SIEBEL eBusiness 7

XML REFERENCE

SIEBEL *e*BUSINESS APPLICATION INTEGRATION VOLUME V
MIDMARKET EDITION

VERSION 7.5

12-BCK97V

SEPTEMBER 2002

Contents

Introduction

This guide explains the details behind the XML support in Siebel eAI, MidMarket Edition.

NOTE: All Siebel MidMarket product names include the phrase *MidMarket Edition* to distinguish these products from other Siebel eBusiness Applications. However, in the interest of brevity, after the first mention of a MidMarket product in this document, the product name will be given in abbreviated form. For example, after Siebel Call Center, MidMarket Edition, has been mentioned once, it will be referred to simply as Siebel Call Center. Such reference to a product using an abbreviated form should be understood as a specific reference to the associated Siebel MidMarket Edition product, and not any other Siebel Systems offering. When contacting Siebel Systems for technical support, sales, or other issues, note the full name of the product to make sure it will be properly identified and handled.

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

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

System Integrators	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution at an enterprise for specific applications and or to develop custom solutions.

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

How This Guide Is Organized

This guide provides reference information on eXtensible Markup Language (XML) and Siebel XML, as well as information on XML integrations, XML converters, Document Type Definitions (DTDs), and XML envelopes. This book is organized in a way that presents distinct blocks of information on Siebel XML as separate chapters. Additional information, as applicable, can be found in the appendices.

This book is Volume 5 of a six-volume set. The full set includes:

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

Additional Resources

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

- *Siebel Tools Online Help, MidMarket Edition* if you plan on using COM, CORBA, or the ActiveX Plug-ins to accomplish integration and also as a reference for Siebel business objects and components.
- *Siebel Business Process Designer Administration Guide, MidMarket Edition*
- *Siebel Enterprise Integration Manager Administration Guide, MidMarket Edition* if you will perform bulk loading or unloading of data.

Revision History

XML Reference: Siebel eBusiness Application Integration Volume V, MidMarket Edition, Version 7.5

Overview 1

XML is the industry standard for precisely representing data from virtually any source, stored in virtually any format. In appearance, it is similar to HTML, but HTML describes a document in terms of how it should display data in a web browser, XML *is* the data (or more precisely, the data from an application represented as XML).

This data can be from an application screen (sometimes called a “screen scraping”), it can be the output from a database or it can be an application executed using processing instructions that run Siebel eScript, for example.

There are also technologies that describe XML documents. These are known as *metadata* because the data within these documents is used to describe and format the information in an XML document. Examples of metadata documents include XSDs (XML Schema Definitions), DTDs (Document Type Definitions), and XDRs (XML Data Reduced), which are supported by Siebel applications.

Siebel and XML

Siebel eAI support for XML allows you to communicate with any Siebel system or external system, or with trading partners that can read and write XML (either arbitrary XML or Siebel XML, also known as the Siebel Message format).

XML documents are delivered directly to and from Siebel applications, or through middleware such as BizTalk using any of the supported transports: HTTP, IBM MQSeries, Microsoft Messaging Queue (MSMQ), File, and so on. XML communicated in this way can query Siebel database, “upsert” (update or insert) data, synchronize the two systems, delete data, or execute a workflow process.

Objects from various systems—Siebel Business Objects, SAP IDOCs—can be represented as common structures that the eAI infrastructure can understand: Integration Objects.

Siebel can also communciate bidirectionally with Web Services using Simple Object Application Protocol (SOAP) XML. See *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition* for more information.

NOTE: If you do a minimal client install, make sure you check the XML parser option; otherwise, you will encounter the following error when attempting to run any client process that uses the XML parser; “Unable to create the Business Service ‘EAI XML Converter.’” The XML parser is included by default in the full install.

XML Integration Objects

The Integration Object type of XML is available within Siebel systems to represent externally defined XML documents, where the object’s XML representation is compliant with the XSD or DTD supplied by your trading partner or external system. This type of integration object supports a complete representation of XML documents.

NOTE: Siebel XSD does not support the use of <import> and <include> elements and the <any> attribute. To implement the <import> or <include> functionality, place the schema definition into a single file.

Bidirectional Data Flow

The following illustration, Figure 1, shows the bidirectional progress of XML documents into and out of Siebel systems.

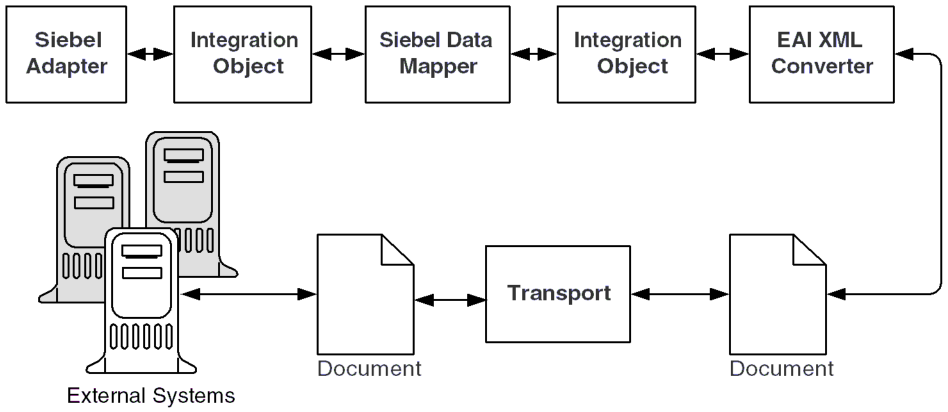


Figure 1. Document to Integration-Object Flow

NOTE: See *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition* for more information on integration objects and Web Services. For an introduction to, and overview of, the integration process, see *Overview: Siebel eBusiness Application Integration Volume I, MidMarket Edition*.

Metadata Support

For sending and receiving information for Siebel Objects in an XML format between Siebel systems and external systems, Siebel systems supports the metadata representations for XML known as XSDs (XML Schema Definitions), DTDs (Document Type Definitions), and XDRs (XML Data Reduced, a Microsoft specification). Support for XSDs and DTDs gives you a way to communicate with external systems using externally defined XML documents, instead of having to use the Siebel XSD and DTD format.

The Siebel application includes a Schema Generator wizard to assist in the creation of XML Integration Objects, using an externally defined XSD or DTD. The XSD and DTD are used to map data between the Siebel application and an external integration object, and to transform data, as needed. These tasks are conducted using the Siebel Data Mapper.

NOTE: XDR support is especially significant if your business uses Microsoft BizTalk Server for information exchange. For more information on XDR support and the Schema Generator Wizard in reference to BizTalk, see *Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition*.

Special Characters in XML Documents

Special characters should be represented in accordance with XML standards for those characters in order for them to be correctly interpreted within Siebel applications. Also, specify the character set you are using if it is not UTF-8 (the default).

Special (Escape) Characters

The EAI XML Converter can handle special characters for inbound and outbound XML, as shown in Table 1. Non-Siebel XML should already handle special characters before integrating into the Siebel application. Special characters are indicated by enclosing the text for the character between an ampersand (&) and a semicolon (;).

NOTE: You can use Microsoft Internet Explorer to check an XML document to see if the special characters have been written correctly.

Table 1. XML Escape Characters (Character Entities)

Character	Entity
<	&lt;
>	&gt;
&	&amp;
“	&quot;
‘	&apos;
Decimal Character	&#09;
Unicode Character	&#x00B0;
Date	Must follow the ISO 8601 format

Declaring the Character Set in Use

You must include the following parameter in the XML version declaration of your XML, XSD, or DTD document to declare the character set in use, if it is not the default of UTF-8:

```
<?xml version="1.0" encoding="US-ASCII"?>
```

Supported character sets include but are not limited to UCS-2, UTF-4, UTF-8, UTF-16, US-ASCII, Unicode (ISO 10646 and ISO-8859-1), and the local codepage of the Siebel server. The character set declaration “encoding” must appear after the version declaration. For example:

```
<?xml version="1.0" encoding="US-ASCII"?>
```

XML Representation of Property Sets 2

This chapter discusses the XML representation of property sets.

Mapping Between Property Sets and XML

An arbitrary property set hierarchy can be serialized to XML and an XML document can be converted to a property set hierarchy using the XML Converter business service. This service is used by the Business Service Simulator screen to save property set inputs and outputs to a file from eScript. See "Siebel XML Converter Comparison" on page 65 for Siebel Workflow information.

Each part of a property set object has a corresponding XML construct. Table 2 shows the mappings between parts of a property set hierarchy and their XML representation.

Table 2. Property Set to XML Mappings

Property Set Component	XML Representation
PropertySet	Element
PropertySet Type	Element name (If Type is not specified, the element name is set to PropertySet)
PropertySet Value	Element Character Data
Property name	Attribute name
Property value	Attribute value
Child Property Set	Child element

Element and Attribute Naming

The property set Type (which maps to an XML element name) and the names of individual properties (which map to XML attribute names) do not necessarily follow the XML naming rules. For example, a name can contain characters such as a space, a quote, a colon, a left parenthesis, or a right parenthesis that are not allowed in XML element or XML attribute names. As a result, you must perform some conversion to generate a valid XML document.

When creating an XML document from a property set hierarchy, the XML Converter will make sure that legal XML names are generated. There are two different approaches provided to handle name translation. The approach is determined by the *EscapeNames* user property on the XML Converter service. This user property can be either `True` or `False`.

- If EscapeNames is `True`, instances of illegal characters are converted to an escape sequence that uses only legal characters. For example, a space is converted to the characters "_spc". When an XML document is parsed to a property set hierarchy, the escape sequences are converted back to the original characters. For example, the name Account (SSE) becomes Account_spc_lprSSE_rpr.

  Table 3 shows the escape sequences that are used by the XML Converter.

