Oracle® Retail

Functional Artifacts Guide

Release 13.1

June 2009

ORACLE®

Oracle Retail Functional Artifacts Guide, Volume 1, Release 13.1 for Windows

Primary Author: David Burch

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server - Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.

(ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

5 XML Standards and Conventions

6 Customization and Extension

A References

Glossary

Preface

The *Oracle Retail Artifacts Guide* provides detailed information about how Oracle Retail artifacts are designed and packaged for release, as well as the standards used in their creation.

Audience

The *Oracle Retail Artifacts Guide* is intended for users of Oracle Retail applications, particularly application integrators, implementation staff, and retail IT personnel.

Related Documents

For more information, see the following documents in the Oracle Retail 13.1 documentation set:

- *Oracle Retail Integration Bus Data Model*
- *Oracle Retail Integration Bus Implementation Guide*
- *Oracle Retail Integration Bus Installation Guide*
- *Oracle Retail Integration Bus Integration Guide*
- *Oracle Retail Integration Bus Operations Guide*
- *Oracle Retail Integration Bus Release Notes*
- *Oracle Retail Integration Bus Hospital Administration Online Help*
- *Oracle Retail Integration Bus Hospital Administration User Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

- https://metalink.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to recreate
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

The purpose of this document is to provide detailed information about the following:

- How artifacts are designed and packaged for release, as well as the standards used for their creation.
- Version strategy and implementation of that strategy, as well as the standards used for base Business Objects.
- Customization strategy and implementation of that strategy, including extension of base Business Objects.

Overview

The Oracle Retail Business Object (BO) is the name given to the logical representation of an Oracle Retail Business Entity. The definition of that object is in the form of an XML Schema (XSD). The schema represents the common object definition for business concepts, such as account, supplier, pruchase order, and item.

Within Oracle Retail, the messages that flow between the retail applications are defined statically through these XML schemas (XSDs). These Business Objects are the message payloads, or the basis of message payloads for the Oracle Retail integration products, such as the Oracle Retail Integration Bus (RIB) and RIB Integration Gateway Services (IGS).

The Oracle Retail integration infrastructure products work with multiple technologies (for example, Java EE and PL/SQL). So they have different ways of representing the same functional XML message structure in the various technologies. The Business Object XSD is the common source used to generate the various manifestations of the BO (payloads) used within the infrastructure products and to ultimately communicate the Business Objects appropriately to the Oracle Retail API technology.

The XSDs are strict, and they are used by Oracle Retail standard tools to produce the design time physical objects used by the application's API technology (PL/SQL or Java), as well as the runtime validations used by the various integration components.

Note: See the *Oracle Retail Functional Artifact Generator Guide*.

Retail Functional Artifacts is the collective name given to the package containing the base Business Object XSDs and the various representations of the same structure/definition in these different technologies. These representations include the following:

- For Java EE, the artifacts are JAXB Java Beans. JAXB is a standard Java XML binding technology. It provides the mechnaism to convert XML instances to Java objects (and vice versa) in a standard way. The JavaEE Web service infrastructre internally uses JAXB to marshall and unmarshall the SOAP messages. For every Business Object XSD, there is a corresponding JAXB beans.
- For PL/SQL retail applications, the artifacts are Oracle Objects. These objects are user defined database objects that define the XML message structure inside the database. For every Business Object XSD, there is a corresponding Oracle Object SQL defined.

Note: See the *Oracle Retail Functional Artifact Generator Guide*.

2

Packaging Structure and Standards

This section describes the physical packaging structure of the Retail Functional Artifacts. There are two Retail Functional Artifacts packages:

- RibFuncArtifact13.1.1ForAll13.1.1Apps_eng_ga.tar
- RetailFuncArtifact13.1.1ForAll13.1.1Apps_eng_ga.tar

The RibFuncArtifact contains additional artifacts specific to the RIB application, such as TAFR related artifacts. The RetailFunc contains only product independent artifacts. All content is versioned and follows the standards as described in Chapter 4, "Artifact Versioning".

General Packaging Structure

The Retail Functional Artifacts are physically packaged as a directory structure. The following diagram illustrates the structure. Standard UNIX directory formatting is used within the rules associated with the physical packaging of the directory.

```
/retail/integration/base/
                        bo/  (Business Objects)
                             BO  (Object Name)
                                V[1]  (Version - latest)
                                  BO.xsd   (BO Schema)
                    /custom/
                          bo/  (Business Objects)
                               ExtOf<BO>  (Object Name)
                                  V[1]  (Version - latest)
                                    ExtOf<BO>.xsd   (Custom BO Schema)
```

3

Business Object Structure

This section describes the directories that contain the Oracle Retail Business Objects.

