IXRetail Best Practices

Version 2.0

2006/07/14 Candidate Recommendation

Chairman:

Richard Mader	ARTS

Author:

Richard Halter	MIC

Technical Committee:

Dave Moorman	PCMS Datafit
John Fluke	IBM
Frank May	Microsoft
Jim Galloway	Afterbot
Paul Gay	Epson
Jay Heavilon	MARS
Tim Hood	SAP
Henry Tieding	Radioshack
John Hervey	PCATS
Monty Moncrief	Blockbuster
Jerry Rightmer	Oracle
Ron Kleinman	Sun Microsystems
Dean Sleeper	AccessVia
Dave VanHorn	SofTechnics
Warren Backer	Target
Paul Faha	CRS Retail

TABLE OF CONTENTS

1 INTRODUCTION....5
2 USE W3C SPECIFICATION FOR XML SCHEMAS....5
3 TAXONOMY....5
3.1 Use Upper Camel Case....5
3.2 Readability Is More Important Than Tag Length....5
3.3 Abbreviations Shall Not be Used in Element and Attribute Names....6
3.4 Element/Attribute/Enumerations Definitions....6
3.5 Remove Database Entity Names from Attribute Names....7
3.6 Names Should Not Include a Repetition of the Names of Containing Structures.7
3.7 Reuse Names from the ARTS XML Dictionary:....8
3.8 Naming of Complex Types....8
3.9 Use Compound Names When Defined Globally....9
3.10 Element vs Attribute....10
4 FACETS....10
4.1 Enumeration Values....10
4.2 Named Enumeration Guidelines....10
4.3 Named Enumeration Composition....11
4.4 Enumerations in English.....11
4.5 Default Enumerated Attributes....11
4.6 Default Boolean Flags....12
4.7 Date/Time....12
5 COMPONENT REFERENCING....13
5.1 One Namespace....13
5.2 IXRetail is the Default Namespace....13
5.3 IXRetail is the Default and Target Namespaces....14
5.4 Unique URI Value for “schemaLocation”....14
5.5 Relative Addressing for “schemaLocation”....14
5.6 Use Consistent Prefixes for Non-IXRetail Namespaces....15
6 VERSIONING....15
7 COMMON DATA....16
7.1 Separate Common Data Vocabulary Elements....16
7.2 Common Data Versioning....16
7.3 Common Data Repository....16
8 SCHEMA ARCHITECTURE....16
8.1 Clear Top Level Element....16
8.2 Venetian Blind Design Model....17
8.3 Global “simpleType” and “complexType”....17
8.4 No Anonymous Types....18
8.5 Nested Elements Shall Use the “type” Attribute....18
8.6 Base Schema....19
8.7 Publish Logical Groupings of Message Sets....19
9 EXTENDING SCHEMAS....19
10 PUBLISH “FLATTENED” SCHEMAS....19
11 DEPRECATION OF ELEMENTS AND ATTRIBUTES....20

12 DOCUMENT HISTORY 20
13 COMMITTEE MEMBERSHIP 21
14 DOMAIN GLOSSARY 21

1 Introduction

Criteria for "good" XML specifications and XML schema specifications may vary widely depending on the uses to be made of those specifications. Even when it has been agreed to use the W3C specification for XML Schema, there may be no single set of criteria for "good" application of the standards to specifications and determination of the "best" criteria can be even more uncertain. However, when the particular environment of application can be restricted, one can hope to determine, by well-informed consensus, the set of criteria that will lead to the best practice of using XML and related standards in that environment. For this particular document, the particular environment is data interchange between and among information technology applications that support the operation of retail stores or that integrate retail stores with the retail enterprise.

2 Use W3C Specification for XML Schemas

Recommendation	Use W3C specification for XML Schemas instead of DTDs or alternative schema languages.
Rationale	The XML Schema language permits names with local scope whereas DTDs require that all element names be named globally within the namespace.