Table 3. XML Converter Escape Sequences

Character in Property Set	Description	Generated Escape Sequence
	Space	_spc
_	Underscore	_und
“	Double Quote	_dqt
‘	Single Quote	_sqt
:	Colon	_cln
;	Semicolon	_scn
(	Left Parenthesis	_lpr
)	Right Parenthesis	_rpr

Table 3. XML Converter Escape Sequences

Character in Property Set	Description	Generated Escape Sequence
&	Ampersand	_amp
,	Comma	_cma
#	Pound symbol	_pnd
/	(Forward) slash	_slh
?	Question Mark	_qst
<	Less Than	_lst
>	Greater Than	_grt
	Other illegal characters not listed above	_<Unicode character code>

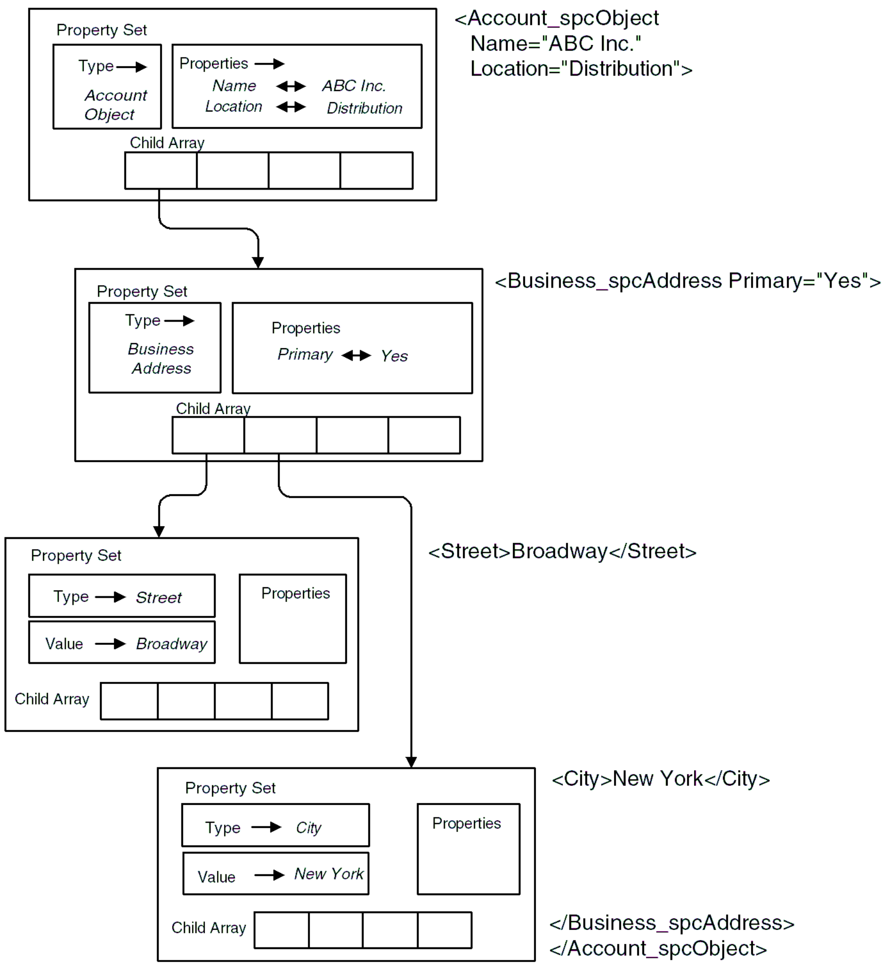
- If EscapeNames is `false`, the XML Converter removes illegal characters. These characters include the space ( ), double quote ("), single quote('), semicolon (;), left parenthesis ((), right parenthesis ()), and ampersand (&). For example, the XML Converter changes the name Account (SSE) to AccountSSE.

NOTE: These conversions are not reversible: the original names cannot be obtained from the XML names.

If a property set instance does not have a value for its Type member variable, the XML Converter uses the name “PropertySet” for the corresponding element’s name.

Example Property Set and XML

Figure 2 illustrates an example property set hierarchy and the XML that would be generated for each component of the hierarchy. The XML was generated with the EscapeNames user property set to `True`.



```
<Account_spcObject
  Name="ABC Inc."
  Location="Distribution">
<Business_spcAddress Primary="Yes">
<Street>Broadway</Street>
<City>New York</City>
</Business_spcAddress>
</Account_spcObject>
```

Figure 2. Property Set and XML with EscapeNames Set to True

Properly Formatted Property Sets

Property sets are used internally to represent Siebel eAI data. A property set is a logical memory structure that is used to pass the data between business services.

To benefit from using the XML Converter, be sure that any code you use (such as eScript or Siebel VB) correctly represents property sets within Siebel applications for the XML Converter Business Service, including necessary arguments and values. An example of such code is shown below.

```
Set Inputs = TheApplication.NewPropertySet

REM Fill in Siebel Message Header
Inputs.SetType "SiebelMessage"
Inputs.SetProperty "MessageId", ""
Inputs.SetProperty "MessageType", "Integration Object"
Inputs.SetProperty "IntObjectName", "Sample Account"

Set svc = theApplication.GetService("EAI XML Converter")
Set XMLInputs = theApplication.NewPropertySet
Set XMLOutputs = theApplication.NewPropertySet

XMLInputs.AddChild Inputs

svc.InvokeMethod "PropSetToXML", XMLInputs, XMLOutputs
```

XML Representation of Integration Object Instances 3

Any integration object instance can be represented as an XML document (or created from a properly formatted XML document). This makes it convenient to save an object to a file for viewing or to send it over a transport, such as HTTP or IBM MQSeries. You can control the format of the XML document through the integration object definition in the Siebel repository. You can use the EAI XML Converter business service to perform translations between integration object instances and the corresponding XML representation.

Integration Objects

Integration objects are logical representations of Siebel business objects or of external application data, such as SAP Business Application Programming Interfaces (BAPIs) or externally-defined XML documents. An integration object is metadata stored in the Siebel repository. One integration object can be mapped to another integration object. Instances of integration objects are used in integration processes for data exchange.

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

Integration objects are made up of three distinct data sections: the canonical, the external, and the XML, as shown in Figure 3.

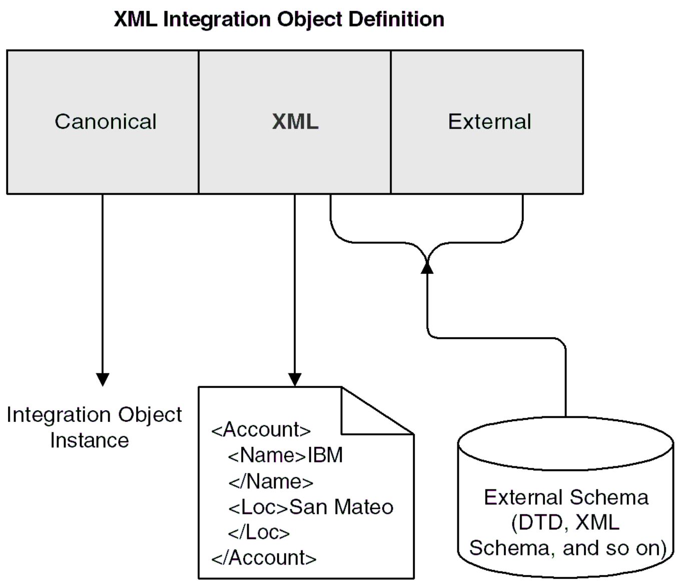

XML Integration Object Definition
Canonical
XML
External
Integration Object Instance
<Account>
<Name>IBM
</Name>
<Loc>San Mateo
</Loc>
</Account>
External Schema (DTD, XML Schema, and so on)

Figure 3. XML Integration Object Definition

The integration object schema in the Siebel repository is composed of the three data sections shown in Table 4.

Table 4. Integration Object Data Type

Name	Purpose
Canonical section	Stores information about an object in a common representation. The names used for objects, components, and fields are the names that the designer wishes to be visible in Siebel systems. The data types are the Siebel business component field types that are used by the Object Manager.
External section	Stores information about how the object, component, or field is represented in the external system. For integration objects based on business objects, this may include the business object names, component names, and field names and data types. For integration objects based on a SAP IDOC, this may include the names used within SAP as well as information needed by the IDOC adapter to parse and generate IDOCs, such as the field offsets.
XML section	Stores the mapping between an integration object definition and its XML representation. This allows any integration object to be represented as XML.

Elements and Attributes

An XML document consists of one or more *elements*. An element consists of a start tag and an end tag that enclose character data, nested elements, or both. For example, here is a simple element called *Element1*, with two tags containing character data:

```
<Element1>
This is character data.
</Element1>
```

The next example shows an element nested within another element. Parent-child relationships are frequently represented using nested elements.

```
<Element1>
  <NestedElement>
  data
  </NestedElement>
</Element1>
```

Elements can have attributes that refine or modify the element's default attributes. An attribute is a key and value pair of strings, contained within the start tag of an element. In the following example, *status* is an attribute that is assigned the value *test*. Attributes are frequently used to specify metadata about an element.

```
<Element1 status="test">
This is character data.
</Element1>
```

In the Siebel representation, objects and components are represented by XML elements. A set of integration object *instances* of a given type are nested within the *object* element for that type.

An element represents each *component*. Child components are nested within their parent's elements. Fields can be either elements nested within their containing component element or attributes of the component element. You can set the XML Style attribute of the integration component field definition to specify which style represents a given field.

How XML Names Are Derived from Integration Objects

When Siebel Tools generates the XML representation of your integration object, it derives the XML element and attribute names from the Siebel repository names of the integration object, its components, and fields. Siebel repository names can, however, contain characters not permitted in an XML name—for example, blank spaces. Thus, some translation needs to be performed to make sure a valid XML name is derived from such a repository name. In addition, XML element names must be unique in the document in which they are defined. This can cause a parsing problem if two integration components have fields with the same name.

To handle these issues, Siebel Tools stores a separate name in the XML Tag attribute of the integration object, component, and field. When you create an integration object using a wizard, the *XML Tag* attribute is initialized to the value of the *Name* column, with any illegal characters removed from the name. In addition, Siebel Tools might add a number to the tag name if the same name is already in use by a different object, component, or field. You can change the XML names after the integration object has been created, if necessary.

Elements Within a Siebel Integration Object Document

An integration object can be textually represented as an XML document. In order to exchange data using the Siebel integration object document, you need to have an understanding of its XML structure, including elements and attributes. The document can potentially contain up to five different types of elements:

- SiebelMessage Element
- Object List Element
- Integration Component Element
- Component Container Element
- Integration Field Element

SiebelMessage Element

When integration object documents are sent to an external system, they may be encapsulated within a *SiebelMessage* element. This element identifies the document as a Siebel message and indicates that the document contains integration object instances. It can also provide metadata, such as the integration object type and a message ID.

NOTE: The SiebelMessage element is optional. The presence of this element is determined at run time through arguments to the EAI XML Converter Business Service.

Since the Object List element (see "Object List Element" on page 29) is optional, SiebelMessage can contain a Root component element to allow for cases when the Object List element is left blank (omitted).

