Copyright © 2005, 2006, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

PRODUCT MODULES AND OPTIONS. This guide contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel’s Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.
Contents

Chapter 1: What’s New in This Release

Chapter 2: Overview
Siebel EAI and XML 9
  XML Integration Objects 10
  Bidirectional Data Flow 10
Metadata Support for XML 10
Special Characters in XML Documents 11
  Special (Escape) Characters 11
  Declaring the Character Set in Use 11

Chapter 3: XML Representation of Property Sets
Mapping Between Property Sets and XML 13
Element and Attribute Naming 13
Property Set Examples and Their XML Representation 15
Properly Formatted Property Sets 16

Chapter 4: XML Representation of Integration Object Instances
Integration Objects 19
Elements and Attributes 20
How XML Names Are Derived from Integration Objects 21
Elements Within a Siebel Integration Object Document 21
  SiebelMessage Element 22
  Object List Element 22
  Integration Component Elements 23
  Component Container Elements 24
  Integration Field Elements 24
Example XML Document 24
XML Schema Definitions (XSDs) 26
Document Type Definitions (DTDs) 26
## Chapter 5: XML Integration Objects and the XSD Wizard

Creating XML Integration Objects with the XSD Wizard 29

- Selecting Source Object in the XSD Wizard 30

Supported XSD Elements and Attributes 30

- Structure of XSD XML Integration Object 36
  - XSD Specific Integration Object Properties 36
  - XSD Specific Integration Object User Properties 36
  - XSD Specific Integration Component Properties 37
  - XSD Specific Integration Component User Properties 37
  - XSD Specific Integration Component Field Properties 37
  - XSD Specific Integration Component Field User Properties 38

## Chapter 6: XML Integration Objects and the DTD Wizard

Creating XML Integration Objects with the DTD Wizard 39

- Selecting Source Object in DTD Wizard 40

How the DTD Wizard Creates XML Integration Objects 40

- Elements 40
- Attributes 41
- Element’s #PCDATA 41
- Names 41
- Hierarchy 42
- Connectors 42
- Cardinality 42

## Chapter 7: Siebel XML Converters

EAI XML Converter 46

XML Hierarchy Converter 46

EAI Integration Object to XML Hierarchy Converter 48

XML Converter 48

Siebel XML Converter Business Services Comparison 49

EAI XML Converter Business Service 50

- Integration Object Hierarchy to XML Document Method Arguments 51
- XML Document to Integration Object Hierarchy Method Arguments 52

XML Hierarchy Converter Business Service 54

- XML Document to XML Hierarchy Method Arguments 55
- XML Hierarchy to XML Document Method Arguments 56

EAI Integration Object to XML Hierarchy Converter Business Service 58
Integration Object Hierarchy to XML Hierarchy Method Arguments 59
XML Hierarchy to Integration Object Hierarchy Method Arguments 60

XML Converter Business Service 61
  Property Set To XML Method Arguments 61
  XML To Property Set Method Arguments 62
EAI XML Write to File Business Service 62
EAI XML Read from File Business Service 65

Chapter 8: XML Integration Scenarios
Scenario 1: Integration Using Siebel XML 71
  Designing the Integration 71
  Running the Integration 72
Scenario 2: Integration Using External XML and an XSD or DTD 72
  Designing the Integration 72
  Running the Integration 73

Appendix A: Using XML Files
Using an XML Document as Input 75
Inserting File Attachments Using XML 77

Index
What’s New in XML Reference: Siebel Enterprise Application Integration, Version 8.0

This document has been updated to reflect product name and user interface changes. This content was previously published as XML Reference: Siebel Enterprise Application Integration Volume V Version 7.5.3.

For more information about changes to the user interface, see Siebel Fundamentals.
What’s New in This Release
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 while HTML explains 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 Oracle’s Siebel eScript, for example.

There are also technologies that explain 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 EAI 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 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, and Oracle application data—can be represented as common structures that the EAI infrastructure can understand: Integration Objects.

Siebel can also communicate bidirectionally with Web Services using Simple Object Application Protocol (SOAP) XML. For details, see Integration Platform Technologies: Siebel Enterprise Application Integration.

**NOTE:** If you do a minimal client installation, 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 installation.
**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 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.

![Diagram of bidirectional data flow](image)

Figure 1. Document to Integration-Object Flow

**NOTE:** For details on integration object and Web Services, see *Integration Platform Technologies: Siebel Enterprise Application Integration*. For an overview of Siebel EAI, see *Overview: Siebel Enterprise Application Integration*.

**Metadata Support for XML**

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.

### 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).

**NOTE:** To edit an XML document including binary or encoded data, use editors such as Notepad or Word that do not convert the data upon saving the file.

### 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 (;). Also, if the XML is passed in a URL, then URL encoding of special characters is required as shown in Table 1.

<table>
<thead>
<tr>
<th>Character</th>
<th>Entity</th>
<th>URL Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>&lt;</td>
<td>%26lt%3B</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>&gt;</td>
<td>%26gt%3B</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>&amp;</td>
<td>%26amp%3B</td>
</tr>
<tr>
<td><code>&quot;</code></td>
<td>&quot;</td>
<td>%26quot%3B</td>
</tr>
<tr>
<td><code>'</code></td>
<td>'</td>
<td>%26apos%3B</td>
</tr>
<tr>
<td>Unicode Character (Decimal)</td>
<td>	</td>
<td>%26%2309%3B</td>
</tr>
<tr>
<td>Unicode Character (Hex)</td>
<td>°</td>
<td>%26%23x00B0%3B</td>
</tr>
<tr>
<td>Date</td>
<td>Must follow the ISO 8601 format</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### 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
<?xml version="1.0" encoding="US-ASCII"?>
```
Supported character sets include but are not limited to ASCII, UTF-8, UTF-16 (Big or Small Endian), UCS4 (Big or Small Endian), EBCDIC code pages IBM037 and IBM1140 encodings, ISO-8859-1, and Windows-1252. This means that the XML parser can parse input XML files in these encodings.

The followings are the corresponding encodings names to be used in the XML declaration:


Also, the output can be in one of the following XML encodings:

UTF-8, UTF-16, or local code page

The character set declaration encoding must appear after the version declaration. For example:

```xml
<?xml version="1.0" encoding="US-ASCII"?>
```
This chapter discusses the XML representation of property sets and the mapping between property sets and XML. It also discusses the elements and attributes naming conversion performed by the XML Converter.

**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.

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>
<thead>
<tr>
<th>Property Set Component</th>
<th>XML Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySet</td>
<td>Element</td>
</tr>
<tr>
<td>PropertySet Type</td>
<td>Element name (If Type is not specified, the element name is set to PropertySet)</td>
</tr>
<tr>
<td>PropertySet Value</td>
<td>Element Character Data</td>
</tr>
<tr>
<td>Property name</td>
<td>Attribute name</td>
</tr>
<tr>
<td>Property value</td>
<td>Attribute value</td>
</tr>
<tr>
<td>Child Property Set</td>
<td>Child element</td>
</tr>
</tbody>
</table>

**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 include 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`. 
**True.** 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**

<table>
<thead>
<tr>
<th>Character in Property Set</th>
<th>Description</th>
<th>Generated Escape Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space</td>
<td>_spc</td>
</tr>
<tr>
<td>_</td>
<td>Underscore</td>
<td>_und</td>
</tr>
<tr>
<td>`</td>
<td>Double Quote</td>
<td>_dqt</td>
</tr>
<tr>
<td>’</td>
<td>Single Quote</td>
<td>_sqt</td>
</tr>
<tr>
<td>;</td>
<td>Colon</td>
<td>_cln</td>
</tr>
<tr>
<td>;</td>
<td>Semicolon</td>
<td>_scn</td>
</tr>
<tr>
<td>(</td>
<td>Left Parenthesis</td>
<td>_lpr</td>
</tr>
<tr>
<td>)</td>
<td>Right Parenthesis</td>
<td>_rpr</td>
</tr>
<tr>
<td>&amp;</td>
<td>Ampersand</td>
<td>_amp</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
<td>_cma</td>
</tr>
<tr>
<td>#</td>
<td>Pound symbol</td>
<td>_pnd</td>
</tr>
<tr>
<td>/</td>
<td>(Forward) slash</td>
<td>_slh</td>
</tr>
<tr>
<td>?</td>
<td>Question Mark</td>
<td>_qst</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
<td>_lst</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
<td>_grt</td>
</tr>
<tr>
<td></td>
<td>Other illegal characters not listed above</td>
<td>_&lt;Unicode character code&gt;</td>
</tr>
</tbody>
</table>

**False.** 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.
Property Set Examples and Their XML Representation

The following is examples of different types of property sets that are available and their XML representation:

An Arbitrary Property Set

```xml
<?Siebel-Property-Set> <PropertySet> <Person> Jack </Person> </PropertySet>
```

A Siebel Message

```xml
<?Siebel-Property-Set EscapeNames="true"> <PropertySet> <SiebelMessage MessageID="1-111" IntObjectFormat="Siebel Hierarchical" MessageType="Integration Object" IntObjName="Sample Account"> <ListOfSample_spcAccount> ... </ListOfSample_spcAccount> </SiebelMessage> </PropertySet>
```

An XML Hierarchy

```xml
<?Siebel-Property-Set> <PropertySet> _XMLHierarchy <Account> <Contact> ... </Contact> </Account> _XMLHierarchy </PropertySet>
```
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.

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. This includes necessary arguments and values. An example of such code is:

```
Set Inputs = TheApplication.NewPropertySet
```

Properly Formatted Property Sets
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

csvc.InvokeMethod "PropSetToXML", XMLInputs, XMLOutputs
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 Enterprise Application Integration*.

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](image-url)
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canonical section</td>
<td>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.</td>
</tr>
<tr>
<td>External section</td>
<td>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.</td>
</tr>
<tr>
<td>XML section</td>
<td>Stores the mapping between an integration object definition and its XML representation. This allows any integration object to be represented as XML.</td>
</tr>
</tbody>
</table>

### 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:

```xml
<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.

```xml
<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.

```xml
<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. However, Siebel repository names can include characters not permitted in an XML name, such as blank spaces. Thus, some translation must 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 must have an understanding of its XML structure, including elements and attributes. The document can may include 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 includes 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 runtime through arguments to the EAI XML Converter Business Service.

Since the Object List element is optional, SiebelMessage can include a Root component element to allow cases when the Object List element is left blank (omitted). For details on Object List element, see “Object List Element” on page 22.

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 includes 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 ListOfName, 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 include 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:

```xml
<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 include 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 includes 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 includes 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.

```xml
<SiebelMessage MessageId=""
IntObjectName="Sample Account">
<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>
      </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:

```xml
  <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.

**NOTE:** Attachment attributes are not supported in DTD because they are not part of the integration object definition and only appear at runtime.

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:

```xml
<!ELEMENT Name (#PCDATA) >
<!ELEMENT Location (#PCDATA) >
<!ELEMENT Division (#PCDATA) >
<!ELEMENT Description (#PCDATA) >
<!ELEMENT CurrencyCode (#PCDATA) >
<!ELEMENT StreetAddress (#PCDATA) >
<!ELEMENT State (#PCDATA) >
<!ELEMENT PostalCode (#PCDATA) >
```
NOTE: All fields are optional, but if present, they must appear in the correct order. The definition of a field appears only once at the beginning of the DTD, even if its XML tag appears in multiple components. 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.
This chapter discusses the XSD wizard, the supported XSD elements and attributes, and the structure of the XSD XML integration object, such as user properties.

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. Launch Siebel Tools.
2. Select File > New Object.
3. In the New Object Wizards window, select the EAI tab.
4. Double-click the Integration Object icon.
5. 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. Navigate to the location of the XSD or XML file that you want to use as the basis of the XSD and click Next.

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

6. Select the source object, give it a unique name, and then click Next.
7. Click on the plus sign to expand the list and select or clear the fields you need from the component.
8. Click Next to get to the final page to review the messages generated during the process and take necessary action as required.
XML Integration Objects and the XSD Wizard

9 Click Finish to complete the process.

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

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

Selecting Source Object in the XSD Wizard

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.

Supported XSD Elements and Attributes

Not all the XSD schema elements and attributes are supported by the Siebel application. Table 5 and Table 6 list all the XSD elements and attributes with Siebel support levels for them. Following is the terminology used in these tables:

- **Ignored.** This level of support means that processing will continue, and an error is not generated. However, the information given for the specified element or attribute is ignored.
- **Mapped.** This level of support means that the information specified in a given element or attribute is used in the integration object representation.
XML Integration Objects and the XSD Wizard ■ Supported XSD Elements and Attributes

- **Not mapped.** This level of support means that the given element or attribute information is not used. However, children of the element will be processed.

**NOTE:** The Siebel application does not perform any formatting or processing for any of the schema types. All the scalar types such as string, ID, or integer are treated as strings. When converted to an integration object and integration component field, DataType is set to DTYPE_TEXT.

Table 5. XSD Schema Elements and Siebel Support Level

<table>
<thead>
<tr>
<th>Elements</th>
<th>Siebel Support level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Not mapped. Treated as sequence.</td>
<td></td>
</tr>
<tr>
<td>annotation</td>
<td>Mapped.</td>
<td>Mapped as a parent’s <code>comment</code> property. Children may be mapped only if parent of annotation is mapped to a component or field.</td>
</tr>
<tr>
<td>any</td>
<td>Mapped.</td>
<td>Mapped as a XML Hierarchy if <code>namespace</code> attribute cannot be resolved to a schema import definition. Otherwise, all global elements logically replace the <code>any</code> element that are then mapped to an integration object using rules for elements. Acts as a placeholder for any element. For more information about this element, see <em>Integration Platform Technologies: Siebel Enterprise Application Integration</em>.</td>
</tr>
<tr>
<td>anyAttribute</td>
<td>Mapped.</td>
<td>Same as the <code>any</code> element. Act as any placeholder for any attribute. For more information about this element, see <em>Integration Platform Technologies: Siebel Enterprise Application Integration</em>.</td>
</tr>
<tr>
<td>appinfo</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>attribute</td>
<td>Mapped.</td>
<td>Mapped as a field. Storing type information is useful when generating schema either after importing one or manually creating one. Also, useful for type specific formatting, such as <code>xsd:datetime</code>.</td>
</tr>
<tr>
<td>attributeGroup</td>
<td>Mapped.</td>
<td>Mapped as children attributes that are added as fields to the parent element’s component.</td>
</tr>
<tr>
<td>choice</td>
<td>Not mapped. Treated as sequence.</td>
<td></td>
</tr>
</tbody>
</table>
### Supported XSD Elements and Attributes

Table 5. XSD Schema Elements and Siebel Support Level

<table>
<thead>
<tr>
<th>Elements</th>
<th>Siebel Support level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>complexContent</td>
<td>Mapped.</td>
<td>Mapped to add properties and children to the parent element's component. Attributes can affect parent (complexType) and children when restriction and extension are processed.</td>
</tr>
<tr>
<td>complexType</td>
<td>Mapped.</td>
<td>Mapped if global complexType is starting point for integration object that maps to root component. Also mapped when XSDTypeName and XSDTypeNamespace user properties are set on the root or elements component.</td>
</tr>
<tr>
<td>documentation</td>
<td>Mapped.</td>
<td>Mapped if Comment property is on a field, component, or object.</td>
</tr>
<tr>
<td>element</td>
<td>Mapped.</td>
<td>Mapped as a component or field. If element is of simpleType and maxOccurs is at most 1, then map to field, otherwise map to component (complexType).</td>
</tr>
<tr>
<td>enumeration</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>extension</td>
<td>Mapped.</td>
<td>Mapped if merging base type and children into the parent. Extension element affects the parent for complexContent and simpleContent.</td>
</tr>
<tr>
<td>field</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>group</td>
<td>Mapped.</td>
<td>Mapped if adding children to the parent element's component.</td>
</tr>
<tr>
<td>import</td>
<td>Mapped.</td>
<td>Preprocessed to receive the additional schema. Resolve a schemaLocation reference by URI or Local (File). Whatever is defined in imported schema will belong to a different namespace.</td>
</tr>
<tr>
<td>include</td>
<td>Supported.</td>
<td>Preprocessed to receive the additional schema. Resolve a schemaLocation reference by URL or Local (File). Whatever is defined in imported schema can belong to the same namespace.</td>
</tr>
<tr>
<td>key</td>
<td>Ignored.</td>
<td>Defines a unique key.</td>
</tr>
</tbody>
</table>
Table 5. XSD Schema Elements and Siebel Support Level

<table>
<thead>
<tr>
<th>Elements</th>
<th>Siebel Support level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyref</td>
<td>Ignored.</td>
<td>Defines fields for key. Keyref refers to a key that must exist in the document.</td>
</tr>
<tr>
<td>length</td>
<td>Mapped. Does not support lists.</td>
<td>Mapped for field external length and length. Fixed length of string-based content. Also might mean length of a list (number of items).</td>
</tr>
<tr>
<td>list</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>maxLength</td>
<td>Mapped.</td>
<td>Mapped for field external length and length.</td>
</tr>
<tr>
<td>minExclusive, maxExclusive</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>minInclusive, maxInclusive</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>minLength</td>
<td>Mapped.</td>
<td>Mapped for field external length and length.</td>
</tr>
<tr>
<td>notation</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>pattern</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>redefine</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>Mapped.</td>
<td>Mapped when adding children to the parent component or field. Affects its parent: complexContent, simpleContent, simpleType. Remove the elements and attributes that are not specified as the restriction ones. Validate that the elements and attributes used in the restriction are present in the base type.</td>
</tr>
<tr>
<td>schema</td>
<td>Mapped.</td>
<td>Namespace information used for object, component, and field.</td>
</tr>
<tr>
<td>selector</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>sequence</td>
<td>Not mapped.</td>
<td></td>
</tr>
<tr>
<td>simpleContent</td>
<td>Mapped.</td>
<td>Mapped when adding properties and children to the parent element's component.</td>
</tr>
<tr>
<td>simpleType</td>
<td>Mapped.</td>
<td>XSDTypeName and XSDTypeNamespace user properties on parent element's field or component, or attribute's field.</td>
</tr>
</tbody>
</table>
### Table 5. XSD Schema Elements and Siebel Support Level

<table>
<thead>
<tr>
<th>Elements</th>
<th>Siebel Support Level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>union</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>unique</td>
<td>Ignored.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6. XSD Schema Attributes and Siebel Support Level

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Siebel Support Level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>attributeFormDefault</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>base</td>
<td>Mapped.</td>
<td>Mapped if base type is used to create component or field.</td>
</tr>
<tr>
<td>block</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>blockDefault</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>default: attribute</td>
<td>Mapped.</td>
<td>Mapped to XMLLiteral value property only. Provides default value for an attribute when an attribute is missing.</td>
</tr>
<tr>
<td>default: element</td>
<td>Mapped.</td>
<td>Mapped to XMLLiteral value property only. Provides default value for an element when an element is empty.</td>
</tr>
<tr>
<td>elementFormDefault</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>final</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>finalDefault</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>fixed: attribute or element</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>fixed: simpleType</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>form</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>itemType</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>maxOccurs</td>
<td>Mapped.</td>
<td>Maps to the cardinality upper bound on parent element's component. Maps to One or More (unbounded). If you want to preserve the maximum number of occurrences, then new column is needed.</td>
</tr>
<tr>
<td>memberTypes</td>
<td>Ignored.</td>
<td></td>
</tr>
</tbody>
</table>
### Supported XSD Elements and Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Siebel Support Level</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>minOccurs</td>
<td>Mapped.</td>
<td>Maps to the cardinality lower bound on parent element’s component. Maps to Zero or One. If you want to preserve the minimum number of occurrences, then new column is needed.</td>
</tr>
<tr>
<td>mixed</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Mapped.</td>
<td>Maps to the XML Tag of parent element (component, field) or attribute field or to the XSD Type Name on object, component, or field. Name of the schema component.</td>
</tr>
<tr>
<td>namespace: import</td>
<td>Mapped.</td>
<td>Maps to Namespace and XSDNamespace user property on components and fields that are being imported. Namespace for the imported elements and attributes.</td>
</tr>
<tr>
<td>nillable</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>processContents</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>public</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>ref</td>
<td>Mapped.</td>
<td>Mapped if metadata starting from global element or attribute that is being referred to is copied to the referring element (component, field) or attribute field.</td>
</tr>
<tr>
<td>schemaLocation</td>
<td>Mapped.</td>
<td>Mapped if used for preprocessing of import or include</td>
</tr>
<tr>
<td>substitutionGroup</td>
<td>Ignored.</td>
<td></td>
</tr>
<tr>
<td>targetNamespace</td>
<td>Mapped.</td>
<td>Maps to XSD Type Namespace and XML Tag Namespace user properties on the integration object, imported component, or field. Schema targetNamespace to which all schema components definitions in a particular schema belong (children of schema element).</td>
</tr>
<tr>
<td>type</td>
<td>Mapped.</td>
<td>Maps to XSDTypeName user property on element’s component or field, or attribute’s field.</td>
</tr>
<tr>
<td>use</td>
<td>Ignored.</td>
<td></td>
</tr>
</tbody>
</table>
Structure of XSD XML Integration Object

The structure of the XSD XML integration object is same as any other integration object. In this chapter we discuss the properties specific to the XSD XML integration object.

**NOTE:** For details on integration objects, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

XSD Specific Integration Object Properties

Table 7 lists the integration object property that is used to represent XSD as an XML integration Object.

Table 7. Integration Object Properties for Representing XSD

<table>
<thead>
<tr>
<th>Name</th>
<th>Project</th>
<th>Base Object Type</th>
<th>XML Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the integration object. The value is provided through the wizard.</td>
<td>The project that the integration object is built in. The value is provided through the wizard.</td>
<td>XML</td>
<td>XML Tag used to represent the integration object.</td>
</tr>
</tbody>
</table>

XSD Specific Integration Object User Properties

Table 8 lists integration object user properties for representing XSD as an XML integration object.

Table 8. Integration Object User Properties for Representing XSD

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLTagNamespace</td>
<td>targetNamespace</td>
<td>Namespace for the Element XML tags.</td>
</tr>
<tr>
<td>XSDTypeName</td>
<td>Name of the root complexType</td>
<td>Name of the root complexType used to create the integration object. This is only used through WSDL Import.</td>
</tr>
<tr>
<td>XSDTypeNamespace</td>
<td>targetNamespace</td>
<td>The namespace URI of the root complex type. This is only used through WSDL Import.</td>
</tr>
</tbody>
</table>
**XSD Specific Integration Component Properties**

Table 9 lists the integration component property for representing XSD as an XML integration component.

### Table 9. Integration Component Properties for Representing XSD

<table>
<thead>
<tr>
<th>External Name Context</th>
<th>Name</th>
<th>External Name</th>
<th>External Sequence</th>
<th>Cardinality</th>
<th>XML Tag</th>
<th>XML Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath to the schema component starting with the global element.</td>
<td>XML Tag plus a sequence number to make component name unique within the integration object.</td>
<td>Element name</td>
<td>XML Sequence</td>
<td>Based on minOccurs or maxOccurs.</td>
<td>Element Name</td>
<td>Sequence within parent element</td>
</tr>
</tbody>
</table>

**XSD Specific Integration Component User Properties**

Table 10 lists integration component user properties for representing XSD as an XML integration component.

### Table 10. Integration Component User Properties for Representing XSD

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLTagNamespace</td>
<td>targetNamespace</td>
<td>Namespace for the Element XML tags.</td>
</tr>
<tr>
<td>XSDTypeName</td>
<td>Component element type attribute</td>
<td>Type of the element. For instance, for element type=&quot;xyz&quot;, XSDTypeName=xyz.</td>
</tr>
<tr>
<td>XSDTypeNamespace</td>
<td>NamespaceURI for element type</td>
<td>Namespace for the element type. This is the Namespace URI for the element's type name.</td>
</tr>
</tbody>
</table>

**XSD Specific Integration Component Field Properties**

Table 11 lists the integration component field property for representing XSD as an XML integration component.

### Table 11. Integration Component Field Properties for Representing XSD

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Length</th>
<th>External Name</th>
<th>External Length</th>
<th>XML Literal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Tag</td>
<td>DTYPE_TEXT</td>
<td>maxLength or length</td>
<td>Attribute or element name</td>
<td>Length</td>
<td>fixed or default</td>
</tr>
</tbody>
</table>
XSD Specific Integration Component Field User Properties

Table 12 lists integration component field user properties for representing XSD as an XML integration component.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLTagNamespace</td>
<td>targetNamespace</td>
<td>Namespace for element or attribute XML tags.</td>
</tr>
<tr>
<td>XSDTypeName</td>
<td>Field element or attribute XML Schema type name</td>
<td>Type of the element or attribute. For instance, for element type =&quot;xyz&quot;, XSDTypeNamespace=xyz.</td>
</tr>
<tr>
<td>XSDTypeNamespace</td>
<td>NamespaceURI for element or attribute type</td>
<td>Namespace for the element or attribute type. In effect, this is the Namespace URI for the element’s or attribute’s type name.</td>
</tr>
</tbody>
</table>
This chapter discusses the DTD wizard and how it creates XML integration objects.

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

   **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.

5. Select the source object and give it a unique name, and then click Next.
6. Click on the plus sign to expand the list and select or clear the fields based on your business requirements.
7. Click Next to go to the final page to review messages generated during this process and take necessary action.
Selecting Source Object in DTD Wizard

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, and subscriptions. 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.

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+)>```

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