3 Taxonomy

3.1 Use Upper Camel Case

Recommendation	Use Upper "camel" case with no spaces or hyphens between words.
Rationale	This is recommended to be similar to other XML standards and for easier readability than single-case text. Camel case results in the capitalization of the first letter of each word of a compound element or attribute name.
Example	<InventoryControlDocument> or <TransactionSequenceNumber>

3.2 Readability Is More Important Than Tag Length

Recommendation	Readability is more important than tag length.
Rationale	Although the concern for tag length remains, it is important that users be helped to choose the correct tag.
Example	`<POSDepartmentID>` is preferred to `<ID_DPT_POS>`.

3.3 Abbreviations Shall Not be Used in Element and Attribute Names

Recommendation	With a few exceptions, abbreviations should not be used in element and attribute names.
Rationale	The logical view of the ARTS Data Model has avoided abbreviations and this was considered desirable also for element and attribute names. Exceptions include acronyms (such as UCC, EAN, and UPC) and abbreviations (such as ID). A complete list of approved exceptions are identified/maintained in the data dictionary.
Example	UCC, EAN, UPC, SKU, ID, UOM, GTIN, PLU, ISBN, ISSN, EPC, RFID, MUZEID, etc.

3.4 Element/Attribute/Enumerations Definitions

Recommendation	Definitions of all elements/attributes/enumerations shall be included in the schema as annotations.
Rationale	For ease of understanding the schema and to avoid misunderstandings.
Example	<xs:element name=”Name” minOccurs=”0”> <xs:annotation> <xs:documentation>The individual parts of the name</xs:documentation> </xs:annotation> </xs:element>

3.5 Remove Database Entity Names from Attribute Names

Recommendation	Remove database entity names from attribute names where possible.
Rationale	In the ARTS Data Model, the entity name is often used as a prefix for attribute names. This makes importing foreign keys easier in models based on SQL. However, the hierarchical structure of an XML message can eliminate ambiguity and make repeating the entity name unnecessary. Thus, repetition of entity names needlessly increases tag lengths and is undesirable. {This recommendation uses the "entities and attributes" terminology of the ARTS Data Model. But an attribute from the Data Model may be either an element or an attribute in XML.}
Example	<Contact> <Name>John Doe</Name> </Contact>
Discouraged Example	<Contact> <ContactName>John Doe</ContactName> <Contact>

3.6 Names Should Not Include a Repetition of the Names of Containing Structures.

Recommendation	ARTS XML Dictionary has another column to reference the corresponding ARTS Data Model identifier and to eliminate from XML Dictionary names artifacts of the structure (specifically, eliminate repetition of the names of containing structures). {Recommendation "Remove database entity names from attribute names" is related.} The container provides adequate context and using its name in component names is redundant and needlessly lengthens component tags.
Example	A <Customer> element could contain a <Name> element but should not contain a <CustomerName> element.

3.7 Reuse Names from the ARTS XML Dictionary:

Recommendation	Use names from ARTS XML Dictionary when these are appropriate, instead of inventing new names.
Rationale	The ARTS XML Dictionary is a list of names derived from entity and attribute names of the logical view of the ARTS Data Model. The context of the ARTS Data Model provides significant semantics to these names. The names must still be selected and used consistently with all the Best Practices recommendations that IXRetail has adopted.

3.8 Naming of Complex Types

Recommendation	The element names must be distinct from the complex type names by appending “Type” for embedded complex types and “CommonData” for common data complex types.
Rationale	Use of the same name causes confusion/problems with marshalling tools.
Example	<xs:element name=”Response” type=”ResponseType”/> <xs:element name=”Name” type=”NameCommonData”/>
Discouraged Example	<xs:element name=”Response” type=”Response”/>

