user property of the corresponding integration object.

NOTE: This user property is created by wizard and is set to Y. You can turn it off if you wish the converter to error out.

FINS IAA-XML Dispatcher

You can modify both user properties for this business service as shown in Table 27.

Table 27.   User Properties for Converter

Name	Value
DispatcherMapName	Name of the dispatcher map created by the wizard.
XMLEnvIntObjectName	Name of the envelope integration object created by the wizard.

Configuring the Data Transformation Maps

One of the steps in configuring the integration objects for your integration process is to map the components and fields in your internal integration object to the message elements in the external integration object. These maps will be used by the FINS IAA-XML Data Transformation Engine to move the data from one integration object to another. For each integration process you need to configure two maps for inbound and two maps for outbound process. These map names are in the user properties entry of the dispatcher map integration object. The map names are created by the FINS IAA Wizard when it creates the integration objects.

Following four maps are the maps you need to configure for the illustration:

AddPartyRequest_ERqIRqMapIn (server receiving a request)

AddPartyRequest_IRsERsMapOut (server sending out response)

AddPartyResponse_IRqERqMapOut (client sending out a request)

AddPartyResponse_ERsIRsMapIn (client receiving response)

If you wish, you can change the map name in the user properties of the dispatcher map after configuring the data transformation map.

The following procedure walks you through configuring the AddPartyRequest_ERqIRqMapIn data transformation map which is used for receiving an inbound request.

To configure the DTE map

1 Start Siebel Client.

2 Navigate to the Administration - Integration screen > Data Map Editor > Integration Object Map view.

The Integration Object Map applet appears.

3 Create a new record in this applet.

Name. This name must be the same as the data transformation map name created by the wizard and stored in the dispatcher map list. For example, AddPartyRequest_ERqIRqMapIn where ERq indicates external request integration object and IRq indicates internal integration object.

4 Select the source integration object and the target integration objects for the mapping. Keep the following definitions in mind:

Source Object. For an outbound request or inbound response message, the source object is the internal integration object. For an outbound response or inbound request message, the source object is the external integration object. For example, AddPartyReq is the source object for inbound request.

Target Object. For an outbound request or inbound response message, the target object is the external integration object. For an outbound response or inbound request message, the target object is the internal integration object. Fro example, IAAPartyContact is the target object for inbound request.

5 Map the source components and the target components.

6   Map fields to fields.

NOTE: For details on creating data maps, see *Business Processes and Rules: Siebel Enterprise Application Integration.*

Configuring Runtime Event Manager

Oracle supports triggering workflows based on runtime events such as Write Record, which gets triggered whenever a record is written. Using a runtime event allows you to incorporate configured workflow functions into actual applications.

NOTE: For details, see *Siebel Events Management Guide.*

The example in this section describes a runtime event that triggers a workflow when a new contact record is added by users through Siebel Financial Services Application. In the following procedure, a runtime event as well as the corresponding action is defined.

To create a runtime event

1   Start Siebel client.

2   Navigate to the Administration - Runtime Events screen > Events > Events view.

3   Add a new record and set the following fields:

   Sequence. 1

   Object Type. BusComp

   Object Name. Contact

   Events. WriteRecord

   Conditional Expression. LoginName='SADMIN'

4   Select the Action Set Screen from the dropdown menu.

5   Create a new Action Set record and set the following fields:

   Name. Trigger Workflow.

   Action Type. BusService

   Sequence. 1

6   In the More Info applet you can alter the following values to meet you business requirements:

Business Service Name. Workflow Process Manager

Business Service Method. RunProcess

Business Service Context. "ProcessName", "IAA Add Party outbound Workflow"

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 for one or more server task to run. The server task you need to run for your Siebel Connector for IAA-XML is Siebel EAI MQSeries Server Transport. The Siebel EAI MQSeries Server Transport is designed to provide a messaging solution to help you integrate data between Siebel Business Applications and external applications that can interface with the IBM MQSeries. The EAI 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 subsystem

1 Start Siebel client.

2 Navigate to the Administration - Server Configuration screen > Profile Configuration > 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, IAAMQConnSubsys.