```xml
<!ELEMENT car>
<!ATTLIST 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>`.
Hierarchical Relationship

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.

<table>
<thead>
<tr>
<th>Connector</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>,</td>
<td>followed by</td>
<td>(a,b)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one or the other</td>
</tr>
</tbody>
</table>

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

As shown in Table 13, 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. 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>
<thead>
<tr>
<th>DTD Element Occurrence Operator</th>
<th>Description</th>
<th>Integration Component Cardinality Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Appears one time</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>?</td>
<td>Appears 0 or 1 time</td>
<td>Zero or One</td>
<td>Appears 0 or 1 time</td>
</tr>
<tr>
<td>+</td>
<td>Appears 1 or more times</td>
<td>One or More</td>
<td>Appears 1 or more times</td>
</tr>
<tr>
<td>*</td>
<td>May appear 0 or more times</td>
<td>Zero or More</td>
<td>May appear 0 or more times</td>
</tr>
<tr>
<td>No modifier</td>
<td>Appears one time</td>
<td>One</td>
<td>Appears one time</td>
</tr>
</tbody>
</table>
The specification for DTDs supports using parentheses to express complex hierarchical structures. For example:

```xml
<!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 EAI includes four XML converter business services:

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

**NOTE:** XML converters may add unexpected carriage returns throughout the output document, for readability reasons. These characters are not significant and can be removed if the receiving application does not expect them and produces a parsing error. You can use eScript or Siebel VB to remove them.

Table 14 on page 49 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), 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 Business Applications. This type of integration uses the EAI XML Converter business service.

**NOTE:** These converters do not support Shift-JIS page code on UNIX platforms.

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.

**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.
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.

XML Document