3.9 Use Compound Names When Defined Globally

Recommendation	When choosing a name that is global within a namespace, use compound names that describe the specific meaning of the thing being declared or defined.
Rationale	If global names are simple, users will tend to think of them as having a general utility, even when the type was chosen to meet the requirement of only a limited domain, industry segment, or geographic region. A name that is "global within a namespace" is contrasted with names that are "locally-declared" on nested elements according to Recommendation "Nested elements shall use the "type" attribute" When Recommendations "The default and target namespaces shall be IXRetail Namespace' and "Reused simpleType and complexType shall be declared globally" are followed, the set of global types in the IXRetail namespace form a reusable type dictionary. This recommendation helps to avoid inappropriately general use of a name having a specific meaning.
Example	A 'name' global type should not be defined with 'FirstName', 'MiddleIntial', and 'FamilyName' components because these, while common in USA, do not meet all international requirements, or even all USA geographies. A more generic global type with components of 'given' and 'family' could appropriately be named 'GivenAndFamilyNames'.

3.10 Element vs Attribute

Recommendation	Attributes instead of elements should be used in situations, such as: 1. Enumerated datasets such as Function or Activity codes. 2. Boolean Flags. 3. Ordinal sequences of repeating elements, i.e. <MerchandiseHierarchy Level=1> or <Name RelativeOrder=1> When in doubt, use elements.
Rationale	Attributes can and usually do contain important information that is not part of the content of the element, i.e. metadata. This means that, while the results of the attribute are usually visible, often the attribute itself is more important to the XML processor than it is to the person viewing the document. Or in the case of enumerations, attributes allow the definition of default values. If one is using a validating parser, the parser will add missing attributes with the default value. If not using a validating parser, the programmer will know the expected default value and can program accordingly.

4 Facets

4.1 Enumeration Values

Recommendation	Enumeration values should use names only (not numbers) or codes from ARTS approved standards bodies.
Rationale	Numbers only mean something when both sides have the same code set. Names help avoid this coordination.
Example	<xs:enumeration value=”Stock”/> <xs:enumeration value=”Service”/> <xs:enumeration value=”Alteration”/>

4.2 Named Enumeration Guidelines

Recommendation	When names are used for enumeration values they must conform to the guidelines for element or attribute names. If suitable names already exist, they should be used (instead of IXRetail creating new names).
Rationale	Provides for consistency across ARTS schemas
Example	<xs:enumeration value=”PointsEarned”/>

4.3 Named Enumeration Composition

Recommendation	Names composed of natural language words shall try to suggest the meaning of the value.
Rationale	Eases understanding
Example	<xs:enumeration value=”Start”/> <xs:enumeration value=”End”/>
Discouraged Example	<xs:enumeration value=”10”/> <xs:enumeration value=”20”/>

4.4 Enumerations in English.

Recommendation	Enumeration use values should use names consisting of English words.
Rationale	The discussion concluded that English enumeration names were no more a burden on the programmer than English element or attribute names. Names based on a natural language can suggest meanings by appropriate selection of words; numeric values lack this ability. It is helpful to be consistent with names derived from the ARTS Data Model, which use English words. The words displayed to end-users for elements, attributes, and enumerated values need to be chosen for usability by end-users and will probably need to be translated even for English-speakers. Only programmers who do system debugging should be expected to directly deal with XML messages.
Example	<xs:enumeration value=”Home”/> <xs:enumeration value=”Work”/> <xs:enumeration value=”Delivery”/> <xs:enumeration value=”Pickup”/>

4.5 Default Enumerated Attributes

Recommendation	Where possible, enumerated attributes should have Defaults.
Rationale	A default value for all enumerated data types shall be declared to allow for: 1. Reduces the size of the instance document. 2. Validating parsers to insert the default value when the attribute is not present. 3. Non-validating parser users to know in advance what it means for a missing attribute.
Example	<xs:attribute name=”AssetStatus” type=”AssetStatusTypeCode” use=”optional” default=”Approved”/>

4.6 Default Boolean Flags

Recommendation	1. Make flags optional. 2. Only, include flags when the value was TRUE (unless ambiguity results). 3. Make flags attributes.
Rationale	Normal software naming conventions say the flag names should indicate what happens when the attribute is true. So if one wants something to happen, then they include the true condition, i.e. if this is true then do this.
Example	@VoidFlag – indicates this is a void condition. - The normal system operation is not void, i.e. @VoidFlag = false. So under normal operations this flag is not included in the instance document. - Under a void condition, i.e. @VoidFlag = true, this flag becomes necessary to communicate the situation. Therefore this attribute would be included in the instance document.