Base and Custom Directories

The base directories contain the Oracle Retail Business Objects, as delivered with each release and used by Oracle Retail applications within the integration products. The packaging structure reflects the namespace and the versioning within the namespace.

For example:

The ASNInBO.xsd namespace:

```
targetNamespace="http://www.oracle.com/retail/integration/base/bo/ASNInBO/v1"
version="1.0"
xmlns=http://www.oracle.com/retail/integration/base/bo/ASNInBO/v1
```

The ASNInBO.xsd within the packaging structure:

```
retail/integration/base/bo/ASNInBO/v1/ASNInBO.xsd
```

The Custom directories are provided for users to make extensions within the Artifact packages. These directories are contained with the "Custom" subdirectory. They contain a set of directories mimicking the base directory, but prefaced with ExtOf.

For example:

The ExtOfASNInBO.xsd namespace:

```
targetNamespace="http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInBO/v
1"
    version="1.0"
    xmlns=http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInBO/v1
```

The ExtOfASNInBO.xsd within the packaging structure:

```
retail/integration/custom/bo/ExtOfASNInBO/v1/ExtOfASNInBO.xsd
```

Within one artifact deliverable (packaged release) there will be only the latest version of each BO XSD file. For a breaking change the XSD is moved to a directory with the same name as the version number in the changed namespace.

4

Artifact Versioning

This chapter describes versioning strategies and standards.

Common Versioning Strategies

The following describes types of versioning strategies, as explained in Thomas Erl's book, *Web Service Contract Design and Versioning for SOA*:

Strict Versioning

For Strict Versioning, any compatible or incompatible change results in a new version of the service contract. This type of versioning does not support backward or forward compatibility (for example, New Change, New Contract).

Flexible Versioning

For Flexible Versioning, an incompatible change results in a new version of the service contract, and the contract supports backward compatibility but not forward compatibility.

Oracle Retail Versioning Strategy

The version approach proposed for Oracle Retail is the Flexible Strategy, where an incompatible change results in a new version of the service contract, and the contract supports backward compatibility but not forward compatibility.

Loose Versioning

For Loose Versioning, any incompatible change results in a new version of the service contract, and the contract supports backward and forward compatibility.

Comparison Chart

The following chart compares the types of versioning strategies listed above. The values in the Characteristics column are described as follows:

- Strictness: The rigidity of the contract versioning strategy.
- Governance Impact: The amount of governance burden imposed by the strategy.
- Complexity: The overall complexity of the versioning process.

Characteristic	Strict Strategy	Flexible Strategy	Loose Strategy
Strictness	High	Medium	Low
Governance Impact	High	Medium	High
Complexity	Low	Medium	High

Oracle Retail Versioning Strategy

The version approach proposed for Oracle Retail is the Flexible Stategy, where an incompatible change results in a new version of the service contract, and the contract supports backward compatibility but not forward compatiblity.

Reasons for Versioning an Artifact

For this proposal, there are two primary reasons to version an artifact:

- To recognize that an artifact as changed.
- To recognize whether the change is "breaking" or "non-breaking," as described below.

Breaking Changes

A breaking change also is known as non-backward compatibility. For example, newer versions of producers based on the same Business Object will break existing consumers and require a regeneration of the client from the new wsdl.

Non-backward compatible changes include the following:

- Removing an operation.
- Renaming an operation.
- Changing the parameters of an operation, such as data type or order.

Non-breaking Changes

A non-breaking change also is known as backward compatibility. For example, newer versions of producers based on the same Business Object can be deployed without breaking existing consumers.

Backward compatible changes include the following:

- New operations.
- New data types.
- New optional fields to existing data types.
- Type expansion.

Versioning Retail Business Objects (BO) Schemas (XSDs)

Versioning Approach for XSDs

The XSD is versioned by an attribute (version) and the namespace. The namespace is the key versioning point.

Each XSD has a version attribute.