```xml
<?Siebel-Property-Set EscapeNames= "true"?>
<Account>
  <Name>
    Account1
  </Name>
  <Contact City="Toronto">
    Davis, Pace
  </Contact>
</Account>
```

Integration Object

- Service Method Arguments
  - Siebel Message Header
    - <ListOfAccount>
  - Account
    - Name=Account
    - <ListOfContact>
      - Contact
        - City=Toronto
        - <Value>=Davis, Pace

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.
As shown in Figure 5, 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.

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 includes a Siebel Message or an external XML document. This representation is also handled in Siebel Workflow because it allows all the XML documents in memory to be treated as the Hierarchical Service Parameter of type XMLHierarchy.

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

- As illustrated in Figure 5 on page 47, there is a Property Set of type XMLHierarchy that always appears as a child of the Service Method Argument and the parent of the root XML element.
- Elements are represented by Property Sets. The XML tag is the type in the property set and the value assigned to that XML tag is the Value in the property set. For example, if an XML element has a value such as <Contact City="Toronto">Davis, Pace</Contact> as shown in Figure 5 on page 47, then the Value in the property set would be set to Davis, Pace and the Type in the property set would be set to contact.
- 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 47, 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.

**NOTE:** When using this business service, you should not specify an output argument name. The Siebel application automatically maps the newly generated property set to the specified output process property.
Siebel XML Converters Business Services Comparison

Table 14 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 14. Siebel XML Converters Comparisons

<table>
<thead>
<tr>
<th>User Requirement</th>
<th>EAI XML Converter</th>
<th>XML Hierarchy Converter</th>
<th>EAI Integration Object to XML Hierarchy Converter¹</th>
<th>XML Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Siebel Workflow</td>
<td>■</td>
<td>■</td>
<td>❐</td>
<td></td>
</tr>
<tr>
<td>Using Siebel Data Mapper</td>
<td>■</td>
<td>❐²</td>
<td>❐</td>
<td></td>
</tr>
</tbody>
</table>
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 15. The arguments for each method appear in the tables that follow.

### Table 15. EAI XML Converter Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object Hierarchy to XML Document</td>
<td>IntObjHierToXMLDoc</td>
<td>Converts an integration object hierarchy into an XML document.</td>
</tr>
<tr>
<td>XML Document to Integration Object Hierarchy</td>
<td>XMLDocToIntObjHier</td>
<td>Converts an XML document into an integration object hierarchy.</td>
</tr>
</tbody>
</table>
## Integration Object Hierarchy to XML Document Method Arguments

Table 16 describes the input arguments for the Integration Object Hierarchy to XML Document method of the EAI XML Converter.

Table 16. Integration Object Hierarchy to XML Document Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Message</td>
<td>SiebelMessage</td>
<td>Hierarchy</td>
<td>The Integration Object Hierarchy to be converted to XML.</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>The character encoding to use in the XML document. The default is UTF-16 for the Unicode version of Siebel applications.</td>
</tr>
<tr>
<td>Use Siebel Message Envelope</td>
<td>UseSiebelMessageEnvelope</td>
<td>String</td>
<td>Inserts the Siebel Message Envelope into the XML document. The default is True.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConv Errors</td>
<td>String</td>
<td>If some characters cannot be represented in the destination character set (like the local code page), the errors can be ignored. The errors are not ignored by default. For both situations, a warning error entry is created.</td>
</tr>
<tr>
<td>Tags on Separate Lines</td>
<td>Tags on Separate Lines</td>
<td>String</td>
<td>Default is False. When True, a line feed is placed at the end of each tag.</td>
</tr>
<tr>
<td>XML Header Text</td>
<td>XMLHeaderText</td>
<td>String</td>
<td>Text to prepend to the beginning of the XML document data.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateNamespaceDecl</td>
<td>String</td>
<td>Default is False. If True, the namespace declaration will be generated. You must manually create and name this input argument if it is required for your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateProcessingInstructions</td>
<td>String</td>
<td>Default is True. If set to False the Siebel processing instructions are not written. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>
Table 16. Integration Object Hierarchy to XML Document Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>GenerateSchemaTypes</td>
<td>String</td>
<td>Default is False. If set to True, then XSD schema types will be generated if set on the integration objects user properties. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>Namespace</td>
<td>String</td>
<td>If a namespace is defined here, it will override any namespace defined in the user properties of an integration object. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

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

Table 17. Integration Object Hierarchy to XML Document Method Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Document</td>
<td>&lt;Value&gt;</td>
<td>String</td>
<td>The resulting XML document.</td>
</tr>
</tbody>
</table>

**XML Document to Integration Object Hierarchy Method Arguments**

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

Table 18. XML Document to Integration Object Hierarchy Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object Name</td>
<td>IntObjectName</td>
<td>String</td>
<td>Name of the Integration Object to use in cases where the Siebel Message envelope might not be present.</td>
</tr>
</tbody>
</table>
### Table 18. XML Document to Integration Object Hierarchy Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object Lookup Rule Set</td>
<td>IntObjectNameLookupRuleSet</td>
<td>String</td>
<td>Rule Set for the EAI Dispatcher Service for finding out Integration Object Name in cases where the Siebel Message envelope might not be present.</td>
</tr>
<tr>
<td>Validate External Entity</td>
<td>ValidateExternalEntity</td>
<td>String</td>
<td>If True, the parser will be set to validate against external metadata, such as DTDs.</td>
</tr>
<tr>
<td>External Entity Directory</td>
<td>ExternalEntityDirectory</td>
<td>String</td>
<td>The directory to use for finding external entities referenced in the XML document, such as DTDs.</td>
</tr>
<tr>
<td>Truncate Field Values</td>
<td>TruncateFieldValues</td>
<td>String</td>
<td>Default is False. If True, truncate any fields longer than their maximum size, as specified in the Integration Component field definition.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
<tr>
<td>Contains Inline Attachments</td>
<td>ContainsInlineAttachments</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Tags on Separate Lines</td>
<td>Tags on Separate Lines</td>
<td>String</td>
<td>Default is False. When True, a line feed is placed at the end of each tag.</td>
</tr>
<tr>
<td>n/a</td>
<td>ProcessElementsOnly</td>
<td>String</td>
<td>Default is False. If set to True, processing of attributes is skipped. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateProcessingInstructions</td>
<td>String</td>
<td>Default is True. If set to False the Siebel processing instructions are not written. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>
Table 19 describes the output arguments for the XML Document to Integration Object Hierarchy method of the EAI XML Converter.

### Table 19. XML Document to Integration Object Hierarchy Method Output Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Message</td>
<td>SiebelMessage</td>
<td>Hierarchy</td>
<td>The Integration Object Hierarchy to be converted to XML.</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Character encoding of the XML document, detected by the converter independent of the parser.</td>
</tr>
</tbody>
</table>

**XML Hierarchy Converter Business Service**

There are two methods for the XML Hierarchy Converter, as shown in Table 20. The arguments for each method appear in the tables that follow.

### Table 20. XML Hierarchy Converter Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Document to XML Hierarchy</td>
<td>XMLDocToXMLHier</td>
<td>Converts an XML document into an XML Hierarchy.</td>
</tr>
<tr>
<td>XML Hierarchy to XML Document</td>
<td>XMLHierToXMLDoc</td>
<td>Converts an XML Hierarchy into an XML document.</td>
</tr>
</tbody>
</table>
## XML Document to XML Hierarchy Method Arguments

Table 21 describes the input arguments for the XML Document to XML Hierarchy method of the XML Hierarchy Converter.

### Table 21. XML Document to XML Hierarchy Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Document</td>
<td>&lt;Value&gt;</td>
<td>String</td>
<td>The input XML Document. If xml converter Business Services that expect XML Document (EAI XML Converter, XML Converter, XML Hierarchy Converter) are being used, then the <code>&lt;Value&gt;</code> should contain binary buffer rather than text string. When used in workflow, the Data Type for the process property used for XML Document should be “Binary”.</td>
</tr>
<tr>
<td>Escape Names</td>
<td>EscapeNames</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Validate External Entity</td>