4.7 Date/Time

Recommendation	Date and time formats follow ISO-8601 Date and Time Standard.
Rationale	Standardize on format.
Example	Year, Month, Day – 1993-02-14 Time – 10:10:20 Combined Date/Time – 1993-02-14T13:10:30 Complete - 2001-12-17T09:30:47.0Z

5 Component Referencing

5.1 One Namespace

Recommendation	All schemas specified by IXRetail shall use one namespace. http://www.nrf-arts.org/IXRetail/namespace
Rationale	This permits both XML schemas and XML instance documents to use that namespace as the default namespace so that names from this namespace do not have to be explicitly prefixed. Avoiding explicit prefixes in this way helps to keep tags short without ambiguity. Putting IXRetail names in a namespace avoids name collisions with schema specifications from other sources that our users may also need. IXRetail can check that each name that is global within this namespace has a unique declaration. By avoiding multiple namespaces, IXRetail can better limit occurrences of unintended equivalent declarations.

5.2 IXRetail is the Default Namespace

Recommendation	Each XML instance document produced by IXRetail should specify a default namespace and that should be the IXRetail namespace.
Rationale	The use of a default namespace avoids the need to explicitly prefix names from that namespace. This shortens the tags that use names from the IXRetail namespace. It also provides the appropriate example to users of XML schemas specified by IXRetail. (XML schemas should be accompanied with example XML instance documents to show how to use the schemas.)
Example	xmlns=”http://www.nrf-arts.org/IXRetail/namespace/”

5.3 IXRetail is the Default and Target Namespaces

Recommendation	Each XML schema document produced by IXRetail should specify a default namespace and a targetNamespace and both should be the IXRetail namespace.
Rationale	This is done for consistent references to names from the IXRetail namespace. Although this requires that names from XML Schema be explicitly prefixed, this only increases the length of the schema, not the length of instance documents. It does make the handling of all names defined in XML Schema and related XML standards consistent with each other. This also avoids problems with self-reference in <include>d schemas that have no target namespace. (Although, it is not clear why including such schemas would be desirable in an IXRetail schema.)
Example	targetNamespace=”http://www.nrf-arts.org/IXRetail/namespace/”

5.4 Unique URI Value for “schemaLocation”

Recommendation	Each version of a schema produced by IXRetail must have its own URI value for the schemaLocation attribute that is different than the URI value of every other version of every other schema; the URI must be in a hierarchy agreed with ARTS-NRF.
Rationale	Using a version mechanism that parallels the schema-discovery mechanism of validating XML compilers is desirable.
Example	xsi:schemaLocation=”http://www.nrf-arts.org/IXRetail/namespace/ ../PCM-V1.0.0.xsd”

5.5 Relative Addressing for “schemaLocation”

Recommendation	Use relative addressing for schemaLocation
Rationale	Eliminates problems with different operating systems
Example	xsi:schemaLocation=”http://www.nrf-arts.org/IXRetail/namespace/ ../PCM-V1.0.0.xsd”

5.6 Use Consistent Prefixes for Non-IXRetail Namespaces

Recommendation	Use consistent prefixes for names from namespaces that differ from the IXRetail namespace. Do not use a prefix for the IXRetail namespace.
Rationale	Keeping default namespaces and prefixes consistent helps make <include>d schemas have the same meaning as inline textual inclusion which helps people come to the same conclusions as to meaning that compilers do.
Example	xmlns:xs=”http://www.w3.org/2001/XMLSchema” xmlns=”http://www.nrf-arts.org/IXRetail/namespace/”
Discouraged Example	In Schema A use: xmlns:**xsd**=”http://www.w3.org/2001/XMLSchema” In Schema B use: xmlns:**xs**=”http://www.w3.org/2001/XMLSchema”