Topic	Description
Recommendation	Each XSD has a version attribute.
Rationale	The version attribute is used to indicated a change has occurred to the XSD. It is a simple counter that starts at 1.00 and "bumps" when any change is applied.
Example	targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1"

XSD Version Attribute Usage

Topic	Description
Recommendation	For any change, breaking or non-breaking, the version attribute is incremented.
Rationale	For a non-breaking change, the version will increment the point. For a breaking change, the version will bump. The breaking version bump equates to the same version reflected in the namespace.
Example - Non-breaking Change	Original targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" Non-breaking Change targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" version="1.1" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" Breaking Change targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v2" version="2.0" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v2"

XSD Namespace Versioning

Topic	Description
Recommendation	XSD namespace is versioned.
Rationale	For breaking changes, the BO XSD namepsace is versioned.
Example	The following example shows how the versioning appears, before and after. Before targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v1" After targetNamespace="http://www.oracle.com/retail/integration/base/bo/PayTermDesc/v2" version="2.0" xmlns="http://www.oracle.com/retail/integration/base/bo/PayTermBO/v2"

Parent/Child Versioning

Topic	Description
Recommendation	For Parent/Child versioning, if a child version changes (a breaking change) the parent version also must change.
Rationale	If a parent imports an XSD and it has a breaking change, the parent also will have a breaking change. So the versioning of the parent should reflect that.
Example	N/A

5

XML Standards and Conventions

This section defines standards in the areas of naming conventions, XML standards, and customization/extension.

The standards in this document are for the artifacts used in Oracle Retail interration products, such as the RIB, Integration Gateway Services and Web Service Providers created by the Retail SOA Enabler tool. For Oracle AIA this section applies to edge-application Web services and integration patterns. These standards do not overlap the AIA standards for middle layer integrations. For those integrations, the AIA standards are directly applicable.

The standards and conventions described in this section are retail domain specific. The naming conventions and standards are only patterned after AIA concepts.

XML Standards and Conventions

The underscore (_) must be used for readability in naming simple elements.

This standard is a significant deviation from generally accepted WS standards, including AIA. The underscore is used by the Business Objects in the RIB and by Oracle Retail PL/SQL applications.

Topic	Description
Recommendation	The underscore must be used for readability in naming simple elements, with no spaces or hyphens between words; lowerCamel case cannot be used.
Rationale	This is not the generally accepted recommendation in other XML standards. Because Business Objects are translated into Oracle Objects for use within the database by PL/SQL APIs, all elements are made uppercase by default. Readability is enhanced by using the underscore instead of the industry standard lowerCamel case.
Example	<xsd:element name="supplier_name" type="varchar2240"/>

Names must not contain special characters.

Topic	Description
Recommendation	Names must not contain special characters.
Rationale	This is recommended to be similar to other XML standards. Note that the underscore is allowed. This is not the same as other standards.
Examples	space, '-', '.', '$', '%', '#',<, >, &, ?

UpperCamel case must be used for naming complex types.

Topic	Description
Recommendation	UpperCamel case must be used for naming complex types with no spaces or hyphens between words.
Rationale	This is recommended to be similar to other XML standards..
Example	<xsd:complexType name="SupplierType"/>

Avoid XML schema keywords as element names.

Topic	Description
Recommendation	Avoid XML schema keywords as element names. Do not use XSD keywords, such as element, sequence, and complexType as element names.
Rationale	This is recommended to be similar to other XML standards.
Example	<xs:element name="element" type="xs:integer"/>

Avoid Java keywords.

Topic	Description
Recommendation	Avoid Java keywords as element names.
Rationale	This is recommended to be similar to other XML standards.
Examples	abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, false, final, finally, float, for, goto, if, implements, import, instanceof, int,interface, long, native, new, null, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, true, try, void, volatile, while.

Avoid having numeric characters in the name.

Topic	Description
Recommendation	Avoid having numeric characters in the name.
Rationale	This is recommended to be similar to other XML standards.
Example	<xsd:element name="supplier1Name" type="varchar2240"/>

All XSD elements should be annotated.

Topic	Description
Recommendation	The definitions of all elements/attributes/enumerations should be included in the schema as notations. Each XSD file should declare as a namespace: xmlns:retailDoc="http://www.w3.org/2001/XMLSchema
Rationale	This is recommended to be similar to other XML standards. The XSD should be the source of truth and include the business functional documentation of the element. The annotation standard should be independent of any repository specific requirements. The retailDoc namespace will allow intelligent processing in the XML/XSD parser in the future. There can be different behavior for a "documentation" element when the prefix is retailDoc versus anything else (including xs: or xsd:), even though all of them belong to same namespace.
Example	xmlns:retailDoc="http://www.w3.org/2001/XMLSchema" <xs:element name="unit_qty" type="number12_4"> <retailDoc:annotation> <retailDoc:documentation>Contains the number of items expected to be received based on the supplier's ASN for this Item/Shipment combination.</retailDoc:documentation> </retailDoc:annotation> </xs:element>