<td>ValidateExternalEntity</td>
<td>String</td>
<td>If True, the parser will be set to validate against external metadata, such as DTD schemas.</td>
</tr>
<tr>
<td>External Entity Directory</td>
<td>ExternalEntityDirectory</td>
<td>String</td>
<td>Location of external entity files, such as DTD files.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
</tbody>
</table>
Table 22 describes the output arguments for the XML Document to XML Hierarchy method of the XML Hierarchy Converter.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Character encoding of the XML document, detected by the converter, independent of the parser.</td>
</tr>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The Output XML hierarchy.</td>
</tr>
</tbody>
</table>

XML Hierarchy to XML Document Method Arguments

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

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escape Names</td>
<td>EscapeNames</td>
<td>String</td>
<td>Invalid characters in XML tags will be escaped, using Siebel’s internal escape format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ If True, Escape invalid characters (this is the default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ If False, delete invalid characters. (Do not use in XML tags.)</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Outputs the XML character encoding to use. If encoding is blank or not supported, an error is produced.</td>
</tr>
</tbody>
</table>
### Table 23. XML Hierarchy to XML Document Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Header Text</td>
<td>XMLHeaderText</td>
<td>String</td>
<td>A string in a local code page character encoding to be inserted before the XML document’s root element, after the &lt;?xml...?&gt; 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 &lt;myheader data&lt;/myheader&gt; and the XML document output without this parameter is &lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;account&gt;...&lt;/account&gt;, then the document with the XMLHeaderText included will be: &lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;myheader&gt;some data&lt;/myheader&gt;&lt;account&gt;....&lt;/account&gt;</td>
</tr>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The XML hierarchy.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
<tr>
<td>Tags on Separate Lines</td>
<td>Tags on Separate Lines</td>
<td>String</td>
<td>When True, a line feed is placed at the end of each tag. The default is False.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateProcessingInstructions</td>
<td>String</td>
<td>Default is True. If set to False the Siebel processing instructions are not written. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>
Table 24 describes the output argument for the XML Hierarchy to XML Document method of the XML Hierarchy Converter.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
</table>

**EAI Integration Object to XML Hierarchy Converter Business Service**

There are two methods for the EAI Integration Object to XML Hierarchy Converter, as shown in Table 25. 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 25. EAI Integration Object to XML Hierarchy Converter Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object Hierarchy to XML Hierarchy</td>
<td>IntObjHierToXMLHier</td>
<td>Converts an integration object hierarchy to an XML hierarchy.</td>
</tr>
<tr>
<td>XML Hierarchy to Integration Object Hierarchy</td>
<td>XMLHierToIntObjHier</td>
<td>Converts an XML hierarchy to an integration object.</td>
</tr>
</tbody>
</table>
Integration Object Hierarchy to XML Hierarchy Method Arguments

Table 26 and Table 27 describe the arguments for the Integration Object Hierarchy to XML Hierarchy method of the EAI Integration Object to XML Hierarchy Converter.

Table 26. Integration Object Hierarchy to XML Hierarchy Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>Namespace</td>
<td>String</td>
<td>If a namespace is defined here, it will override any namespace defined in the user properties of an integration object.</td>
</tr>
<tr>
<td>Integration Object Hierarchy</td>
<td>SiebelMessage</td>
<td>Hierarchy</td>
<td>The integration object hierarchy to be converted.</td>
</tr>
<tr>
<td>Use Siebel Message Envelope</td>
<td>UseSiebelMessageEnvelope</td>
<td>String</td>
<td>Default is True. If set to True then the Siebel Message Envelope is used in the XML Hierarchy, otherwise the Siebel Message Envelope is not included.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateNamespaceDecl</td>
<td>String</td>
<td>Default is False. If set to True, the namespace declaration will be generated. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateSchemaTypes</td>
<td>String</td>
<td>Default is False. If set to True, then XSD schema types will be generated if set on the integration objects user properties. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

Table 27. Integration Object Hierarchy to XML Hierarchy Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The converted integration object.</td>
</tr>
</tbody>
</table>
### XML Hierarchy to Integration Object Hierarchy Method Arguments

Table 28 and Table 29 describe the arguments for the XML Hierarchy to Integration Object Hierarchy method of the EAI Integration Object to XML Hierarchy Converter.

#### Table 28. XML Hierarchy to Integration Object Hierarchy Input Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains Inline Attachments</td>
<td>ContainsInlineAttachments</td>
<td>String</td>
<td>Default is True. DTYPE_ATTACHMENT fields are assumed to include actual attachment content. If False, then the field is treated as a reference to an external attachment.</td>
</tr>
<tr>
<td>Integration Object Name</td>
<td>IntObjectName</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Strip Name Space</td>
<td>StripNamespace</td>
<td>String</td>
<td>Removes the namespace from XML tags.</td>
</tr>
<tr>
<td>Truncate Field Values</td>
<td>TruncateFieldValues</td>
<td>String</td>
<td>Default is True. If True, truncate any fields longer than their maximum size. If True, report fields that are too long as errors.</td>
</tr>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The hierarchy to be converted.</td>
</tr>
<tr>
<td>n/a</td>
<td>ProcessElementsOnly</td>
<td>String</td>
<td>Default is False. If set to True, processing of attributes is skipped. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

#### Table 29. XML Hierarchy to Integration Object Hierarchy Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Object Hierarchy</td>
<td>SiebelMessage</td>
<td>Hierarchy</td>
<td>The converted integration object.</td>
</tr>
</tbody>
</table>
XML Converter Business Service

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 30. The arguments for each method appear in the tables that follow.

Table 30. XML Converter Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Set to XML</td>
<td>PropSetToXML</td>
<td>Converts a property set hierarchy to XML. Returns the result in the Value field of the Output property set.</td>
</tr>
<tr>
<td>XML to Property Set</td>
<td>XMLToPropSet</td>
<td>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.</td>
</tr>
</tbody>
</table>

Property Set To XML Method Arguments

Table 31 and Table 32 describe the arguments for the Property Set To XML method of the XML Converter.

Table 31. Property Set To XML Method Input Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.</td>
<td>Hierarchical</td>
<td>The entire input property set. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

Table 32. Property Set To XML Method Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
</table>
XML To Property Set Method Arguments

Table 33 describes the input argument for the XML To Property Set method of the XML Converter.

Table 33. XML To Property Set Method Input Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
</table>

Table 34 describes the output argument for the XML To Property Set method of the XML Converter.

Table 34. XML To Property Set Method Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.</td>
<td>Hierarchical</td>
<td>The entire output property set. You must manually create and name this output argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

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 35 describes the EAI XML Write to File business service methods.

Table 35. EAI XML Write to File Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Siebel Message</td>
<td>WriteEAIMsg</td>
<td>Uses the EAI XML Converter.</td>
</tr>
<tr>
<td>Write XML Hierarchy</td>
<td>WriteXMLHier</td>
<td>Uses the XML Hierarchy Converter.</td>
</tr>
<tr>
<td>Write Property Set</td>
<td>WritePropSet</td>
<td>Uses the XML Converter.</td>
</tr>
</tbody>
</table>
Write Siebel Message Method Arguments

Table 36 describes the input arguments for the Write Siebel Message method of the EAI XML Write to File business service.

Table 36. Write Siebel Message Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the file where output is to be written. This is a required field.</td>
</tr>
<tr>
<td>Siebel Message</td>
<td>Siebel Message</td>
<td>Hierarchy</td>
<td>The Integration Object Hierarchy to be converted to XML.</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Character encoding in the XML document. If encoding is blank or not supported, an error is produced.</td>
</tr>
<tr>
<td>Use Siebel Message Envelope</td>
<td>UseSiebelMessageEnvelope</td>
<td>String</td>
<td>Default is True. Insert the Siebel Message Envelope into the XML document.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
<tr>