    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 following parameters:

    MqPhysicalQueueName -<Queue name to receive inbound request message from>

    MqQueueManagerName- <Queue manager name who own the queues>

    MqRespPhysicalQueueName- <Queue name to send response message to>

    MqSleepTime- 100 <or longer if needed>

6 Save the record.

7 Create another Named Subsystem with following name and parameters.

    Name- <Any name>. For example, IAAMQDataSubsys

    Type- EAITransportDataHandlingSubsys

NOTE: The subsystem type that you select should have a checkmark in the Is Named Enabled field.

8 Save the record.

9   In the Enterprise Profile Configuration list, modify following parameters:

> **DispatchWorkflowProcess-** IAA Server Party Package Workflow

10 Save the record.

After creating and configuring your Named Subsystem you need to configure the MQSeries Receiver.

To configure MQSeries Receiver parameters

1   Start Siebel client.

2   Navigate to the Administration - Server Configuration screen > Enterprises > Component Definitions view.

3   Query for MQSeries Server Receiver and set the following:

> **Receiver Connection Subsystem-** Name from . For example, IAAMQConnSubsys.
>
> **Receiver Data Handling Subsystem-** Name from . For example, IAAMQDataSubsys.
>
> **Receiver Method Name-** ReceiveDispatch or ReceiveDispatchSend.
>
> **Default Tasks-** 1 or number of tasks desired.

4   Restart the Siebel Server and make sure 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*.

Configuring Outbound and Inbound Integration

Oracle's Siebel Business Applications provide several sample workflows as reference implementations in the Sample database that can be either used as is or you can modify them based on your business requirement before using them in your environment. The following sections walks you through two of these sample workflow.

NOTE: For details on Siebel Workflow, see *Siebel Business Process Framework: Workflow Guide.*

Sample Workflow IAA Add Party Outbound Workflow

This is a sample workflow process that gets Contact information from database and converts it into IAA-XML AddPartyRequest format. This IAA-XML message is then sent to MQ Series. Other Applications will be able to retrieve the message from MQ.

Figure 5 shows the graphical representation of the IAA Add Party Outbound workflow.

Figure 5.   IAA Add Party Outbound Workflow

Process Properties

The IAA Add Party Outbound Workflow process has the following properties.

Name	Data Type
Converter Request Prop Set	Hierarchy
DTE Request Prop Set	Hierarchy
Dispatch Key	String
Dispatch Map Name	String
Error Code	String
Error Message	String
IAA Request Msg	String
Object Id	String
Output File Name	String
Process Instance Id	String
Siebel Operation Object Id	String
SiebelOperation	String
Transaction Manager Request Prop Set	Hierarchy
XML Response Prop Set	Hierarchy

NOTE: You set workflow process properties when you need a property that will be global to the entire workflow.

The first step, after Start, uses the FINS IAA-XML Transaction Manager to query a party from your Siebel application, using the following input and output arguments.

Input Arguments

Input Argument	Type	Value	Property Name	Property Data Type
DispatcherMapName	Literal	IAADispMap		
IXML Map Path	Process Property		Dispatch Key	String
Primary Row Id	Process Property		Object Id	String
Outbound Operation	Literal	IXMLOperation_QUERY_BY_ID		

Output Arguments

Property Name	Type	Output Argument
Transaction Manager Request Prop Set	Output Argument	XML Hierarchy

The second business service transforms the information using the FINS IAA-XML Data Transformation Engine business service with the Transform To External Hierarchy and following input and output arguments.

Input Arguments

Input Arguments	Type	Property Name	Property Data Type
XML Property Set	Process Property	Transaction Manager Request Prop Set	Hierarchy

Output Arguments

Property Name	Type	Output Argument
DTE Request Prop Set	Output Argument	XML Property Set

The third business service invokes the FINS IAA-XML Converter with the PropSetToXMLPropSet method to convert the IAA-XML Hierarchy to an IAA-XML String with the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Property Name	Property Data Type
XML Hierarchy	Process Property	DTE Request Prop Set	Hierarchy

Output Arguments

Property Name	Type	Output Argument
Converter Request Prop Set	Output Argument	XML Hierarchy

The result of this step is stored in output argument Converter Request Prop Set.