All XSDs should use the date type, xsd:dateTime, and the XML format, 2002-05-30T09:00:00

Topic	Description
Recommendation	All XSDs should use the date type, xsd:dateTime, and the XML format, 2002-05-30T09:00:00
Rationale	This is recommended to be similar to other XML standards.
Example	<xs:element name="ship_date" type="xs:dateTime">

Retail Business Objects Schema

The logical representation of an Oracle Retail business entity is known as the Oracle Retail Business Object. For the purposes of this document, it describes the definition of that object in the form of an XML Schema (XSD). The schema represents the common object definition for business concepts such as Account, Supplier, Purchase Order, and Item.

Business Object (XSD) names should not contain names.

Topic	Description
Recommendation	The Business Object name should not contain message related operations or actions, such as Request or Response.
Rationale	The name should be only a "business" noun and should not necessarily reference the usage of the object.
Example	ItemRequest.xsd or ItemResponse.xsd.

Business Object (XSD) names should not contain CRUD operations.

Topic	Description
Recommendation	The Business Object name should not contain message related operations or actions, such as Create, Update, Process, and Get.
Rationale	The name should be only a "business" noun and should not necessarily reference the usage of the object CRUD operations.
Example	ItemCreate.xsd or ItemMod.xsd.

Each Business Object (XSD) file must define only one top level/global element.

Topic	Description
Recommendation	
Rationale	.
Example	

The Business Object (XSD) file must match the name of hte top level element.

Topic	Description
Recommendation	Every XSD file must define only one top level/global element, and the file name must match the name of the top level element.
Rationale	This is recommended to be similar to other XML standards.
Example	SupplierBO.xsd <xs:element name="SupplierBO"> <retailDoc:annotation> <retailDoc:documentation></retailDoc:documentation> </retailDoc:annotation> <xs:complexType>

Each Business Object (XSD) must have its own namespace.

Topic	Description
Recommendation	Each Business Object must have its own namespace.
Rationale	This is to eliminate conflicts when a service consumer will consume multiple from the same provider and the generate client side code should not have conflict. Each BO having its own namespace also allows versioning of each BO. The file will be located in a directory structure that matches the "path" portion of the namespace URL's for the top level type that is defined in that file.
Example	targetNamespace="http://www.oracle.com/retail/integration/base/bo/SupplierBO/v1" version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/SupplierBO/v1"

Each Business Object (XSD) namespace should not contain application identifiers.

Topic	Description
Recommendation	Each Business Object namespace should not contain application identifiers.
Rationale	Business Objects are not tied to any specific retail applications. These are generic business objects that belong to the full retail domain. So the Business Object namespace should not contain application identifiers.
Example	targetNamespace="http://www.oracle.com/retail/integration/base/bo/SupplierBO/v1" version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/SupplierBO/v1"

Each Business Object (XSD) root should be annotated.

Topic	Description
Recommendation	The XSD should be the source of truth and include the business functional documentation of the element.
Rationale	This is recommended to be similar to other XML standards. The annotation standard is independent of any repository specific requirements for the Business Objects.
Example	<xs:element name="SupplierBO"> < retailDoc:annotation> < retailDoc:documentation> Merchandise Suppliers are suppliers of goods and services to be sold by the retailer to customers </ retailDoc:documentation> </ retailDoc:annotation>

6

Customization and Extension

Customization and extension are customer side activities. The Oracle Retail release of Functional Artifacts coincides with a Oracle Retail Enterprise Release. Any new Business Objects or changes to existing base Business Objects are handled as described in Chapter 4, "Artifact Versioning". The packaging schema is discussed in Chapter 2, "Packaging Structure and Standards".

Business Object customization is defined as changes or extension of the base XSDs made by a customer to satisfy a customer specific business requirement.

Oracle Application Integration Architecture (AIA) and other industry standards have defined an approach using well known tags and packaging locations to separate the custom extensions from the base. This allows the extensions to be preserved as updates to the base are applied. Oracle Retail has implemented the same model.

Business Object Customization Tools and Approaches

The *Oracle Retail Functional Artifacts Guide* describes the tool used to generate the Functional Artifacts from the Business Object XSDs. The guide also contains examples and step-by-step instructions through simple use cases.

The *Oracle Retail Integration Bus Implementation Guide* contains a chapter about RIB customization and extension, with examples that include use cases including customization to the Business Objects.