<td>Tags on Separate Lines</td>
<td>Tags on Separate Lines</td>
<td>String</td>
<td>Default is False. When True, a line feed is placed at the end of each tag.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateNamespaceDecl</td>
<td>String</td>
<td>Default is False. If set to True then the namespace declaration will be generated. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateProcessingInstructions</td>
<td>String</td>
<td>Default is True. If set to False then the Siebel processing instructions are not written. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>
Table 36. Write Siebel Message Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>GenerateSchemaTypes</td>
<td>String</td>
<td>Default is False. If set to True then XSD schema types will be generated if set on the integration objects user properties. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
<tr>
<td>n/a</td>
<td>Namespace</td>
<td>String</td>
<td>If a namespace is defined here, it will override any namespace defined in the user properties of an integration object. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

Write Property Set Method Arguments

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

Table 37. Write Property Set Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the file where output is to be written. This is a required field.</td>
</tr>
<tr>
<td>n/a</td>
<td>Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.</td>
<td>Hierarchical</td>
<td>The entire input property set. You must manually create and name this input argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>
Write XML Hierarchy Method Arguments

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

Table 38. Write XML Hierarchy Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the file where output is to be written. This is a required field.</td>
</tr>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The XML Hierarchy Property Set.</td>
</tr>
<tr>
<td>Escape Names</td>
<td>EscapeNames</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Outputs XML character encoding to use. If encoding is blank or not supported, an error is produced.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
<tr>
<td>Tags on Separate Lines</td>
<td>Tags on Separate Lines</td>
<td>String</td>
<td>Default is False. When True, a line feed is placed at the end of each tag.</td>
</tr>
<tr>
<td>n/a</td>
<td>GenerateProcessingInstructions</td>
<td>String</td>
<td>Default is True. If set to False the Siebel processing instructions are not written.</td>
</tr>
</tbody>
</table>

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

### Table 39. EAI XML Read from File Business Service Methods

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Siebel Message</td>
<td>ReadEAIMsg</td>
<td>Uses the EAI XML Converter</td>
</tr>
<tr>
<td>Read Property Set</td>
<td>ReadPropSet</td>
<td>Uses the XML Converter</td>
</tr>
<tr>
<td>Read XML Hierarchy</td>
<td>ReadXMLHier</td>
<td>Uses the XML Hierarchy Converter</td>
</tr>
</tbody>
</table>

### Read Siebel Message Method Arguments

Table 40 describes the input arguments for the Read Siebel Message method of the EAI XML Read from File business service.

### Table 40. Read Siebel Message Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the file to be read. This is a required field.</td>
</tr>
<tr>
<td>Integration Object Name</td>
<td>IntObjectName</td>
<td>String</td>
<td>Name of the Integration Object to use in cases where the Siebel Message header is not present.</td>
</tr>
<tr>
<td>Integration Object Lookup Rule Set</td>
<td>IntObjectLookupRuleSet</td>
<td>String</td>
<td>Rule Set for the EAI Dispatcher Service for finding the Integration Object Name in cases where the Siebel Message header is not present.</td>
</tr>
<tr>
<td>External Entity Directory</td>
<td>ExternalEntityDirectory</td>
<td>String</td>
<td>Directory to use for finding external entities referenced in the XML document, such as DTDs.</td>
</tr>
<tr>
<td>Truncate Field Values</td>
<td>TruncateFieldValues</td>
<td>String</td>
<td>Default is True. If True, truncate any fields longer than their maximum size. If False, report fields that are too long as errors.</td>
</tr>
</tbody>
</table>
Table 40. Read Siebel Message Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
<tr>
<td>n/a</td>
<td>ProcessElementsOnly</td>
<td>String</td>
<td>Default is False. If set to True, processing of attributes is skipped.</td>
</tr>
</tbody>
</table>

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

Table 41. Read Siebel Message Method Output Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Message</td>
<td>SiebelMessage</td>
<td>Hierarchy</td>
<td>The Integration Object Hierarchy converted from XML.</td>
</tr>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Outputs XML character encoding to use. If encoding is blank or not supported, an error is produced.</td>
</tr>
</tbody>
</table>

**Read Property Set Method Arguments**

Table 42 describes the input argument for the Read Property Set method of the EAI XML Read from File business service.

Table 42. Read Property Set Method Input Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the file to be read. This is a required field.</td>
</tr>
</tbody>
</table>
Table 43 describes the output argument for the Read Property Set method of the EAI XML Read from File business service.

### Table 43. Read Property Set Method Output Argument

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>Child type of the hierarchical process property containing the entire property set, service method arguments, and child property set.</td>
<td>Hierarchical</td>
<td>The entire output property set. You must manually create and name this output argument if it is required by your business needs.</td>
</tr>
</tbody>
</table>

### Read XML Hierarchy Method Arguments

Table 44 describes the input arguments for the Read XML Hierarchy method of the EAI XML Read from File business service.

### Table 44. Read XML Hierarchy Method Input Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>FileName</td>
<td>String</td>
<td>The name of the XML file to read. This is a Required field.</td>
</tr>
<tr>
<td>Escape Names</td>
<td>EscapeNames</td>
<td>String</td>
<td>Invalid characters in XML tags will be escaped, using Siebel’s internal escape format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ If True, process Escape characters (this is the default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ If False, do not process Escape characters.</td>
</tr>
<tr>
<td>External Entity Directory</td>
<td>ExternalEntityDirectory</td>
<td>String</td>
<td>Directory for external entities such as DTD files.</td>
</tr>
<tr>
<td>Ignore Character Set Conversion Errors</td>
<td>IgnoreCharSetConvErrors</td>
<td>String</td>
<td>Default is False. If the Siebel application cannot represent a given character set—such as the local code page character set—conversion errors are logged, including a warning log entry. When set to True, only a warning message is logged.</td>
</tr>
</tbody>
</table>
Table 45 describes the output arguments for the Read XML Hierarchy method of the EAI XML Read from File business service.

Table 45. Read XML Hierarchy Method Output Arguments

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Character Encoding</td>
<td>XMLCharEncoding</td>
<td>String</td>
<td>Character encoding of the XML document, detected by the converter independent of the parser.</td>
</tr>
<tr>
<td>XML Hierarchy</td>
<td>XMLHierarchy</td>
<td>Hierarchy</td>
<td>The XML Hierarchy property set.</td>
</tr>
</tbody>
</table>
XML Integration Scenarios

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.

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 XML schema: XSD, DTD, or XDR” on page 71.
- Create a new workflow. For details, see “To create a new workflow” on page 72.

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

1. Launch 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.
4. Complete the steps of the wizard:
   a. Select a business service from the Business Service drop-down list.
   b. Select the 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 details on Siebel DTD Wizard, see Transports and Interfaces: Siebel Enterprise Application Integration.

5. Load the schema into the external system.
**XML Integration Scenarios**  ■ Scenario 2: Integration Using External XML and an XSD or DTD

---

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

---

**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 EAI 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 73.

- 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 73.
Create a new workflow.
For details, see “To create a new workflow” on page 72.

To create the Siebel integration object
1 Start Siebel Tools and select File > New Object.
2 Select the EAI tab.
3 Double-click the Integration Object icon.
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 details on Integration Object Wizard, see Integration Platform Technologies: Siebel Enterprise Application Integration.
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 details on Siebel Data Mapper, see Business Processes and Rules: Siebel Enterprise Application Integration.

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.
XML Integration Scenarios

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

This appendix discusses using XML files as an input as well as inserting a file attachment into the Siebel Database using XML.

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](image)

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