As the last step of the workflow, the forth business service of the workflow invokes the XML Hierarchy Converter with the XML Hierarchy to XML Document method to generate the outgoing XML document using the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Value	Property Name	Property Data Type
EscapeNames	Literal	false		
XMLHierarchy	Process Property		Converter Request Prop Set	Hierarchy

Output Arguments

Property Name	Type	Output Argument
IAA Request Msg	Output Argument	<Value>

Sample Workflow IAA Server Party Package Workflow

This is a sample workflow process that handles an inbound IAA Party Packages from external application. Figure 6 shows the graphical representation of the IAA Server Party Package Workflow.
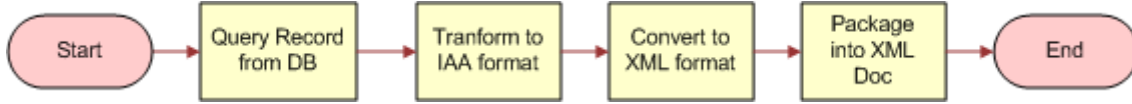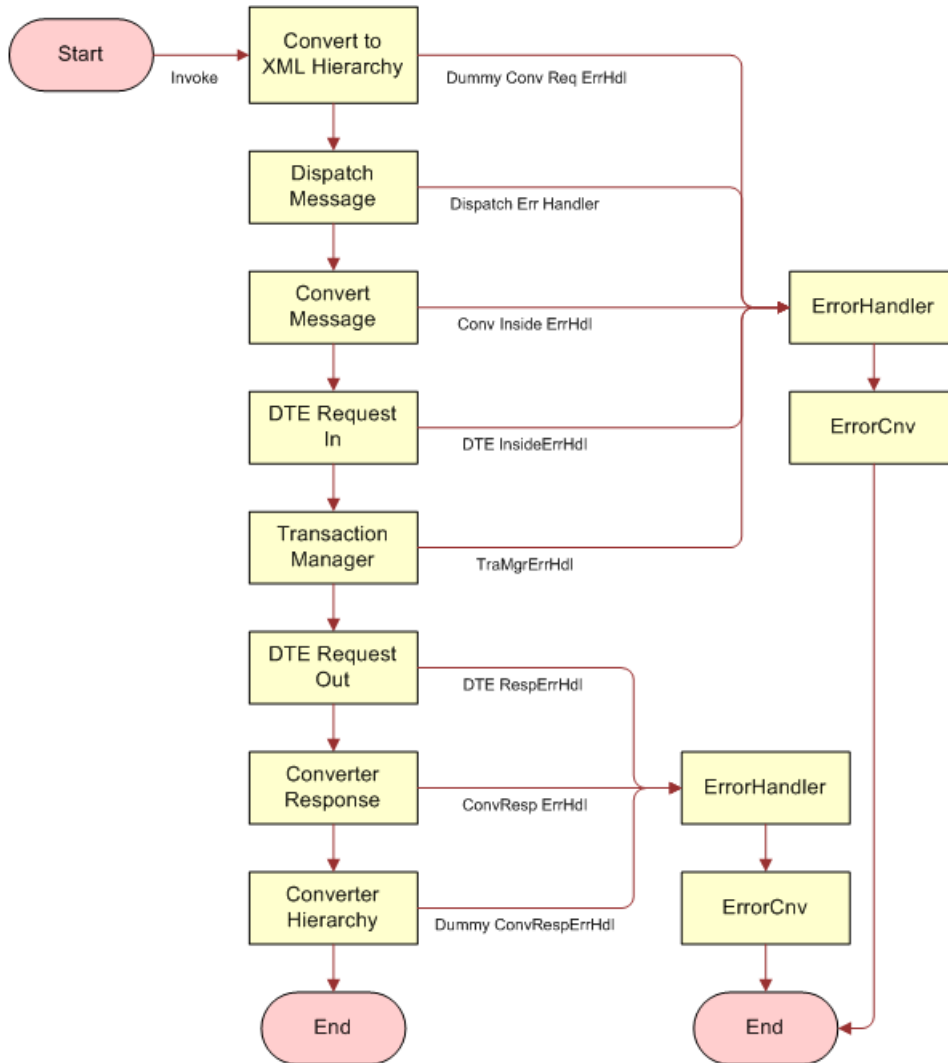

Figure 6.   IAA Server Party Package Workflow

Process properties

The IAA Add Party Outbound Workflow process has the following properties.