The Retail SOA Enabler Tool uses Functional Artifacts as one of its tools inputs in the process of creating Web services.

Business Object Customization Standards

This section defines the standards used in the creation and management of the Functional Artifacts as they relate to the customization and extension of the base Business Objects (XSDs). These standards should be observed in any customer side effort to permit the preservation of changes when there are new releases of base objects.

Each Business Object schema module must import its custom Business Object schema module:

Topic	Description
Recommendation	Each BO schema must import its custom BO schema module.
Rationale	The custom schema is well defined and in the packaging structure.
Example	targetNamespace=http://www.oracle.com/retail/integration/base/bo/ASNInBO/v1 version="1.0" xmlns="http://www.oracle.com/retail/integration/base/bo/ASNInBO/v1" xmlns:ExtOfASNInBO=http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInBO/v1 xmlns:retailDoc=http://www.w3.org/2001/XMLSchema xmlns:xs="http://www.w3.org/2001/XMLSchema"> <xs:import namespace="http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInBO/v1" schemaLocation="../../../../custom/bo/ExtOfASNInBO/v1/ExtOfASNInBO.xsd"> </xs:import>

Each BO XSD complexType must have a local element named "Custom<Parent Type Name>" added.

Topic	Description
Recommendation	Each BO XSD complexType must have a local element named "Custom<Parent Type Name>" added.
Rationale	A local element named "Custom<Parent Type Name>" exists at the end of every EBO type. For example, CustomASNInBO.
Example	<xs:element maxOccurs="1" minOccurs="0" ref="ExtOfASNInBO:ExtOfASNInItem"> </xs:element>

The "Custom<Parent Type Name>" must not be recursively defined.

Topic	Description
Recommendation	The "Custom<Parent Type Name>" must not recursively define a note named "Custom<Parent Type Name>" within it.
Rationale	A Custom<Parent Type Name> must be one level deep, or in relatively flattened structure (to reduce complexity).
Example	

The custom BO schema module must be named ExtOf<BO>.

Topic	Description
Recommendation	The naming convention for the custom XSD for an BO must be as follows: ExtOf<BO>.
Rationale	The naming convention standardizes the names for namespace, packaging, and tool generation of Objects.
	ASNInBO.xsd has a corresponding ExtOfASNInBO.xsd:
Example	`xmlns="http://www.oracle.com/retail/integration/base/bo/ASNInBO/v1"` `xmlns:ExtOfASNInBO="http://www.oracle.com/retail/integration/custom/bo/ExtOfASNInBO/v1"`

A

References

References

The following is a list of reference materials providing more information on the topics covered in this guide.

- [XML] Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml
- [XMLSchema1] W3C Recommendation, XML Schema Part 1: Structures, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
- [XMLSchema2] W3C Recommendation, XML Schema Part 2: Datatypes,

  http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
- [OAGINDR] OAGi OAGIS 9.0 Naming and Design Rules Standard version 0.7, Published September 30, 2005
- [CCTS] UN/CEFACT Core Components Technical Specification Draft 2.2, Published March 31,2006.

  www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf
- [UNCEFACTNDR] UN/CEFACT XML Naming and Design Rules, Draft 2.0, Published January20, 2006

  http://www.unece.org/cefact/xml/xml_index.htm
- Oracle Application Integration Architecture - Enterprise Object Library 2.2: Enterprise Business Objects and Messages XML Naming and Design Rules Part No. E12769-01 August 2008.
- ARTS SOA Best Practices Technical Report Version 1.2.0 November 19, 2008 - Candidate Recommendation.
- "Web Service Contract Design and Versioning for SOA",Thomas Erl, et. al., September 2008, Prentice Hall.
- WS-I Basic Profile 1.1. http://www.ws-i.org/Profiles/BasicProfile-1.1.html

Glossary

The following is a list of acronyms used within this guide.

ABCS

Application Business Connector Service

AIA

Application Integration Architecture

ABM

Application Business Message

ABO

Application Business Object

BO

Retail Business Object

EBM

Enterprise Business Message

EBO

Enterprise Business Object

ESB

Enterprise Service Bus

ODI

Oracle Data Integrator

POJO

Plain Old Java Object

RGBU

Retail Global Business Unit

RTG

Retail Technology Group

SOA

Service Oriented Architecture: This term refers to an architectural style of building reliable distributed systems that deliver functionality as services, with additional emphasis on loose coupling between interacting services.