Attributes

The SiebelMessage element can contain a number of attributes, which are known as the Message Header attributes. The set of standard attributes, described in the following subsections, have well-defined meanings. In addition, you can add arbitrary attributes to the SiebelMessage element.

An XSD or DTD for the document can be dynamically generated inline to include all present attributes. The standard attributes are described in the following subsections.

IntObjectName

The name of the integration object type contained within the message. If the message is an integration object message, you must specify this property.

MessageId

A unique ID for a given message as it flows through a connector. This is an optional field that might be useful for tracking message processing through the system.

Child Elements

For integration object messages, the SiebelMessage element contains exactly one *object list element*. Since only one object list element is permitted in each XML document, only one *integration object type* can be represented in a given document.

Object List Element

The *object list element* is a container for the integration object instances. The XML Tag attribute value that you specify in the integration object definition becomes the name of this element. By default, an integration object wizard generates an XML Tag value of *ListOf<Name>*, where *<Name>* is the name of the integration object, with any illegal XML characters removed—for example, spaces.

NOTE: The Object List element is optional. The XML element is not generated if the Object List element is blank (omitted) in the integration object definition.

Attributes

None.

Child Elements

The object list element can contain one or more instances of the integration object's *root component element*. A root component element corresponds to one integration object instance.

Integration Component Elements

An *integration component element* corresponds to an *integration component type* in the repository definition.

Component parent-child relationships are represented by a structure in which the child components of a given component type are nested within a component container element. The *component container element* is, in turn, nested within the *parent component instance.*

Thus, all components within an integration object instance are indirectly nested within the root component. Only one instance of the root component is allowed for each object instance. The root component is nested within the object list element. The object list element permits multiple integration object instances of a given type within the XML document.

The field children of an integration component element can be either elements or attributes, depending on the XML Style setting for each field. The component container elements of a given component appear after the fields in the XML document.

In the following example, Contact child components are nested within the Account component instance:

```
<Account>
. . .
Account Field Elements
. . .
  <ListOfContacts>
    <Contact> . . . Contact 1 . . . </Contact>
    <Contact> . . . Contact 2 . . . </Contact>
  </ListOfContacts>
<Account>
```

Attributes

Any field that has an XML Style set to Attribute is an attribute of its component element. The name of the attribute is the same as the XML Tag of the field.

Child Elements

An integration component element can contain integration field elements and component container elements. The field elements must appear before the component container elements. The name of a field element is determined by the value of its XML Tag attribute, as defined in Siebel Tools.

Component Container Elements

An *integration component container* encloses a list of *child component instances* of the same type. The integration component container organizes child component instances by type and permits the specification of empty containers—functionality needed by the EAI Siebel Adapter. All component types, except the root component, are enclosed within container elements.

By default, the name of a *component container element* is *ListOf* plus the element name of the component type it encloses. For example, the component container for Contact is *ListOfContact*. You can override the default name by specifying a name in the XML Container element field of the component's definition.

Another option is to leave the container element blank. In that case, the component element is the child of the parent component element.

Attributes

None.

Child Elements

Zero or more instances of the component element associated with the container.

Integration Field Elements

An *integration field element* contains the value of the specified field. It must appear in an instance of its parent integration object type. If a field element has no contents (signified by a start tag immediately followed by an end tag), it is interpreted to mean that the field's value should be set to empty. The same is true when a field’s value is empty; the field element will have a start tag immediately followed by an end tag.

The order in which XML fields appear within their parent component element is determined by the Sequence field in the Tools definition of the field.

All fields are optional. If a field element is not present in a component element, the field is not created in the integration object instance.

Child Elements

Integration component fields have a property called XML Parent Element. If this property contains the name of another field, then that field (either as an attribute or as an element) appears as a child of its parent field’s element.

Example XML Document

The following XML document represents an instance of the Sample Account integration object. This document contains one account instance: A. K. Parker Distribution. The instance has one business address and two contacts.

Note that the PhoneNumber field of the business address appears as an *attribute*. This means that the XML Style in the field's definition in Siebel Tools is set to the Attribute style. The rest of the fields are represented by XML elements.

```
<SiebelMessage MessageId=""
IntObjectName="Sample Account">
<ListofSampleAccount>
  <Account>
    <Name>A. K. Parker Distribution</Name>
    <Location>HQ-Distribution</Location>
    <Organization>Siebel Organization</Organization>
    <Division></Division>
    <CurrencyCode>USD</CurrencyCode>
    <Description></Description>
    <HomePage></HomePage>
    <ListOfBusinessAddress>
      <BusinessAddress PhoneNumber="6502955000">
      <City>Menlo Park</City>
      <Country>United States of America</Country>
      <FaxNumber></FaxNumber>
      <StreetAddress>1000 Industrial Way</StreetAddress>
      <Province></Province>
      <State>CA</State>
      <PostalCode>94025</PostalCode>
    </BusinessAddress>
    </ListOfBusinessAddress>
    <ListOfContact>
      <Contact>
        <FirstName>Stan</FirstName>
        <JobTitle>Senior Mgr of MIS</JobTitle>
        <LastName>Graner</LastName>
        <MiddleName></MiddleName>
        <Organization>Siebel Organization</Organization>
        <PersonalContact>N</PersonalContact>
      </Contact>
    <Contact>
      <FirstName>Susan</FirstName>
      <JobTitle>President and CEO</JobTitle>
      <LastName>Grant</LastName>
      <MiddleName></MiddleName>
      <Organization>Siebel Organization</Organization>
      <PersonalContact>N</PersonalContact>
    </Contact>
    <Contact>
</ListOfContact>
</Account>
</ListofSampleAccount>
</SiebelMessage>
```

XML Schema Definitions (XSDs)

The XML Schema Definition (XSD) language describes the content of an XML document. The definition can describe which elements are allowed and how many times the element can be seen. The schema can be used to generate an integration object through Siebel Tools. The feature is accessed through the Integration Object Builder.

Here is an example of an XSD for the Sample Account integration object as generated by Siebel Tools:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://siebel.com/xsd/SampleAccount.xsd"
xmlns:xsdLocal="http://siebel.com/xsd/SampleAccount.xsd"

>

<xsd:element name = "elem1" type ="xsd:string" minOccurs ="0"
maxOccurs = "1"/>

<xsd:element name = "elem2" type ="xsd:string" minOccurs ="0"
maxOccurs="unbounded"/>

</xsd:schema>
```

Document Type Definitions (DTDs)

The Document Type Definition (DTD) provides metadata describing the structure of an XML document. It can be used by validating XML parsers to make sure that a given document instance conforms to the expected structure—that is, the structure defined in the DTD.

You can generate the DTD for an integration object by using the Generate Schema feature in Siebel Tools. The feature is activated by clicking the Generate Schema button in Siebel Tools after selecting a given integration object definition.

The SiebelMessage element is optional. It can be omitted by selecting the “No Envelope” option in the Generate XML Schema wizard.

The DTD for the message header is generated in the actual XML document at run-time. The generation of this inline DTD and a reference to the external portion is enabled through the GenerateDTD parameter of the EAI XML Converter.

Here is an example of a DTD for the Sample Account integration object as generated by Siebel Tools:

```
<!-- Siebel DTD Generation -->
<!-- Shared Element List. These elements are guaranteed -->
<!-- to have the same datatype, length, precision, and scale.-->
<!ELEMENT Name (#PCDATA) >
<!ELEMENT Location (#PCDATA) >
<!ELEMENT Division (#PCDATA) >
<!ELEMENT Description (#PCDATA) >
<!ELEMENT CurrencyCode (#PCDATA) >
<!ELEMENT StreetAddress (#PCDATA) >
<!ELEMENT State (#PCDATA) >
<!ELEMENT PostalCode (#PCDATA) >
<!ELEMENT Country (#PCDATA) >
<!ELEMENT City (#PCDATA) >
<!ELEMENT Organization (#PCDATA) >
<!ELEMENT ListofSampleAccount (Account+) >
<!ELEMENT Account (Name?,
      Location?,
      Organization?,
      Division?,
      CurrencyCode?,
      Description?,
      HomePage?,
      LineofBusiness?, BusinessAddress?, Contact?)>
```