Name	Data Type	Default String
<Value>	String	
Cnv Req Prop	Hierarchy	
Conv Resp Prop	Hierarchy	
Core Req Prop	Hierarchy	
DTE Req Prop	Hierarchy	
DTE Resp Prop	Hierarchy	
DTE Store Prop	Hierarchy	
Disp Req Prop	Hierarchy	
Err Hierarchy Prop	Hierarchy	
Error Code	String	
Error Message	String	
Inbound Request	String	<Value>
Inbound Result	String	
Input File Name	String	C:\DeleteOutbound.xml
Object Id	String	
OnlyIOI	String	true
Output File Name	String	C:\DeleteInbound.xml
Process Instance Id	String	
Siebel Operation Object Id	String	
Tran Req Prop	Hierarchy	
Trans Code	String	
Trans Error Message	String	
Trans Resp Prop	Hierarchy	

NOTE: You set workflow process properties when you need a property that will be global to the entire workflow. Also note that name <Value> for the first process property and the default string for the Inbound Request process property has to be typed in the exact form shown here in order to use MQSeries Receiver server component. Siebel Workflow is case-sensitive, and <Value> is a different property than <value>.

The first step, after Start, uses the XML Hierarchy Converter with the XML Document to XML Hierarchy to convert the XML message received to an XML hierarchy, using the following input and output arguments.

Input Arguments

Input Argument	Type	Value	Property Name	Property Data Type
XML Document	Process Property		Inbound Request	String
Escape Name	Literal	false		

Output Arguments

Property Name	Type	Output Argument
Core Req Prop	Output Argument	XML Hierarchy

The second business service dispatches the message using the FINS IAA-XML Dispatcher business service with the Dispatch Message method and following input and output arguments.L

Input Arguments

Input Arguments	Type	Value	Property Name	Property Data Type
DispatcherMapName	Literal	IAADispMap		
XMLEnvIntObjectName	Literal	IAA Envelope		
XML Property Set	Process Property		Core Req Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
Disp Req Prop	Output Argument	XML Property Set

The third business service invokes the FINS IAA-XML Converter with the XMLPropSetToPropSet method to convert the IAA-XML Hierarchy to property sets with the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Value	Property Name	Property Data Type
XMLEnvIntObjectName	Literal	IAA Envelope		
XML Hierarchy	Process Property		Disp Req Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
Cnv Req Prop	Output Argument	XML Hierarchy

Next step, the forth business service of the workflow invokes the FINS IAA-XML Data Transformation Engine with the Transform To Siebel Hierarchy method to transform the message to Siebel hierarchy using the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Property Name	Property Data Type
XML Property Set	Process Property	Cnv Req Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
DTE Req Prop	Output Argument	XML Property Set

The fifth business service of the workflow invokes the FINS IAA-XML Transaction Manager with the Execute Transaction method and following input and output arguments.

Input Arguments

Input Arguments	Type	Value	Property Name	Property Data Type
Only produce Integration Object Instance	Process Property		OnlyIOI	String
StatusObject	Literal	true		
XML Property Set	Process Property		DTE Req Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
Trans Code	Output Argument	ErrMsgId
Trans Error Message	Output Argument	ErrText
Trans Resp Prop	Output Argument	XML Property Set

Then the workflow invokes the FINS IAA-XML Data Transformation Engine with the Transform To External Hierarchy method to transform the message to external hierarchy using the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Property Name	Property Data Type
IAA DTE Property Set	Process Property	DTE Store Prop	Hierarchy
XML Property Set	Process Property	Trans Resp Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
DTE Resp Prop	Output Argument	XML Property Set

Once the message is in the external hierarchy, the workflow invokes the FINS IAA-XML Converter with the PropSetToXMLPropSet method to convert the message to XML hierarchy using the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Value	Property Name	Property Data Type
XMLEnvIntObjectName	Literal	IAA Envelope		
XML Hierarchy	Process Property		DTE Resp Prop	Hierarchy

Output Arguments

Property Name	Type	Output Argument
Conv Resp Prop	Output Argument	XML Hierarchy

As a last step, the workflow invokes the XML Hierarchy Converter with the XML Hierarchy to XML Document method to covert the XML hierarchy to the XML document using the input and output arguments shown below.