6 Versioning

Recommendation	1. An attribute called “MajorVersion” which contains the major version number. Because this indicates a break in backward compatibility, it and its contents are mandatory. 2. An attribute called “MinorVersion” which contains the minor version number. It indicates additional elements but doesn’t break backward compatibility. The attribute would be mandatory but the content would require the vendor to validate. 3. An attribute called “FixVersion” which contains the fix number. It indicates a bug was fixed in the schema and simply makes it work. The attribute would be mandatory but the content would require the vendor to validate.
Rationale	Matches the ARTS Process.
Example	<xs:element name=”PCM”> <xs:complexType> <xs:attribute name=”MajorVersion” use=”required” fixed=”02”/> <xs:attribute name=”MinorVersion” fixed=”00”/> <xs:attribute name=”FixVersion” fixed=”0”/> </xs:complexType>

7 Common Data

7.1 Separate Common Data Vocabulary Elements

Recommendation	Vocabulary Elements are the fundamental building blocks, name, address, etc, when aggregated create a domain grammar. Each Vocabulary Element will be contained within its own schema file (XSD).
Rationale	1. Eases maintenance and versioning. 2. Eliminates extraneous Vocabulary Elements from the published schema which reduces the size of the published schema.
Example	NameCommonData-V1.0.0.xsd

7.2 Common Data Versioning

Recommendation	Each common data schema file will be versioned following the ARTS Process Document.
Rationale	Eases changes to Vocabulary Elements without breaking existing implementations or affecting other Vocabulary Elements.
Example	"SchemaNameV0.0.0.XSD"

7.3 Common Data Repository

Recommendation	All common data versioned schemas will be kept in a common repository accessible by all work teams.
Rationale	1. Maximize reuse opportunities, 2. Reduce Maintenance/Enhancement problems

8 Schema Architecture

IXRetail Architecture Goals

1. Create useful schema that properly models the domain
2. Create easy to use schema
3. Create easy to expand schemas to add vendor specific needs
4. Clearly identify components contained in standardized messages

Refer to "IXRetail Architectural History 20060320.pdf". Available for download from WWW.NRF-ARTS.org

8.1 Clear Top Level Element

Recommendation	Create one clearly identifiable top level element.
Rationale	One top level element makes the schema easier to use and to understand.

8.2 Venetian Blind Design Model

Recommendation	Create the schema as a set of complex types (building blocks) following the Venetian Blind Model. Then build the schema as a collection of complex types.
Rationale	1. Maximizes reuse. The primary components of reuse are the complex type definitions. 2. Maximizes cohesiveness – the complex types group together related data. 3. Minimizes coupling
Discouraged Examples	Russian Doll Design – characterized by the extensive use of anonymous types. Salami Slice Design – characterized by extensive use of global elements and "ref" type definitions.

8.3 Global "simpleType" and "complexType"

Recommendation	Where domain experts believe a type is likely to be reused, either a simpleType or a complexType should be declared globally in the namespace instead of defining the type anonymously in the Element declaration.
Rationale	Because type names are not usually used in tags (except, for example, as the value of the 'xsi:type' attribute), type names can be constructed with sufficient roots and modifiers to identify the appropriate domain without necessarily causing long tags. This differs from element names, which are always used in tags; so, the alternative of global element names would lengthen tags. When the same set of values is used for the same set of meanings, a reusable type should be defined. Domain experts should verify this sameness of use.