```
<!ELEMENT HomePage (#PCDATA) >
<!ELEMENT LineofBusiness (#PCDATA) >
<!ELEMENT BusinessAddress (BusinessAddress*) >
<!ELEMENT BusinessAddress (City?,
      Country?,
      FaxNumber?,
      StreetAddress?,
      Province?,
      State?,
      PostalCode?)>
<!ATTLIST BusinessAddress PhoneNumber CDATA #IMPLIED >
<!ELEMENT FaxNumber (#PCDATA) >
<!ELEMENT Province (#PCDATA) >
<!ELEMENT Contact (Contact*) >
<!ELEMENT Contact (CellularPhone?,
      FirstName?,
      HomePhone?,
      JobTitle?,
      LastName?,
      MiddleName?,
      Organization?,
      PersonalContact?,
      Account?,
      AccountLocation?)>
<!ELEMENT CellularPhone (#PCDATA) >
<!ELEMENT FirstName (#PCDATA) >
<!ELEMENT HomePhone (#PCDATA) >
<!ELEMENT JobTitle (#PCDATA) >
<!ELEMENT LastName (#PCDATA) >
<!ELEMENT MiddleName (#PCDATA) >
<!ELEMENT PersonalContact (#PCDATA) >
<!ELEMENT Account (#PCDATA) >
<!ELEMENT AccountLocation (#PCDATA) >
```

A few comments about this DTD:

- If a field’s XML tag appears in multiple components, its definition appears only once in the DTD, at the beginning. When creating XML tag names for fields, the wizard only reuses a field name if all instances have the same datatype, length, precision, and scale.
- All fields are optional, but if present, they must appear in the correct order.

XML Integration Objects and the XSD Wizard 4

This chapter discusses XML integration objects and the XSD wizard.

Creating XML Integration Objects with the XSD Wizard

Siebel eAI provides two different wizards to create XML integration objects. An XML integration object is essentially an integration object with a base object type of XML. This wizard parses the XML Schema Definition (XSD) file to create an XML integration object.

To create an integration object

1 Select File > New Object.

The New Object window appears.

2 Select the EAI tab.

3 Double-click the Integration Object icon.

The Integration Object Builder wizard appears.

4 Complete the Integration Object Builder initial page:

   a Select the project from the first drop-down list.

   b Select EAI XSD Wizard as the Business Service.

   c Naviagate to the path to the location of the XSD or XML file that you want to use as the basis of the XSD.

   NOTE: The Simplify Integration Object Hierarchy option creates a simpler and flatter internal representation of the XML integration object. Note that this does not change the external representation. Having a simpler internal representation makes declarative data mapping easier.

   d Click Next.

   The dialog box changes to the next page.

5 Select the source object, name it, and click Next.

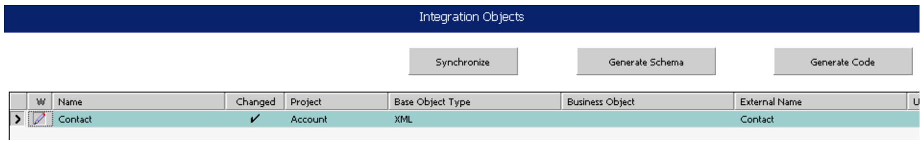
6 Click on the plus sign to expand the list to select or clear the fields you need from the component.

7 Click Next.

The final page appears.

8 Click Finish.

You will see the integration object you created in the Integration Objects list, as shown in the following figure.



W	Name	Changed	Project	Base Object Type	Business Object	External Name	U
	Contact	✓	Account	XML		Contact	

NOTE: You must review the integration objects and the integration components created by the Wizard and complete their definitions based on your requirements.

Root Elements

Each XML document has exactly one root or document element. The root element corresponds to the integration object. However, because a XSD or DTD file can be used by a vendor to specify the XML documents that it can generate, the root element can not be inferred from the XSD or DTD file. For example, Ariba can generate XML for contracts, order requests, subscriptions, and so on. A single file, describes the possible XML documents.

As a reference when determining the root element, use an XML document that best represents the XML documents you are integrating. The root element is the root of the XML hierarchy tree. No part of the root element appears within the content of any other element. For all other elements, the <Start> </Start> tag appears within the content of another element.

To view any XML hierarchy, with collapsible and expandable elements, use an XML editor, an XML reader, or an XML-capable browser such as Microsoft Internet Explorer.

Integration Object Properties

Table 5 lists XSD mapping information for integration objects.

Table 5. Integration Object Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
Base Object Type	Canonical	Required	= “XML”
Inactive	Canonical		= “FALSE”
Name	Canonical	Required	= Input value from wizard
Project Name	Canonical	Required	= Input value from wizard

Integration Object User Properties

Table 6 lists XSD mapping information for integration objects.

Table 6. Integration Object User Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
XSDTypeName			= “type”
XSDTypeNamespace			= “targetNamespace”

Integration Component Properties

Table 7 lists XSD mapping information for integration components.

Table 7. Integration Component Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
Cardinality	Canonical		Based on minOccurs or maxOccurs.
External Name	External	Required	= Element name
External Name Context	External	Required	= Name
External Name Sequence	External	Required	= 1 if the element is unique. Increment for components that have the same external name.
External Sequence	External		XML Sequence
Inactive	Canonical		= “FALSE”
Name	Canonical	Required	= XML tag
Parent Integration Component	Canonical	Optional	= Blank for root = Parent element name otherwise
XML Sequence	XML		= Sequence within parent element
XML Tag	XML	Optional	= “Element Name”

Integration Component User Properties

Table 8 lists XSD mapping information for integration components.

Table 8. Integration Object User Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
XSDTypeName			= “type”
XSDTypeNamespace			= “targetNamespace”

Integration Component Field Properties

Table 9 lists XSD mapping information for integration component fields.

Table 9. Integration Component Fields Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
External Length	External	Optional	= Length
External Name	External	Required	= Attribute or element name
External Sequence	External		= XML Sequence
Field Type	Canonical	Optional	= "Data"
Inactive	Canonical		= "FALSE"
Length	Canonical	Optional	= maxLength or length
Name	Canonical	Required	= XML Tag
Data Type	Canonical	Required	= "DTYPE_TEXT"
XML Literal Value	XML	Optional	= fixed or default
XML Parent Field	XML	Optional	= Name of the parent field that maps to the parent XML element
XML Sequence	XML	Optional	= Sequence in the Parent element.
XML Style	XML	Optional	= "Attribute" if it is from an attribute = "componentElementValue" if this field holds value for Parent component element. = "Element"
XML Tag	XML	Optional	= "Element or Attribute name"

Integration Component Field User Properties

Table 10 lists XSD mapping information for integration components.

Table 10. Integration Object User Properties Mapping from XSD

Property Name	Property Data Type	Required	Property Value
XSDTypeName			= “type”
XSDTypeNamespace			= “targetNamespace”

XML Integration Objects and the DTD Wizard 5

This chapter discusses XML integration objects and the DTD wizard.

Creating XML Integration Objects with the DTD Wizard

Siebel eAI provides two different wizards to create XML integration objects. An XML integration object is essentially an integration object with a base object type of XML. This wizard parses an external Document Type Definition (DTD) file to create an XML integration object.

To create an integration object

1. Select File > New Object.

   The New Object window appears.

2. Select the EAI tab.
3. Double-click the Integration Object icon.

   The Integration Object Builder wizard appears.

4 Complete the Integration Object Builder initial page:

a Select the project from the first drop-down list.

b Select EAI DTD Wizard as the Business Service.

c Navigate to the path to the location of the DTD or XML file that you want to use as the basis of the DTD.

NOTE: The Simplify Integration Object Hierarchy option creates a simpler and flatter internal representation of the XML integration object. Please note that this does not change the external representation. Having a simpler internal representation makes declarative data mapping easier.

---

d Click Next.

The dialog box changes to the next page.

5 Select the source object and name it, and then click Next.

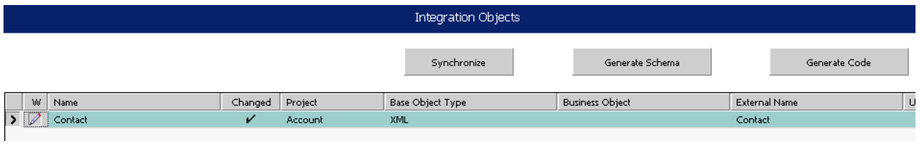
6 Click on the plus sign to expand the list to select or clear the fields you need from the component.

7 Click Next.

The final page appears.

8 Click Finish.

You will see the integration object you created in the Integration Objects list, as shown in the following figure.



W	Name	Changed	Project	Base Object Type	Business Object	External Name	U
	Contact	✔	Account	XML		Contact	

NOTE: You must review the integration objects and the integration components created by the Wizard and complete their definitions based on your requirements.

Root Elements

Each XML document has exactly one root or document element. The root element corresponds to the integration object. However, because an XSD or DTD file can be used by a vendor to specify the XML documents that it can generate, the root element cannot be inferred from the XSD or DTD file. For example, Ariba can generate XML for contracts, order requests, subscriptions, and so on. A single file describes the possible XML documents.

As a reference when determining the root element, use an XML document that best represents the XML documents you are integrating. The root element is the root of the XML hierarchy tree. No part of the root element appears within the content of any other element. For all other elements, the <Start></Start> tag appears within the content of another element.

To view any XML hierarchy, with collapsible and expandable elements, use an XML editor, an XML reader, or an XML-capable browser such as Microsoft Internet Explorer.

Integration Object Properties

Table 11 lists DTD mapping information for integration objects.

Table 11. Integration Object Properties Mapping from DTD

Property Name	Property Data Type	Required	Property Value
Base Object Type	Canonical	Required	= "XML"
Inactive	Canonical		= "FALSE"
Name	Canonical	Required	= Input value from wizard
Project Name	Canonical	Required	= Input value from wizard
XML Tag	XML	Required	= Root element name

Integration Component Properties

Table 12 lists DTD mapping information for integration components.

Table 12. Integration Component Properties Mapping from DTD

Property Name	Property Data Type	Required	Property Value
Cardinality	Canonical		Cardinality is mapped as shown in Table 14 on page 57.
External Name	External	Required	= Element name
External Name Context	External	Required	= Name
External Name Sequence	External	Required	= 1 if the element is unique. Increment for components that have the same external name.
External Sequence	External		Increment for components belonging to the same parent.
Inactive	Canonical		= "FALSE"
Name	Canonical	Required	= Element name
Parent Integration Component	Canonical	Optional	= Blank for root = Parent element name otherwise
XML Sequence	XML		= Sequence within parent element
XML Tag	XML	Optional	= "Name" with modification to fit XML tag restriction.

Integration Component Field Properties

Table 13 lists DTD mapping information for integration component fields.

Table 13. Integration Component Fields Properties Mapping from DTD

Property Name	Property Data Type	Required	Property Value
External Length	External	Optional	= Length
External Name	External	Required	= Attribute or element name
External Required	External	Optional	= "TRUE" if the attribute is required; that is, it has a default value, or the default modifier is #REQUIRED or FIXED. = "TRUE" for the element.
External Sequence	External		= Blank for attribute Increment for elements belonging to same parent.
Field Type	Canonical	Optional	= "Data"
Inactive	Canonical		= "FALSE"
Length	Canonical	Optional	=
Name	Canonical	Required	= Attribute or element name
Data Type	Canonical	Required	= "DTYPE_TEXT"
Required	Canonical	Optional	= External Required property
XML Literal Value	XML	Optional	= Fixed or Default
XML Parent Field	XML	Optional	= Name of the parent field that maps to the parent XML element
XML Sequence	XML	Optional	= Sequence in the Parent element.

Table 13. Integration Component Fields Properties Mapping from DTD

Property Name	Property Data Type	Required	Property Value
XML Style	XML	Optional	= "Attribute" if it is from an attribute = "componentElementValue" if this field holds value for the Parent component element. = "Element"
XML Tag	XML	Optional	= "Name"

How the DTD Wizard Creates XML Integration Objects

Integration objects consist of elements, attributes, PCDATA, names, hierarchy, connectors, and cardinality.

CAUTION: The DTD Wizard will take out the recursion by breaking loops. Entities in XML at run time are not supported.

Elements

Generally, XML elements map to components within integration objects. However, in many cases the component is so simple that it is a performance optimization to map these elements into component fields of the parent element rather than as child components.

Elements are expressed this way (within brackets and starting with an exclamation point):

```
<!ELEMENT car (year, model, color+)>
```

An element can be mapped to a component field when the following three properties are satisfied:

- The component field must match an element within the DTD.
- The component field must match the cardinality of the element in the DTD; in other words, if the DTD specifies only one instance of this element type is valid, all subsequent appearances of this element type are illegal.
- The element must appear *within* the root element; any element appearing outside of the root is illegal.

When an element is mapped to component field, the component field has the property XML Style set to Element.

Attributes

Attributes contain additional information related to an element, and can be either required or implied (optional) and may optimally have a default value. For example, an element might be “car” with “soundsystem,” “transmission,” and “doors” as attributes. “Soundsystem” can be any text and is required; “transmission” is required because there is a default listed; “other” is optional. This would be expressed this way:

```
<!ELEMENT car>

<!ATTLLST car

        soundsystem            CDATA        #REQUIRED

        transmission            (automatic | manual | 5-speed-manual) "automatic"

        other            CDATA        #IMPLIED>
```

Attributes are always mapped to component fields and are related directly to elements. The component field within Siebel application has the XML Style property set to Attribute.

Element’s #PCDATA

If the element is mapped to an integration component, then its #PCDATA is mapped to a component field `<!Element> #PCDATA`. If the element is mapped to a field, then the `#PCDATA` is mapped to the value of the field.

Names

Name is the name of the component or the field of the integration object. Because these names have to be unique within an integration object, the names may have suffixes attached to make them unique.

- Property *External Name* is the name of the attribute or the element in the external system, such as CustName.
- Property *XML Tag* is the name of the tag in the XML, such as <customer>.

Hierarchy

The parent components of integration components in an integration object correspond to their parents in XML. For integration component fields, if the property *XML Parent Field* is set, then the field in the same component with its *Name* value equal to the *XML Parent Field* corresponds to the parent in the XML. This happens because elements can be mapped to fields of integration components.

For integration component fields, if the property *XML Parent Field* is not set, then the parent component corresponds to the parent in the XML.

Connectors

Connectors specify the order of elements and “either/or” relationships, if one exists.

Connector	Explanation	Example
,	followed by	(a,b)
|	one or the other	(a | b)

CAUTION: The Siebel DTD wizard does not support “one or the other” (|) relationships expressed in DTDs. “One or the other” (|) will be treated the same as “followed by” (,).

Cardinality

The DTD syntax allows you to specify a cardinality—that is, the number of times an element can appear within an XML document—for elements using the modifiers question mark (?), plus sign (+), and asterisk (*), or none. (See Table 14.) Elements with a cardinality, or occurrence, specified in a DTD map only to Integration Components. The Cardinality property in the Integration Component within Siebel maps to the specified *cardinality* information in the DTD.

Table 14. Rules for Mapping for Cardinality Within a DTD

DTD Element Occurrence Operator		Integration Component Cardinality Property	
	Appears once and only once		
?	Appears 0 or 1 time	“Zero or One”	Appears 0 or 1 time
+	Appears 1 or more times	“One or More”	Appears 1 or more times
*	May appear 0 or more times	“Zero or More”	May appear 0 or more times
No modifier	Appears exactly one time	“One”	Appears exactly one time

CAUTION: The specification for DTDs supports using parentheses to express complex hierarchical structures. For example:

```
<!ELEMENT rating ((tutorial | reference)*, overall)+ >
```

The DTD Wizard uses the operator (?, *, +, or “none”) closest to the child element as that child element’s cardinality. In addition, the DTD Wizard will ignore such grouping by parentheses as illustrated above.

Siebel XML Converters 6

Siebel eAI includes four XML converter business services:

- EAI XML Converter
- XML Hierarchy Converter
- EAI Integration Object to XML Hierarchy Converter
- XML Converter

Table 15 on page 65 outlines the differences among these converters. Using these converters, Siebel eAI supports three types of standard XML integrations:

XML integration using Siebel XML. This integration uses XML that conforms to the XML Schema Definition (XSD) or Document Type Definition (DTD) or schema generated from any Siebel integration object. Siebel XML is generated by the external application or by a third-party product. This type of integration uses the EAI XML Converter business service.

XML integration without using Integration Objects. For this integration, any necessary data mapping and data transformation must be handled using custom eScripts. This type of integration uses the XML Hierarchy Converter business service.

XML integration using XML Integration Objects. With this integration, XML integration objects are mapped to Siebel Integration Objects using Siebel Data Mapper and are based on external XSDs or DTDs. XML integration objects are used to map data between the external application and Siebel eBusiness Applications. This type of integration uses the EAI XML Converter business service.

NOTE: There are also two associated business services for XML that combine XML Converters with file reading and writing, which are useful for testing and debugging during the development phase. These are the EAI XML Read from File business service and the EAI XML Write to File business service.

You can specify most parameters for the XML Converters as either business service method arguments or as user properties on the business service. If a business service method argument and a user property have the same name, the business service method argument always takes precedence over the user property.

EAI XML Converter

The EAI XML Converter uses integration object definitions to determine the XML representation for data. It converts the data between an integration object hierarchy and an XML document. Figure 4 shows the translation of an XML document into an integration object property set in Siebel application and back again. The integration object property set of type Siebel Message will appear as a child of the Service Method Arguments property set.

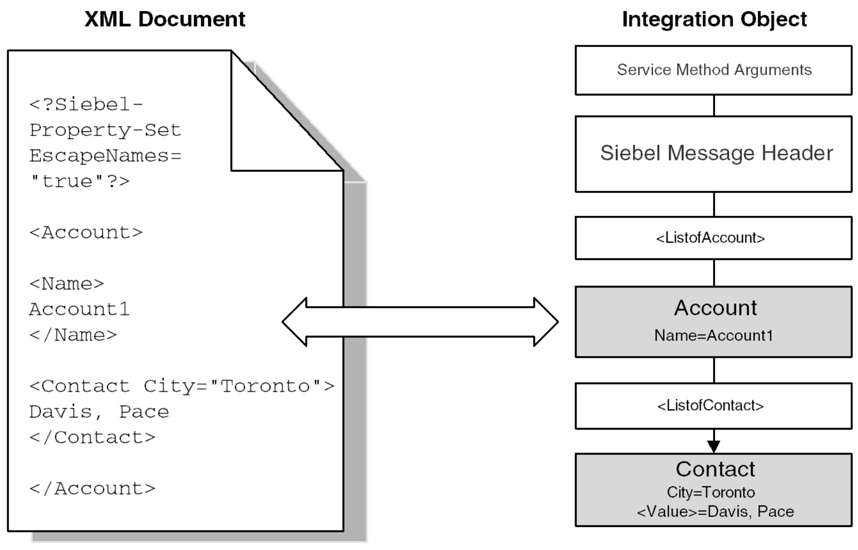


Figure 4. XML Document to Integration Object

XML Hierarchy Converter

The XML Hierarchy Converter does not use integration object metadata, but instead relies on simple rules for converting between an XML hierarchy and an XML document. The important distinction between this service and the XML Converter is a Property Set of type XMLHierarchy, which is always presented as a child of Service Method Arguments and as a parent of the XML document root element (see Figure 5).

As shown in the figure, every XML element becomes a property set where the XML tag name becomes the Type. For example, the XML element `Contact` becomes a property set of the type Contact in Siebel application. In addition, every XML attribute becomes a property within the element's property set. For example, if the attribute of the XML element "`Contact`" is `City = "Toronto`"), "City = Toronto" will be a property for "Contact."

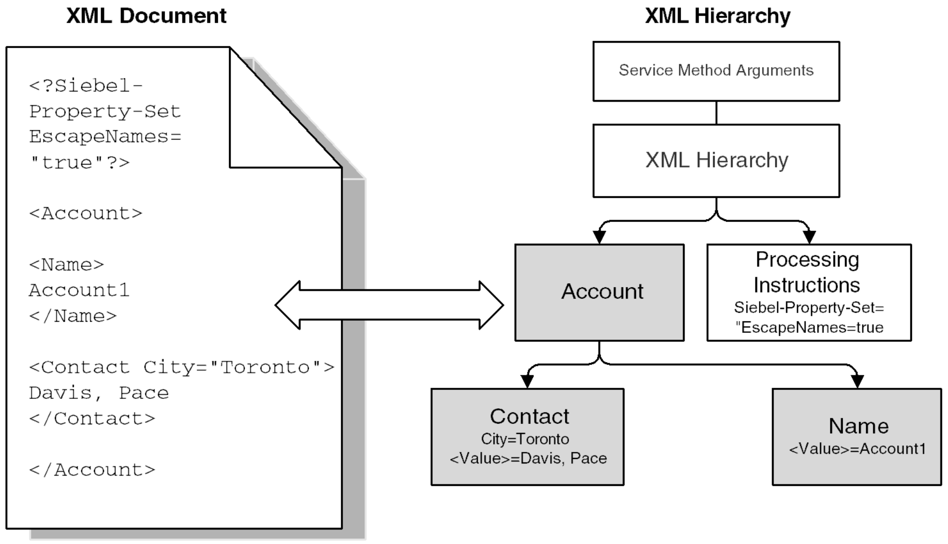

XML Document
<?Siebel-Property-Set EscapeNames="true"?>
<Account>
<Name> Account1 </Name>
<Contact City="Toronto"> Davis, Pace </Contact>
</Account>
XML Hierarchy
Service Method Arguments
XML Hierarchy
Account
Processing Instructions Siebel-Property-Set= "EscapeNames=true
Contact City=Toronto <Value>=Davis, Pace
Name <Value>=Account1

Figure 5. XML Hierarchy Representation of XML Document Structure

The convenience of having this representation is that the XML Hierarchy Converter can convert to and from this representation in the same way, independent of whether or not the XML document contains a Siebel Message or an external XML document. This representation is also handled in Siebel Workflow because it allows all of the XML documents in memory to be treated as the Hierarchical Service Parameter of type XML Hierarchy.

The following should be noted about XML Hierarchy representation in Siebel application:

- There is a Property Set of type XML Hierarchy that always appears as a child of the Service Method Argument and the parent of the root XML element (as shown at the top of the gray “XML Hierarchy” portion of Figure 5 on page 61).

- Elements are represented by Property Sets. The XML tag is the type in the property set. If an XML element has a value (such as `<Contact City=”Toronto”>Davis, Pace</Contact>` in Figure 5 on page 61), then the value there is set in the Siebel XML Value column in the element’s Property Set.

- Attributes are represented as properties on the Property Set that represent the attribute’s element.

- Child elements are represented as child property sets and Parent elements as Parent property sets.

- Processing instructions are represented as a child Property Set of type ProcessingInstructions, which is at the same level as the root element (the child of XML Hierarchy). In Figure 5 on page 61, the root element is Account.

EAI Integration Object to XML Hierarchy Converter

The EAI Integration Object to XML Hierarchy Converter can be used if additional types of XML processing are needed, such as adding new elements, attributes, or envelopes to in-memory integration object property sets. XML Hierarchy property sets can be manipulated using eScript and Siebel VB.

XML Converter

The XML converter uses no integration object metadata. The rules for converting between XML documents and property sets are essentially the same as the XML Hierarchy Converter.

This service, however, does not create an XML hierarchy property set, but instead the XML document's root element becomes a Type top-level property set (for example, Service Method Arguments). The service is intended for importing and exporting hierarchical data (arguments, definitions, and so on) and for passing property set arguments to and from business services.

Figure 6 shows the translation of an XML document into a property set representation within Siebel XML, and back again.

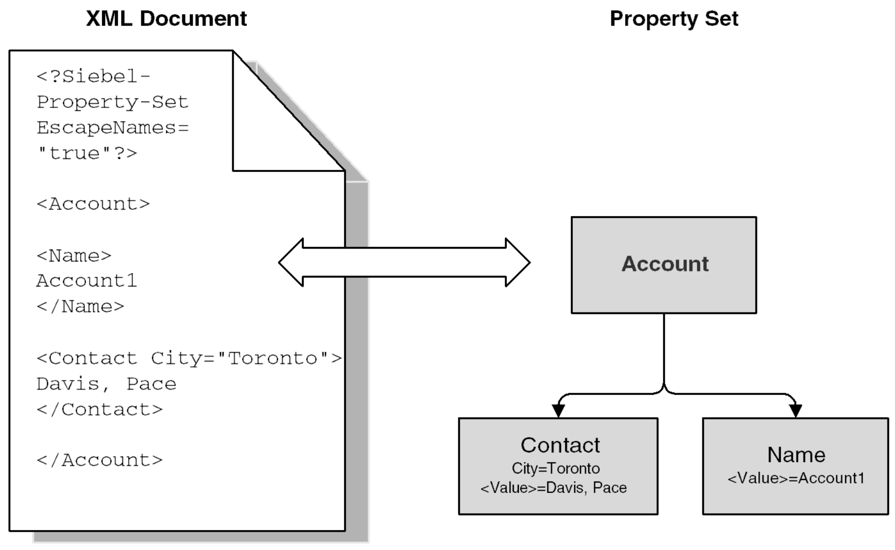


Figure 6. XML Document to Property Set Representation

Siebel XML Converter Comparison

Table 15 shows the basic differences between the four XML Converter business services. The table also gives guidelines on the appropriate usage.

Key ■ Supported by converter ❏ Supported in conjunction with second converter

Table 15. Siebel XML Converter Comparisons

User Requirement	**EAI XML Converter**	**XML Hierarchy Converter**	**EAI Integration Object to XML Hierarchy Converter**[1]	**XML Converter**
Using Siebel Workflow	■	■	❏	
Using Siebel Data Mapper	■	❏[2]	❏	
Using eScripts for data transformation in your integration	■	■	❏	
Using custom XML envelopes in your integration		■	❏	
Using Dispatch Service in your integration		■	❏	
Your XML represents Business Service arguments			❏	■
Serializing Property Sets as XML			❏	■
Internal Representation				
	Siebel Message (Integration Object Instance)	XML Hierarchy	Siebel Message (Integration Object Instance)	Property Set
Prerequisites				
Create an Integration Object definition	■		■	

1. Must be used in conjunction with the XML Hierarchy Converter.
2. Must be used in conjunction with the Integration Object Hierarchy Converter.

EAI XML Converter Methods and Arguments

There are two methods for the EAI XML Converter: Integration Object Hierarchy to XML Document and XML Document to Integration Object Hierarchy, as described in Table 16. The arguments for each method appear in the tables that follow.

Table 16. EAI XML Converter Methods

Display Name	Name	Description
Integration Object Hierarchy to XML Document	IntObjHierToXMLDoc	Converts an integration object hierarchy into an XML document.
XML Document to Integration Object Hierarchy	XMLDocToIntObjHier	Converts an XML document into an integration object hierarchy.

Integration Object Hierarchy to XML Document Method Arguments

Table 17 describes the input arguments for the Integration Object Hierarchy to XML Document method of the EAI XML Converter.

Table 17. Integration Object Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
Siebel Message	SiebelMessage	Hierarchy	The Integration Object Hierarchy to be converted to XML.
XML Character Encoding	XMLCharEncoding	String	The character encoding to use in the XML document. The default is UTF-16 for the Unicode version of Siebel applications.
Use Siebel Message Envelope	UseSiebelMessageEnvelope	String	Inserts the Siebel Message Envelope into the XML document. The default is True.
Ignore Character Set Conversion Errors	IgnoreCharSetConv Errors	String	If some characters cannot be represented in the destination character set (like the local codepage), the errors can be ignored. The errors are not ignored by default. For both situations, a warning error entry is created.

Table 17. Integration Object Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
Tags on Separate Lines	Tags on Separate Lines	String	When True, a line feed is placed at the end of each tag. The default is False.
XML Header Text	XMLHeaderText	String	Text to prepend to the beginning of the XML document data.

Table 18 describes the output argument for the Integration Object Hierarchy to XML Document method of the EAI XML Converter.

Table 18. Integration Object Hierarchy to XML Document Method Output Argument

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The resulting XML document.

XML Document to Integration Object Hierarchy Method Arguments

Table 19 describes the input arguments for the XML Document to Integration Object Hierarchy method of the EAI XML Converter.

Table 19. Integration Object Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The input XML document.
Integration Object Name	IntObjectName	String	Name of the Integration Object to use in cases where the Siebel Message envelope might not be present.
Integration Object Lookup Rule Set	IntObjectNameLookupRuleSet	String	Rule Set for the EAI Dispatcher Service for finding out Integration Object Name in cases where the Siebel Message envelope might not be present.
Validate External Entity	ValidateExternalEntity	String	If True, the parser will be set to validate against external metadata, such as DTDs.

Table 19. Integration Object Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
External Entity Directory	ExternalEntityDirectory	String	The directory to use for finding external entities referenced in the XML document, such as DTDs.
Truncate Field Values	TruncateFieldValues	String	If True, truncate any fields longer than their maximum size, as specified in the Integration Component field definition.
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.
Contains Inline Attachments	ContainsInlineAttachments	String	This is True if the file attachment content was included in the original XML document; otherwise it is False. From MIME (Multipurpose Internet Mail Extensions) Converter only.
Tags on Separate Lines	Tags on Separate Lines	String	When True, a line feed is placed at the end of each tag. The default is False.

Table 20 describes the output arguments for the XML Document to Integration Object Hierarchy method of the EAI XML Converter.

Table 20. Integration Object Hierarchy to XML Document Method Output Arguments

Display Name	Name	Data Type	Description
Siebel Message	SiebelMessage	Hierarchy	The Integration Object Hierarchy to be converted to XML.
XML Character Encoding	XMLCharEncoding	String	Character encoding of the XML document, detected by the converter independent of the parser.

XML Hierarchy Converter Methods and Arguments

There are two methods for the XML Hierarchy Converter, as shown in Table 21. The arguments for each method appear in the tables that follow.

Table 21. EAI XML Converter Methods

Display Name	Name	Description
XML Document to XML Hierarchy	XMLDocToXMLHier	Converts an XML document into an XML Hierarchy.
XML Hierarchy to XML Document	XMLHierToXMLDoc	Converts an XML Hierarchy into an XML document.

XML Document to XML Hierarchy Method Arguments

Table 22 describes the input arguments for the XML Document to XML Hierarchy method of the XML Hierarchy Converter.

Table 22. XML Document to XML Hierarchy Method Input Arguments

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The input XML Document.
Escape Names	EscapeNames	String	Invalid characters in XML tags will be escaped, using Siebel's internal escape format. ■ If True, process Escape characters (this is the default). ■ If False, do not process Escape characters.
Validate External Entity	ValidateExternalEntity	String	If True, the parser will be set to validate against external metadata, such as DTD schemas.
External Entity Directory	ExternalEntityDirectory	String	Location of external entity files, such as DTD files.
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.

Table 23 describes the output arguments for the XML Document to XML Hierarchy method of the XML Hierarchy Converter.

Table 23. XML Document to XML Hierarchy Method Output Arguments

Display Name	Name	Data Type	Description
XML Character Encoding	XMLCharEncoding	String	Character encoding of the XML document, detected by the converter, independent of the parser.
XML Hierarchy	XMLHierarchy	Hierarchy	The Output XML hierarchy.

XML Hierarchy to XML Document Method Arguments

Table 24 describes the input arguments for the Integration Object Hierarchy to XML Document method of the XML Hierarchy Converter.

Table 24. XML Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
Escape Names	EscapeNames	String	Invalid characters in XML tags will be escaped, using Siebel's internal escape format. ■ If True, Escape invalid characters (this is the default). ■ If False, delete invalid characters. (Do not use in XML tags.)
XML Character Encoding	XMLCharEncoding	String	Outputs the XML character encoding to use. If encoding is blank or not supported, an error is produced.

Table 24. XML Hierarchy to XML Document Method Input Arguments

Display Name	Name	Data Type	Description
XML Header Text	XMLHeaderText	String	A string in a local codepage character encoding to be inserted before the XML document's root element, after the <?xml...?> declaration. This allows custom processing instructions or an XML header to be inserted before the XML document data starts. For instance, if the header text is `<myheader>data</myheader>` and the XML document output without this parameter is `<?xml version="1.0" encoding="UTF-8"?><account>..</account>`, then the document with the XMLHeaderText included will be: `<?xml version="1.0" encoding="UTF-8"?><myheader>some data</myheader><account>.......</account>`
XML Hierarchy	XMLHierarchy	Hierarchy	The XML hierarchy.
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.
Tags on Separate Lines	Tags on Separate Lines	String	When True, a line feed is placed at the end of each tag. The default is False.

Table 25 describes the output argument for the Integration Object Hierarchy to XML Document method of the XML Hierarchy Converter.

Table 25. XML Hierarchy to XML Document Method Output Argument

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The output XML Document.

EAI Integration Object to XML Hierarchy Converter Methods and Arguments

There are two methods for the EAI Integration Object to XML Hierarchy Converter, as shown in Table 26. The arguments for each method appear in the tables that follow.

NOTE: You can use the XML Hierarchy property sets to manipulate in memory XML hierarchies, such as to add new elements, attributes, or envelopes. An XML Hierarchy property set can be converted to and from an Integration Object property set using EAI Integration Object to XML Hierarchy Converter. An XML Hierarchy property set can be converted to and from an XML document using the XML Hierarchy Converter.

Table 26. EAI Integration Object to XML Hierarchy Converter Methods

Display Name	Name	Description
Integration Object Hierarchy to XML Hierarchy	IntObjHierToXMLHier	Converts an integration object hierarchy to an XML hierarchy.
XML Hierarchy to Integration Object Hierarchy	XMLHierToIntObjHier	Converts an XML hierarchy to an integration object.

Integration Object Hierarchy to XML Hierarchy Method Arguments

Table 27 and Table 28 describe the arguments for the Integration Object Hierarchy to XML Hierarchy method of the EAI Integration Object to XML Hierarchy Converter.

Table 27. Property Set To Integration Object Hierarchy to XML Hierarchy Input Argument

Display Name	Name	Data Type	Description
Namespace	Namespace	String	If a namespace is defined here, it will override any namespace defined in the user properties of an integration object.
Integration Object Hierarchy	SiebelMessage	Hierarchy	The integration object hierarchy to be converted.