Input Arguments

Input Arguments	Type	Property Name	Property Data Type
XML Hierarchy	Process Property	Conv Resp Prop	Hierarchy

Output Arguments

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

The MQSeries Receiver component picks up the output from the converter and send it back to the queue.

If the workflow process encounters an error, it will stop processing and passes all process properties to the ErrorHandler to generate a meaningful error to notify the external system.

NOTE: For details on error handling process, see IAA Server Party Package Workflow in Sample database.

Sample IAA Party Package

The IAA Party Package used through out this chapter is included in the Sample database of Siebel Financial Services application for reference implementation. The package contains all the integration objects, data transformation maps, and workflows needed.

The "FINS IAA Insurance Party Application" project, in Siebel Tools, includes all the integration objects. You can list all the transformation maps in Siebel Financial Services application with a query specification of "IAA*Party*" in the Comments field of the Integration Object Map screen. You can also list all the workflows in Siebel Financial Services application with a query specification of "IAA Connector" in the Group field of Workflow Processes screen.

The Party package supports the following message sets:

AddPartyRequest

AddPartyResponse

ModifyPartyRequest

ModifyPartyResponse

DeletePartyRequest

DeletePartyResponse

FindPartyRequest

FindPartyResponse

Sample IAA Policy Package

Although the IAA Party Package was used throughout the scenario provided in this chapter, you can use the IAA Policy Package instead. The IAA Policy Package is also included in the sample database of Siebel Financial Services application. The package contains all the integration objects, the data transformation maps, and the workflows needed for the scenario.

The "FINS IAA Insurance Application" project, in Oracle's Siebel Tools, includes all the integration objects. You can list all the transformation maps in the Siebel Financial Services application with a query specification of "IAA*Policy*" in the Comments field of the Integration Object Map screen. You can also list all the workflows in Oracle's Siebel Financial Services application with a query specification of "IAA Connector" in the Group field of Workflow Processes screen.

The Policy package supports the following message sets:

SubmitApplicationRequest <for AutoPolicy>

SubmitApplicationResponse <for AutoPolicy>

SubmitApplicationRequest <for HomePolicy>

SubmitApplicationResponse <for HomePolicy>

SubmitApplicationRequest <for LifePolicy>

SubmitApplicationResponse <for LifePolicy>

InquiryFinancialServicesAgreementsRequest <for AutoPolicy>

InquiryFinancialServicesAgreementsResponse <for AutoPolicy>

InquiryFinancialServicesAgreementsRequest <for HomePolicy>

InquiryFinancialServicesAgreementsResponse <for HomePolicy>

InquiryFinancialServicesAgreementsRequest <for LifePolicy>

InquiryFinancialServicesAgreementsResponse <for LifePolicy>

A Data Types

This appendix provides descriptions of all data types used by the Siebel Connector for IAA-XML.

Definition of Data Types

The data types used in IAA-XML are as follows:

String

A string of characters (optionally containing blanks) for which a maximum length can be specified. String indicates an element that allows character data up to a maximum number of characters. The number after the hyphen specifies the maximum number of characters. For example, S-12 specifies an element of characters with maximum length 12 characters. S-8 indicates an element with no maximum length. It is expected that character type elements may contain multibyte representations of characters in some implementations, depending on the allowable character sets.

Text

A string of characters (optionally containing blanks) for which a maximum length cannot realistically be fixed.

Binary

A finite sequence of binary octets. The definition consists of three logical elements: content type, binary data and binary data length.

The Binary data type is a compound type consisting of three logical elements as shown in the following table:

Tag	Type	Usage	Description
<contentType>	Enumeration	Optional	Specified in IETF RFC 2046.
<binLength>	Integer	Required	Identifies the size of the binary data in number of bytes.
<binData>	Raw Binary Data	Required	Binary data.

Boolean

A logical TRUE or FALSE condition.

Date

An indication of a particular day in the Gregorian calendar. Elements of data type Date (as shown in the following table), 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).

This data type must contain a 4-digit year, and may contain a month number and a day number.