```xml
<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>
        </Position>
      </ListOfPosition>
    </Employee>
  </ListOfSampleEmployees>
</SiebelMessage>
```
Using XML Files

Using an XML Document as Input

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 Business 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 such as 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.

```xml
<SiebelMessage MessageId="" IntObjectName="Sample Employee"/>
```
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.

For integration with external systems using File Attachments, refer to the chapter “Siebel EAI and File Attachments” in *Integration Platform Technologies: Siebel Enterprise Application Integration*.

For integration between Siebel instances, the support for attachments is built into the Siebel Adapter and the EAI XML Converter. The integration between Siebel instances can occur when generating or reading XML, which is further defined in the next section.

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

```xml
<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 negatively affect 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.
Generating XML. In the case of the Attachment business component being used, the Siebel Adapter will correctly perform the query. Then, the EAI XML Converter will include the attachment in XML.

Reading XML. If XML was generated by the EAI XML Converter as described previously, then the EAI XML Converter will read such XML and correctly bring attachments into memory. After which, the Siebel Adapter will insert them into Oracle’s Siebel database.
Symbols
#PCDATA, mapping of 41

A
attributes described and example 20
DTD wizard, used by to create XML integration object 41

B
base64, using to convert binary file to Siebel database 77
binary file, inserting into the Siebel database 77
business services. See XML converter business service details

C
Canonical section, integration object data type 20
character set, declaring in use 11
component container element, in Siebel integration object documents 24
component, described 21
connectors, about and table of 42

data flow, document-to-integration object flow (diagram) 10
Document Type Definition 26
Document Type Definitions. See DTDs

DTD Wizard
integration objects, about using to create 40, 43
integration objects, creating procedure 29, 30, 39, 40

DTDs
See also scenarios
metadata support, about 10
parentheses, about using for complex hierarchical structures 43

E
EAI XML Converter
about and XML document to integration object (diagram) 46
converter comparison, table of 49
Integration Object Hierarchy to XML Document, input and output arguments (table) 51
methods, described 50
XML Document to Integration Object
Hierarchy method, input and output arguments (table) 52

EAI XML Read from File business service
about 45
methods, described 65
Read Property Set method, arguments (table) 67, 68
Read Siebel Message method, input arguments (table) 66
Read Siebel Message method, output arguments (table) 67
Read XML Hierarchy method, output arguments (table) 69

EAI XML Write to File business service
about 45
methods, described 62
Write EAI Message method, input arguments (table) 63
Write Property Set method, input arguments (table) 64
either/or relationships, about 42
elements
See also attributes; Siebel integration object document
described and example 20
mapping to components, about 40
#PCDATA, mapping of 41
entities in XML at run-time, support of 40
escape characters, using in XML documents 11
External section, integration object data type 20

H
How XML Names Are Derived from Integration Objects 21

I
incoming XML scenario
external XML and DTD, setting up 72
instances, described 20
integration component container, in Siebel integration object documents 24
Integration Component Elements 23
integration components
element, in Siebel integration object documents 23
properties, table of 37
Integration Field Elements 24
integration object 19
Integration Object Hierarchy to XML Document method
arguments, input and output, table of 51
described 50
integration objects
See also XML converters
about and hierarchical architecture (diagram) 19
attributes, about 41
component or field name, about 41
connectors, about and table of 42
elements, about mapping to components 40
integration object data type, table of 20
#PCDATA, about mapping element to 41
properties, table of 36, 37, 38
Siebel systems, about used in 10
XML integration object, about and object diagram 19
XML Parent Field, about 42
integration objects, creating
DTD Wizard, about using to create 40, 43
DTD Wizard, creating procedure 29, 30, 39, 40
IntObjectName 22

J
JPEG images
Siebel database, inserting into 77

M
MessageId 22
metadata
described 9
support of 10

N
name, about component or integration object field 41
names of XML elements 21

O
Object List Element 22
one or the other relationships, about support by the DTD Wizard 42
outbound integration, scenario setting up 71, 72

P
Property Set to XML method
described 61
output arguments, table of 59, 60, 61
property sets
See also XML Converter; EAI XML Write to File
business service properly-formatted, example of 16

R
Read Property Set method
arguments, table of 67, 68
described 66
Read Siebel Message method
described 65
input arguments, table of 66
output arguments, table of 67
Read XML Hierarchy method
described 66
output arguments, table of 69
relationships, supported by DTD Wizard 42

S
scenarios
integration using external XML and a DTD 72
integration using Siebel XML 71, 72
Schema Generator Wizard, about 10
screen scraping, about 9
Siebel applications, XML support of 9
Siebel database, inserting binary file into 77
Siebel integration object document
component container element, about 24
integration component element, about and example 23
Siebel XML
scenario, integration using 71, 72
XML integration, about 45
SiebelMessage Element 22

W
Write Property Set method
described 62
input arguments, table of 64
Write Siebel Message method
described 62
input arguments, table of 63
Write XML Hierarchy method
described 62

X
XDR, about metadata support 10
XML
about 9
Siebel applications support of 9
XML Converter
about and XML document to property set representation (diagram) 48
converter comparisons, table of 49
methods, described 54, 61
Property Set to XML method, arguments (table) 59, 60, 61
XML converter business service details
EAI XML Read from File business service, using 65
EAI XML Write to File business service, using 62
XML converters
converter comparison, table of 49
EAI XML Converter, about and XML document to integration object (diagram) 46
EAI XML Converter, using 50
property sets, example of properly-formatted 16
XML Converter, about and XML document to property set representation (diagram) 48
XML Converter, using 61
XML document, using as input in a workflow, about and example 75, 77
XML Hierarchy Converter, about and representation of XML document structure (diagram) 46, 48
XML Hierarchy Converter, using 54
XML Data Reduced, about metadata support 10
XML document
See also XML Converter
attributes, described and example 20
element, described and example 20
input in a workflow, about using as 75
input in a workflow, sample XML document 75, 77
XML Document to Integration Object Hierarchy method
described 50
input arguments, table of 52
XML Document to XML Hierarchy method
described 54
input arguments, table of 55
output arguments, table of 56
XML Document, example of 24
XML documents
character set in use, declaring 11
data flow, document-to-integration object flow (diagram) 10
escape characters, using and table of 11
XML DTD 26
XML element
naming of 21
XML elements. See elements
XML Hierarchy Converter
about and representation of XML document structure (diagram) 46, 48
converter comparison, table of 49
XML Document to XML Hierarchy method, input arguments (table) 55
XML Document to XML Hierarchy method, output arguments (table) 56
XML Hierarchy to XML Document method, input arguments (table) 56
XML Hierarchy to XML Document method
described 54
input arguments (table) 56
XML integration objects
about and object diagram 19
integration object data type, table of 20
XML integration, types of 45
XML Parent Field, about 42
XML section, integration object data type 20
XML Tag attribute 21
XML to Property Set method
described 61