Use Siebel Message Envelope	UseSiebelMessageEnvelope	String	If true (the default), the Siebel Message Envelope is used in the XML Hierarchy, otherwise the Siebel Message Envelope is not included.

Table 28. Property Set To Integration Object Hierarchy to XML Hierarchy Output Argument

Display Name	Name	Data Type	Description
XML Hierarchy	XMLHierarchy	Hierarchy	The converted integration object.

XML Hierarchy to Integration Object Hierarchy Method Arguments

Table 29 and Table 30 describe the arguments for the XML Hierarchy to Integration Object Hierarchy method of the EAI Integration Object to XML Hierarchy Converter.

Table 29. Property Set To XML Hierarchy to Integration Object Hierarchy Input Argument

Display Name	Name	Data Type	Description
Contains Inline Attachments	ContainsInlineAttachments	String	TRUE by default. DTYPE_ATTACHMENT fields are assumed to contain actual attachment content. If FALSE, then the field is treated as a reference to an external attachment.
Integration Object Name	IntObjectName	String	Integration Object Name can be specified if the Siebel Message envelope is not present in the XML hierarchy. The service generates the envelope automatically if this parameter is present.
Strip Name Space	StripNamespace	String	Removes the namespace from XML tags.
Truncate Field Values	TruncateFieldValues	String	TRUE by default. If TRUE, truncate any fields longer than their maximum size. If TRUE, report fields that are too long as errors.
XML Hierarchy	XMLHierarchy	Hierarchy	The hierarchy to be converted.

Table 30. Property Set To XML Hierarchy to Integration Object Hierarchy Ouput Argument

Display Name	Name	Data Type	Description
Integration Object Hierarchy	Siebel Message	Hierarchy	The converted integration object.

XML Converter Methods and Arguments

Use the XML Converter when you want to convert any property set to XML, or convert an XML document that is not a Siebel EAI Integration Object Message to a property set.

There are two methods for the XML Hierarchy Converter, as shown in Table 31. The arguments for each method appear in the tables that follow.

Table 31. EAI XML Converter Methods

Display Name	Name	Description
Property Set to XML	PropSetToXML	Converts a property set hierarchy to XML. Returns the result in the Value field of the Output property set.
XML to Property Set	XMLToPropSet	Converts an XML document stored in the Value field of the property set to a property set hierarchy. Returns the result in the Output property set.

Property Set To XML Method Arguments

Table 32 and Table 33 describe the arguments for the Property Set To XML method of the XML Converter.

Table 32. Property Set To XML Method Input Argument

Display Name	Name	Data Type	Description
n/a	Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.	Hierarchical	The entire input property set.

Table 33. Property Set To XML Method Output Argument

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The output XML document.

XML To Property Set Method Arguments

Table 34 and Table 35 describe the arguments for the XML To Property Set method of the XML Converter.

Table 34. XML To Property Set Method Input Argument

Display Name	Name	Data Type	Description
XML Document	< Value >	String	The input XML document.

Table 35. XML To Property Set Method Output Argument

Display Name	Name	Data Type	Description
n/a	Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.	Hierarchical	The entire output property set.

XML Converter Business Service Details

This section describes the EAI XML Write to File and the EAI XML Read from File business services.

Using the EAI XML Write to File Business Service

Use the EAI XML Write to File business service when you want to create an XML document from a property set hierarchy and write the resulting document to a file. This business service supports all XML converters. Table 36 describes the EAI XML Write to File business service's Write EAI Message method.

Table 36. EAI XML Write to File Business Service Methods

Display Name	Name	Description
Write Siebel Message	WriteEAIMsg	Uses the EAI XML Converter
Write XML Hierarchy	WriteXMLHier	Uses the XML Converter
Write Property Set	WritePropSet	Uses the XML Hierarchy Converter

Write Siebel Message Method Arguments

Table 37 describes the input arguments for the Write EAI Message method of the EAI XML Write to File business service.

Table 37. Write Siebel Message Method Input Arguments

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the file where output is to be written. This is a required field.
Siebel Message	Siebel Message	Hierarchy	The Integration Object Hierarchy to be converted to XML.
XML Character Encoding	XMLCharEncoding	String	Character encoding in the XML document. If encoding is blank or not supported, an error is produced.
Use Siebel Message Envelope	UseSiebelMessageEnvelope	String	Insert the Siebel Message Envelope into the XML document. The default is True.

Table 37. Write Siebel Message Method Input Arguments

Display Name	Name	Data Type	Description
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.
Tags on Separate Lines	Tags on Separate Lines	String	When True, a line feed is placed at the end of each tag. The default is False.

Write Property Set Method Arguments

Table 38 describes the input arguments for the Write Property Set method of the EAI XML Write to File business service.

Table 38. Write Property Set Input Arguments

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the file where output is to be written. This is a required field.
n/a	Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.	Hierarchical	The entire input property set.

Write XML Hierarchy Method Arguments

Table 39 describes the input arguments for the Write XML Hierarchy method of the EAI XML Write to File business service.

Table 39. Write XML Hierarchy Method Input Arguments

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the file where output is to be written. This is a required field.
XML Hierarchy	XMLHierarchy	Hierarchy	The XML Hierarchy Property Set.
Escape Names	EscapeNames	String	Invalid characters in XML tags will be escaped, using Siebel's internal escape format. ■ If True, Escape invalid characters (this is the default). ■ If False, delete Escape characters.
XML Character Encoding	XMLCharEncoding	String	Outputs XML character encoding to use. If encoding is blank or not supported, an error is produced.
Ignore Character Set Conversion Errors	IgnoreCharSetConv Errors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.
Tags on Separate Lines	Tags on Separate Lines	String	When True, a line feed is placed at the end of each tag. The default is False.

Using the EAI XML Read from File Business Service

Use the EAI XML Read from File business service when you want to create a property set hierarchy in the Siebel environment from an XML document stored as a file. This business service supports both standard and EAI XML conversion.

Table 40 describes the three EAI XML Read from File business service's methods. The arguments for each method appear in the tables that follow.

Table 40. EAI XML Read from File Business Service Methods

Display Name	Name	Description
Read Siebel Message	ReadEAIMsg	Uses the EAI XML Converter
Read Property Set	ReadPropSet	Uses the XML Converter
Read XML Hierarchy	ReadXMLHier	Uses the XML Hierarchy Converter

Read Siebel Message Method Arguments

Table 41 describes the arguments for the Read Siebel Message method of the EAI XML Read from File business service.

Table 41. Read Siebel Message Method Input Arguments

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the file to be read. This is a required field.
Integration Object Name	IntObjectName	String	Name of the Integration Object to use in cases where the Siebel Message header is not present.
Integration Object Lookup Rule Set	IntObjectLookupRuleSet	String	Rule Set for the EAI Dispatcher Service for finding the Integration Object Name in cases where the Siebel Message header is not present.
External Entity Directory	ExternalEntityDirectory	String	Directory to use for finding external entities referenced in the XML document, such as DTDs.

Table 41. Read Siebel Message Method Input Arguments

Display Name	Name	Data Type	Description
Truncate Field Values	TruncateFieldValues	String	If True, truncate any fields longer than their maximum size. If False, report fields that are too long as errors.
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.

Table 42 describes the output arguments for the Read Siebel Message method of the EAI XML Read from File business service.

Table 42. Read Siebel Message Method Output Arguments

Display Name	Name	Data Type	Description
Siebel Message	SiebelMessage	Hierarchy	The Integration Object Hierarchy converted from XML.
XML Character Encoding	XMLCharEncoding	String	Outputs XML character encoding to use. If encoding is blank or not supported, an error is produced.

Read Property Set Method Arguments

Table 43 and Table 44 describe the arguments for the Read Property Set method of the EAI XML Read from File business service.

Table 43. Read Property Set Method Input Argument

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the file to be read. This is a required field.

Table 44. Read Property Set Method Output Argument

Display Name	Name	Data Type	Description
n/a	Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.	Hierarchical	The entire output property set.

Read XML Hierarchy Method Arguments

Table 45 describes the input arguments for the Read XML Hierarchy method of the EAI XML Read from File business service.

Table 45. Read XML Hierarchy Input Method Arguments

Display Name	Name	Data Type	Description
File Name	FileName	String	The name of the XML file to read. This is a Required field.
Escape Names	EscapeNames	String	Invalid characters in XML tags will be escaped, using Siebel's internal escape format. ■ If True, process Escape characters (this is the default). ■ If False, do not process Escape characters.
External Entity Directory	ExternalEntityDirectory	String	Directory for external entities such as DTD files.
Ignore Character Set Conversion Errors	IgnoreCharSetConvErrors	String	False by default. If the Siebel application cannot represent a given character set—such as the local codepage character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.

Table 46 describes the output arguments for the Read XML Hierarchy method of the EAI XML Read from File business service.

Table 46. Read XML Hierarchy Method Output Arguments

Display Name	Name	Data Type	Description
XML Character Encoding	XMLCharEncoding	String	Character encoding of the XML document, detected by the converter independent of the parser.
XML Hierarchy	XMLHierarchy	Hierarchy	The XML Hierarchy property set.

XML Integration Scenarios 7

To help you implement the use of XML technologies at your organization, this chapter gives you three business scenarios. These scenarios detail the steps involved in creating the following two types of XML integrations:

- An integration using Siebel XML
- An integration using an external XML document that uses an XSD or a DTD

These scenarios provide high-level overviews of the procedures. Specifics on how to complete a process can be found elsewhere in the Siebel documentation set.

Scenario 1: Integration Using Siebel XML

This scenario presents general steps for setting up an inbound integration using XML.

Designing the Integration

For an inbound Siebel XML integration, you complete two major steps:

- Use the Generate Schema wizard in Siebel Tools to create an XSD or a DTD for the incoming XML.

  For details, see "To create the Siebel integration object" on page 88.

- Create a new workflow.

  For details, see "To create a new workflow" on page 86.

To create the XML schema: XSD, DTD, or XDR

1. Start Siebel Tools and navigate to the Integration Objects list.
2. Select an integration object from the list.

3. Click the Generate Schema button at the top of the Integration Objects list.

   The Generate Schema wizard appears.

4. Complete the steps of the wizard:

   a. Select a business service from the Business Service drop-down list.

   b. Select “EAI Siebel Message Envelope Service” from the Envelope drop-down list.

   c. Browse to a file location and type a file name to generate the schema—for example, “`ListOfSiebelOrder.xml`”—and click Save.

   NOTE: For information on using the Siebel DTD Wizard, see *Transports and Interfaces: Siebel eBusiness Application Integration Volume III, MidMarket Edition*.

   ---

5. Load the schema into the external system.

To create a new workflow

1. Start a Siebel application and navigate to the Workflow Process Designer.

2. Create a new workflow that will take the XML file, convert it to Siebel XML format (if necessary) using the Siebel EAI XML Converter business service, call the EAI Data Transformation Engine to perform the data transformation, and call the Siebel Adapter to modify the Siebel database as needed (upsert, delete, query, and so on).

   NOTE: The Siebel application uses an instance of the integration object you created to map the incoming XML data to fields (rows and columns) within the Siebel database.

   ---

3. Test your workflow using the Workflow Process Simulator.

4. Save your workflow.

NOTE: For information on creating and testing workflows, see *Siebel Business Process Designer Administration Guide, MidMarket Edition*.

Running the Integration

In this scenario, assume that either an external application has generated Siebel XML that requires no translation or Siebel XML is XML that conforms to the Siebel XSD or DTD.

At runtime, the Siebel application:

- Calls the EAI XML Adapter.
- Calls the EAI XML Converter to convert the incoming XML to a Siebel message.
- Calls the Siebel Adapter and updates the Siebel database with the new information just received from the incoming (external) XML document.

Scenario 2: Integration Using External XML and an XSD or DTD

This scenario presents general steps for setting up an integration based on incoming XML which has been defined in an XSD or a DTD.

Designing the Integration

For an outbound Siebel XML integration, you complete three high-level procedures:

- Create a new external Siebel integration object.

  For details, see "To create the Siebel integration object" on page 88.

- Use Siebel Data Mapper to map the fields in the external Integration Object with an internal Siebel Integration Object.

  For details, see "To map the data" on page 89.

- Create a new workflow.

  For details, see "To create a new workflow" on page 86.

To create the Siebel integration object

1. Start Siebel Tools and select File > New Object.

   The New Object window appears.

2. Select the EAI tab.
3. Double-click the Integration Object icon.

   The Integration Object Builder wizard appears.

4. Complete the Integration Object Builder initial page:
   a. Select the Siebel project from the first drop-down list.
   b. Select EAI XSD or EAI DTD Wizard as the Business Service.
   c. Navigate to the path and file of the location of the XSD, DTD, or XML file that you want to use as the basis of the DTD.

   NOTE: For information on using the Integration Object Wizard, see *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II, MidMarket Edition*.

5. Save the new integration object.

To map the data

1. Start a Siebel application and navigate to the Siebel Data Mapper.
2. Create the data mapping between the external integration object and an internal Siebel integration object.
3. Save the mapping.

   The new data mapping rules are now in the Siebel database.

NOTE: For information on using the Siebel Data Mapper and on the Data Transformation Engine (DTE), see *Business Processes and Rules: Siebel eBusiness Application Integration Volume IV, MidMarket Edition*.

Running the Integration

In this scenario, assume that the external application has generated external XML and includes an associated XSD or a DTD.

At runtime, the Siebel application:

- Calls the EAI XML Converter to convert incoming XML to a Siebel Message.
- Calls the EAI Data Mapping Engine to transform the external integration object to an internal integration object.
- Calls the Siebel Adapter and updates the Siebel database with the new information just received from the incoming (external) XML document.

Using XML Files A

This appendix discusses using XML files.

Using an XML Document as Input

You can use XML documents as input in a workflow, by calling business services to convert them to Siebel Property Sets and calling business services to process the data from XML documents as required. Figure 7 illustrates a sample workflow that uses the Siebel Adapter Insert or Update method.

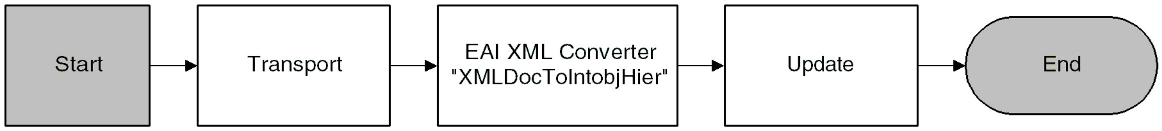


Figure 7. Workflow Using Siebel Adapter with Upsert Method

The following is an example of a sample XML document containing employee information that will get “upserted” by the Siebel adapter in the previous workflow. Just before the Siebel adapter step in the workflow is invoked, the variable Employee Message will contain the XML document in a hierarchical format.