Date			
Data Type	**Type**	**Usage**	**Description**
<year>	Short	Required	4-digit year value.
<month>	Short	Optional	Number of the represented month. Value must be within the range 1 through 12. Must be included if <Day> is included. If absent, the value is assumed to be 1 (January).
<day>	Short	Optional	Number of the represented day. Value must be within the range 1 through 31. If absent, the value is assumed to be 1.

Time

An indication of a particular time in a day expressed with a maximum precision of one microsecond. Elements of data type Time (as shown in the following table), 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 for each calendar date. Based on the IFX 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.

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.

You should always specify the time zone to avoid ambiguous communication between clients and servers. 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).

Time			
Data Type	**Type**	**Usage**	**Description**
<hour>	Short	Required	Number of the represented hour. Value must be within the range 0 through 23.
<minute>	Short	Optional	Number of the represented minute. Value must be within the range 0 through 59. Must be included if <Second> is included. If absent, the value defaults to 0.
<second>	Short	Optional	Number of the represented second. Value must be within the range 0 through 60. The value 60 is used only to represent leap seconds. Must be included if <Fraction> is included. If absent, the value defaults to 0.

Time			
Data Type	**Type**	**Usage**	**Description**
<fraction>	Short	Optional	Number of represented microseconds. Value must be within the range 0 through 999,999. Particular implementations may choose to allow representations of smaller fractions. If absent, the value defaults to 0.
<utcOffset>	Short	Optional	Offset from UTC in minutes. Value must be within the range -720 through +720. Value is typically a multiple of 60 (an exact number of hours), but the offset may also include half and quarter hours. Generally should be included. If absent, the value defaults to 0, that is, UTC.

Timestamp

An indication of a particular date and time expressed with a precision of one microsecond. Elements of data type Timestamp (as shown in the following table), 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.

Timestamp is 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.

Timestamp			
Data Type	**Type**	**Usage**	**Description**
<year>	Short	Required	4-digit year value.
<month>	Short	Required	Number of the represented month. Value must be within the range 1 through 12.
<day>	Short	Required	Number of the represented day. Value must be within the range 1 through 31.
<hour>	Short	Required	Number of the represented hour. Value must be within the range 0 through 23.
<minute>	Short	Required	Number of the represented minute. Value must be within the range 0 through 59.
<second>	Short	Required	Number of the represented second. Value must be within the range 0 through 60. The value 60 is used only to represent leap seconds.

Timestamp			
Data Type	**Type**	**Usage**	**Description**
<fraction>	Short	Required	Number of represented microseconds. Value must be within the range 0 through 999,999. Particular implementations may choose to allow representations of smaller fractions.
<utcOffset>	Short	Optional	Offset from UTC in minutes. Value must be within the range -720 through +720. Value is typically a multiple of 60 (an exact number of hours), but the offset may also include half and quarter hours. Generally should be included. If absent, the value is assumed to be 0, i.e. time is assumed to be UTC.

TimeDuration

A duration of time expressed in years, months, days, hours, minutes, and seconds as described in the following table:

TimeDuration			
Data Type	**Type**	**Usage**	**Description**
<years>	Short	Required	4-digit year value.
<months>	Short	Required	Number of the represented month. Value must be within the range 1 through 12.
<days>	Short	Required	Number of the represented day. Value must be within the range 1 through 31.
<hours>	Short	Required	Number of the represented hour. Value must be within the range 0 through 23.
<minutes>	Short	Required	Number of the represented minute. Value must be within the range 0 through 59.
<seconds>	Short	Required	Number of the represented second. Value must be within the range 0 through 60. The value 60 is used only to represent leap seconds.

Number

A numeric count not requiring any units.

Byte

A signed integer between -128 and +127, of type Number.

Integer

A signed integer between -2147483648 and +2147483647, of type Number.

Short

A signed integer between -32768 and +32767, of type Number.

Decimal

Decimal indicates a numeric value that is up to fifteen digits in length, excluding any punctuation such as 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 left of the leftmost digit to the right of the rightmost digit (e.g., +.12345678901234 is acceptable while 12345678901234567 is not).

NOTE: The sign is always optional. If it is absent, the value is assumed to be positive.

Percentage

A percentage.