8.4 No Anonymous Types

Recommendation	All type definitions shall be named complexTypes or simpleTypes.
Rationale	Allows for maximum reuse.
Example	<xs:element name="XXX" type="XXXType"/> <xs:complexType name="XXXType"> <xs:sequence> <xs:element name="YYY" type="xs:string"/> </xs:sequence> </xs:complexType>
Discouraged Example	<xs:element name="XXX"> <xs:complexType> <xs:sequence> <xs:element name="YYY" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element>

8.5 Nested Elements Shall Use the "type" Attribute

Recommendation	Schemas should use nested elements that use the 'type' attribute or an inline type declaration (simpleType or complexType) instead of the 'ref' attribute that references a global element.
Rationale	Whenever possible, local element naming should be used so that names can be kept short. The global part of the IXRetail namespace should be reserved for names with well-defined meanings; such global names should be constructed with sufficient roots and modifiers to identify their domain of use. The guideline for "Element versus type" applies when reuse of declarations is suggested.
Example	<xs:element name="XXX" type="XXXType"/> <xs:complexType name="XXXType"> <xs:sequence> <xs:element name="YYY" type="xs:string"/> </xs:sequence> </xs:complexType>
Discouraged Example	<xs:element ref="XXX"> <xs:element name="XXX"> <xs:complexType name="XXXType"> <xs:sequence> <xs:element name="YYY" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element>

8.6 Base Schema

Recommendation	Create one large schema to model the domain.
Rationale	1. Allows for the tailoring of the schema to meet logical groupings. 2. Leveraging the schema in other verticals.

8.7 Publish Logical Groupings of Message Sets

Recommendation	Publish a "logically grouped" schema, for example "Customer" can be split into a "Customer Information" group and a "Customer Loyalty" group.
Rationale	1. Creates a manageable set of published schemas. 2. Reduces the size of the published schema with respect to the size of the Uber Schema. 3. Allows conformance at a logical grouped level.

9 Extending Schemas

Recommendation	Schemas will be extended to allow for vendor specific needs following "IXRetail Extending Schemas Technical Report-20050704.PDF". Available for download from WWW.NRF-ARTS.org
Rationale	Provides for a consistent methodology for adding vendor/retailer specific information to the ARTS standard schemas.

10 Publish "Flattened" Schemas

Recommendation	Prior to publication, all schemas will be flattened. That is, all imports or includes will be resolved by "copying" the included/imported schema(s) into the published schema.
Rationale	1. Reduces versioning issues 2. Reduces tool problems associated with the use of include/import.

11 Deprecation of Elements and Attributes

Recommendation	Annotate schema to identify deprecated element/attribute with note on what is replacing it. a. Identify element/attribute being replaced. b. Identify the version where the element/attribute will be removed.
Example	<xs:complexType name="XXXType"> <xs:annotation> <xs:documentation>Deprecated to be replaced by YYYType in Version 2.0.0</xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="ZZZ" type="xs:string"/> </xs:sequence> </xs:complexType>

12 Document History

Version History

Ver	Date	Sections	Description of Change
1	8/2002	All	Initial Release
2	4/11/2006	All	Document reformatted into the IXRetail standard format. - Added 3.4 to add definitions to the schema - Added 3.8 to solve Unique Particle Attribution error - Added 3.10 to sort out the difference between an attribute and an element - Added 4.5 to define defaults - Added 4.6 to define Boolean Values - Added 4.7 to define date/time - Added 5.5 the use of relative addressing for schema location - Added 6.0 to define how to do versioning - Added all of 7.0 to manage common data - Added 8.2 to identify the schema architecture - Added 8.4 to prohibit the use of anonymous types. - Added 8.6 and 8.7 to create a large schema for use in multiple domains and then tailor release for a specific domain. - Added 9 to link to the extension paper - Added 10 to solve release issues with common data - Added 11 to define a deprecation method

13 Committee Membership

Chairman:

Version 1:

Paul Golick	IBM

14 Domain GLOSSARY

Term	Definition
Facet	Metadata constraints applied to W3C XML Schema data types.
Venetian Blind Model	Disassembles the domain into individual components by creating complex type definitions.