```
<SiebelMessage MessageId="" IntObjectName="Sample Employee">

  <ListOfSampleEmployees>

    <Employee>

      <FirstName>Pace</FirstName>

      <MiddleName></MiddleName>

      <LastName>Davis</LastName>

      <LoginName>ADIOTATI</LoginName>
      <PersonalTitle>Mr.</PersonalTitle>
      <EMailAddr>pdavis@pcssiebel.com</EMailAddr>
      <JobTitle>Field Sales Representative</JobTitle>
      <Phone>4153296500</Phone>
      <Private>N</Private>
        <ListOfPosition>
          <Position>
            <Name3>Field Sales Representative - S America</Name3>
            <Division>North American Organization</Division>
            <Organization>North American Organization</Organization>
            <ParentPositionName>VP Sales</ParentPositionName>
            <PositionType>Sales Representative</PositionType>
              <ListOfPosition_BusinessAddress>
                <Position_BusinessAddress>
                <City>San Mateo</City>
                <Country>USA</Country>
                <FaxNumber></FaxNumber>
                <PhoneNumber></PhoneNumber>
                <PostalCode>94175</PostalCode>
                <State>CA</State>
                <StreetAddress>1855 South Grant St</StreetAddress>
              </Position_BusinessAddress>
            </ListOfPosition_BusinessAddress>
          </Position>
      </ListOfPosition>
    </Employee>
  </ListOfSampleEmployees>
</SiebelMessage>
```

This EAI XML document shows an integration object called “Sample Employee” as specified by the IntObjectName attribute of the Siebel Message element.

The Sample Employee object has three integration components you can view using Siebel Tools:

- Employee—A root component
- Position—A Child of Employee
- Position Business Address—A Child of Position

An upsert to this integration object is determined by the user key on the root component. In the Sample Employee Integration object provided as part of the sample database, the user key for the Employee integration object is Login name. Therefore, if the login name is unique, a new employee is inserted. If the system finds the login name already in the Siebel eBusiness Applications, then it would perform an update. The above XML document will create a new employee whose name is Pace Davis and assign the position “Field Sales Representative - S America” to this person. You could also specify a new position and have the employee be assigned to the new position. This can be extended to other methods like “Delete” or “Query.” If you want to delete an employee, the user key is the only element that needs to be specified.

Example. In the following example, the employee with login name ADD1 will be deleted.

```
<SiebelMessage MessageId="" IntObjectName="Sample Employee">

      <ListOfSampleEmployees>

         <Employee>

            <LoginName>ADD1</LoginName>

         </Employee>

      </ListOfSampleEmployees>

   </SiebelMessage>
```

Example. Query on all employees with the first name Pace and Last name starting with D.

```
<SiebelMessage MessageId="" IntObjectName="Sample Employee">

     <ListOfSampleEmployees>

        <Employee>

           <FirstName>Pace</FirstName>

           <LastName>D*</LastName>

        </Employee>

     </ListOfSampleEmployees>

  </SiebelMessage>
```

CAUTION: When defining these business components, be aware that the precise definition can greatly impact mobile clients and regional clients. There are setup options to allow all attachments to automatically download to mobile clients that have visibility to the underlying row. This could be quite problematic, especially for large files.

The preferred setup is “demand mode,” whereby mobile client users trying to open an attachment will see a message asking if they want to download the file the next time they synchronize. This is known as the “deferred” approach and gives users control over what files they do or do not download.

Inserting File Attachments Using XML

There may be times when you have an attachment that you want to insert into the Siebel database, such as an image file in JPEG format. This could be a customer's picture, a site picture, an item or part image, a text document, and so on. To send the file to the Siebel database, you would include the text in the body of the XML file (recoding it in base64 if it is binary). Then, using the HTTP Transport, for example, and various EAI adapters, you would send that XML file to the Siebel database.

This section explains the process for inserting XML documents with attachments into the Siebel database.

To insert a file into the Siebel database using XML

1. If the file is binary or in a different character set from the XML document, first encode it in Base64 format.
2. Insert this text string into an XML document as an element value. There should be an attribute called AttachmentIsTextData (see Step 4 below).
3. Create a workflow that calls the EAI XML Converter and then the Siebel EAI Adapter. You can also use the EAI Dispatch Service. Set where you want the data to be stored in the Siebel database (optional) as well as other parameters specific to the situation.
4. The EAI XML Converter reads the incoming XML file and parses it. One of the items the EAI XML Converter searches for in the incoming XML document is a property called AttachmentIsTextData.
   - If AttachmentIsTextData is set to False, then the EAI XML Converter will do a reconversion from the base64 encoding format to binary using a base64 decoding algorithm.
   - If AttachmentIsTextData is set to True, then the EAI XML Converter reads the string as simply text data, and no conversion takes place.

5. The EAI Siebel Adapter updates the Siebel database and file system according to Step 4.

   - The EAI Siebel Adapter uploads the data into the Siebel database.
   - The file is saved in the Siebel file system. (The information for the path referred to in the file, if there is one, is also saved in Siebel database.)

NOTE: You can either choose to have the system insert the file into a default location in the Siebel file system or you can specify a directory that will store all incoming binary files. Siebel application names the file using a combination of the underlying table name and the row ID of the associated record in the business component attachment table. The metadata record (the *_ATTT table) stores the *actual* name of the file.

Index