Amount

A numeric count including units, such as liters, inches, or kilometres per liter. For example, 150 km/l. An amount is a compound data type consisting of two logical elements as described in the following table:

Tag	Type	Usage	Description
<amount>	Decimal	Required	Amount.
<unit>	String	Required	Unit.

Currency Amount

A monetary amount including the currency. A Currency Amount is a compound data type consisting of two logical elements as described in the following table:

Tag	Type	Usage	Description
<amount>	Decimal	Required	Amount.
<currencyCode>	String	Required	Currency code.

All monetary amounts in IAA-XML are handled with the Currency Amount data type. When included, this data type contains a decimal value for the amount, and an optional three-letter currency code defined in ISO-4217.

Enumeration

A value out of a limited set, each with a specific mutually exclusive meaning.

Enumeration is a Value type that has a limited number of specified valid values, each of which is represented by a tag of up to 80 characters each.

At this point, the IAA-XML specification does not provide a syntax to define permitted values for an Enumeration type. Where defined, the permitted values will be found in the description of a property.

Short

A signed integer between -32768 and +32767.

Identifier

A value without business meaning that uniquely distinguishes an occurrence.

Object reference

An identifier, unique across both space and time, with respect to the space of all Object references.

XML Schema, IAA-XML, and IFX Data Types

The W3C committee is defining a schema to be used for XML message definition and validation. One of the advantages of using the XML schema over DTDs is the ability to use and validate data types. The schema implementation is not yet available, however, forcing the first implementation of IAA-XML to use DTDs.

Oracle has examined the XML schema draft specification of data types, found at http://www.w3c.org/TR/2000/WD-xmlschema-2-20000225, and compared them with the IFX data types, as shown in Table 28.

Table 28.	XML Schema Draft Specification Comparison

IAA-XML Data Types	IFX Data Type	XML Schema Primitive Data Type
String	Character	String
Text		
Not supported	Narrow Character	String with a length attribute specified
Binary	Binary	Binary
Boolean	Boolean	Boolean
Not supported	YrMon	TimeInstant with the with the day, hour, minute, and second omitted

Table 28.   XML Schema Draft Specification Comparison

IAA-XML Data Types	IFX Data Type	XML Schema Primitive Data Type
Date	Date	TimeInstant with the hour, minute, and second omitted
Time	Time	TimeInstant with the year, month, and day omitted
Not supported	DateTime	TimeInstant with seconds expressed as integers
Timestamp	Timestamp	TimeInstant with decimal values included in seconds
Amount		
Currency	AmountCurrency	AmountNot supported
Not supported	Closed Enum	Supported by enumeration attribute of each data type
Enumeration	Open Enum	Supported by enumeration attribute of each data type
Short	Long	Supported by length attribute on decimal, with scale attribute of 0
Identifier	Identifier	ID
Not supported	Phone Number	Not supported
Decimal	Decimal	Decimal
Not supported	Universally Unique Identifier (UUID)	Not supported
Not supported	URL	URI-reference

XML schema has additional primitive data types for float (floating point number), double (double precision floating point number), TimeInstant, TimeDuration, RecurringInstant, IDREF (like IDREF in DTDs), ENTITY (like ENTITY in DTDs), and NOTATION (like NOTATION in DTDs). In addition, XML schema has derived data types that are specializations of primitive data types, such as language, integer, date, time, and name. In addition it supports user-defined data types.

Furthermore, the date and time formats for IFX, although functionally equivalent, are implemented differently. The IFX timestamp specification is the same as the XML schema format. If IAA-XML users choose to implement IFX conventions for other date and time formats in adapters, those adapters will have to change when XML schema has been adopted. Except for timestamp, IAA-XML DTDs will not use the IFX date and time data types at this time; they are defined only for consistency.

Date and Time Formats

Time is specified at the hub in a character format at the moment, with no validation expected. Siebel highly recommends users follow the ISO8601 standard, which is the proposed standard for dates and times in XML schemas. The ISO8601standard specifies time as follows:

CCYY-MM-DDThh:mm:ss.sssZ

Where CCYY is the year, MM is the month, DD is the day, T is the literal T used as a date-time separator, hh is the hours using a 24 hour clock, mm is minutes, and ss.sss is seconds. The Z indicates that the time is in Coordinated Universal Time (UTC). To indicate a timezone different from UTC follow the time with ±hh:mm to signify the difference from UTC and omit the 'Z'. The followings are examples of valid and equivalent times:

2000-04-06T19:30:40Z signifying GMT

2000-04-06T20:30:40+1:00 signifying CET

2000-04-06T13:30:40-6:00 signifying CT (US)

ISO8601 also specifies the format for time duration. It allows the omissions of parts of the date and time. This can be useful when you are only interested in the underwriting year or the renewal month and year. The following representation is used to achieve this:

PnYnMnDTnHnMnS

An optional preceding '-' (before the P) may be specified to signal a negative duration. For example, P1Y2M3DT10H30M signifies 1year, 2 months, 3 days, 10 hours and 30 minutes; -P10Y means 10 years ago. The specification may be truncated on the right. Time periods can be specified using several combinations of times: the start instant and a duration, the start instant and end instant, or the end instant and a negative duration.

NOTE: For details, refer to the ISO standard document.

B   Troubleshooting

This appendix describes troubleshooting for the following types of problems:

- Runtime event setting
- Integration object setup

Runtime Event Setting Issues

You might encounter problems due to incorrect setting of your Runtime Event.

Double Triggering the Same Workflow

Problem

After configuring the runtime event to trigger a workflow process, the system did not return or returned with the error message, "Try to read or write to the invalid memory address" at runtime.

Reason

Inside the workflow process, it may trigger the same runtime event, which is the caller of the current workflow. Therefore, the same workflow that was triggered is triggered again for the second time, and results in an infinite loop.

Solution

Use applet as event object type instead of using business component. This will restrict the event to focus on only one view instead of multiple views as business components are reused in many different ways. Users may not encounter the problem mentioned when configuring event using the business component object type only if it is certain that the selected business component will not be changed during the processing of that particular workflow.

Runtime Event Setup

Problem

Workflow is not triggered although event has been configured.

Solution

Verify the setting for the process name in Runtime Event Administration view is correct. For example, workflow process name is "IAA outbound workflow".

> Field NameField Value
>
> Business Service NameWorkflow Process Manager
>
> Business Service MethodRunProcess
>
> Business Service Context"ProcessName", "IAA outbound workflow"
>
> **NOTE:** Space is required between "ProcessName" and "IAA outbound workflow."

Workflow Process Setup for Runtime Event Support.

Problem

Workflow is not executed although event has been configured.

Solution

Verify the setting for the branch type within the workflow start step. This should be "Default" unless you have defined workflow runtime event from Workflow Designer, which in this case will be "Condition".

Problem

ObjectId is not passed to the workflow when using Runtime Event Management to trigger the workflow.

Solution

The business object name of the workflow is not setup correctly. Please select the one desired.

Integration Object Setup Issues

Another setting that could cause error in your integration process is your integration object settings.

Inactive All Unused Integration Component Fields

Problem

FINS IAA-XML Transaction Manager returns error message indicating error occurs in some component fields.

Solution

Inactive all unused integration component fields for internal integration objects created by the FINS IAA Wizard to solve the problem. In some cases, two or more integration component fields are based on the same table column, and causes SQL error in object manager.

Verify User Keys of Integration Components

Problem

FINS IAA-XML Transaction Manager cannot create or update a record.

Solution

Make sure the current user keys of the integration components are properly configured. User Keys created by FINS IAA Wizard may not fully meet customized situation. Hence, please customize the user key combination to allow the transaction manager be able to uniquely identify a business component record.

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

Setup the Integration Object User Property and Literal Values

Problem

FINS IAA-XML Converter generates XML string without some required fields.

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 need.

Out-of-Sync Database

Problem

After compiling your .srf file and moving it to your test environment you can not locate your integration objects or DTE maps.

Solution

Move your integration objects and DTE maps from your development database to your test database.

Index

E

EAI MQSeries Transport
  configuring   55
EAI Siebel Wizard
  integration objects, creating   46, 48

I

IBM MQSeries   55
integration components
  selecting   48
Integration Object Builder wizard
  integration components, selecting   48
  integration objects, creating   46, 48
integration objects

creating   46, 48

P

process properties   57, 61

S

Siebel EAI Transports. *See* **transports**
Siebel Tools
  integration objects, creating   46, 48

T

transports
  about   30